# An Evaluation Method and Comparison of Modern Cluster-based Highly Available Database Solutions

Raju Shrestha[a] and Tahzeena Tandel
*Oslo Metropolitan University (OsloMet), Oslo, Norway*

Keywords: High Availability, Database, Cloud Computing, Replication, Multi-Master, Clustering, Evaluation, Comparison.

Abstract: High availability in cloud computing is a top concern which refers to keeping a service operational and available for the maximum amount of time without downtime. In any given application and service, the high availability of the database plays a critical role in the application's high availability as a whole. Modern cluster-based multi-master technologies provide high availability database solutions through synchronous replication. Different database management systems offer different technologies and solutions for high availability. This paper proposes a comprehensive method for the evaluation of high availability database solutions. As a comparative case study, two modern high availability database solutions which are open-source and available for free use, namely the Percona XtraDB Cluster and the MySQL NDB Cluster are used. Benchmark tests with standard CRUD database operations and analysis of the test results show that no single solution is superior to the other in all scenarios and needs. Therefore, one should choose an appropriate solution wisely depending on the needs for an application or service. The proposed evaluation method would be useful to get insights into which solution is effective for a given application and hence it can be used in making an informed choice among different solutions at hand.

## 1 INTRODUCTION

An application or service is said to be highly available if it is operational and available for a higher than normal period without downtime (Endo et al., 2016). Availability is usually expressed as apercentage of uptime in a given year (e.g., 99.999%) (Piedad and Hawkins, 2008). As most of the modern cloud-based applications and services are database driven, high availability of database or highly availabe database (HADB) is critical to the high availability of the application or service as a whole.

Data replication techniques are primarily used for making databases highly available in distributed systems (Gopinath and Sherly, 2018) such as peer-to-peer systems (Ezéchiel et al., 2017), mesh networking (Milani and Navimipour, 2017) and distributed data management systems (Liu and Vlassov, 2013). Many technologies and solutions have been proposed for achieving database high availability ranging from simple and traditional data replication to clustering for providing as highly available as 99.999%. HADB aims for reduced downtime as well as for reduced response time and increased throughput (Hvasshovd et al., 1995). Different applications demand different levels of availability (Dahiya and Ahlawat, 2015). While mission-critical applications cannot afford any downtime and demand continuous service uptime, i.e., 100% availability of both read and write access. The higher the level of availability needed, the higher will be the level of complexity and cost associated with it.

In recent years, modern cluster-based architectures have gained much attention as alternatives to traditional master-slave architectures because of their promising features to deliver continuous availability and enhanced performance. They are mostly based on multi-master technology and synchronous replication to address data inconsistency, eliminate single point of failure, and use architectures designed to provide higher levels of availability. Percona XtraDB Cluster (Percona, 2022) and MySQL NDB Cluster (MySQL, 2022) are the two examples of major cluster-based technologies that are available open-source.

[a] https://orcid.org/0000-0001-6409-8893

Having many different solutions available, an obvious challenge one has to face is the selection of a suitable solution among them. In this paper, a comprehensive method for evaluating and comparing HADB solutions is provided. We believe the method helps get insight into the different technologies and in turn, helps choose a suitable HADB solution for a given application. As a case study, two popular technologies, Percona XtraDB Cluster and MySQL NDB, Cluster, are evaluated and compared both quantitatively and qualitatively.

## 2 BACKGROUND AND RELATED WORKS

Data replication is a primary approach used for highly available databases. There are basically two data replication technologies: master-slave replication and cluster-based multi-master replication. In master-slave database replication, a master database server handles data writes, while multiple slave servers are used to read data, thus supporting read scalability (Wiesmann and Schiper, 2005; Curino et al., 2010). Any changes in the master database are replicated in the slaves asynchronously. When the master fails, one of the slaves is automatically promoted to the master. Asynchronous replication doesn't guarantee the delay between applying changes on the master and propagation of changes to the slaves, which may cause data inconsistencies when something goes wrong in the master database server in the middle of a transaction. Thus, master-slave replication doesn't guarantee consistency, one among the three: consistency, availability, and partition tolerance in Brewer's CAP theorem (Brewer, 2012). Some database systems support semi-synchronous master-slave replication (Dimitri, 2017) to mitigate the problem. Multi-master cluster-based database replication techniques address data inconsistency problems through synchronous replication. In synchronous replication, users can both read from and write to any of the replicas (Ezéchiel et al., 2017). For high availability databases, different database management systems support different solutions. Section 3 gives an overview of some of the major HADB solutions offered by different Database Management Systems (DBMSs).

A performance-based comparative study between master-slave and cluster-based HADB solutions showed that master-slave replication performs equal or better in terms of throughput and response time (Shrestha, 2017). However, cluster-based solution is superior when it comes to high availability, data consistency, and scalability as it offers instantaneous failover, no loss of data, and both read and write scalability. Another study found that container-based setup performed better compared to virtual machine-based setup (Shrestha, 2020). In both studies, performance was evaluated through Sysbench online transaction processing benchmark tests using two quantitative performance metrics, throughput, and response time.

(Ha et al., 2016) compared performance of HADB in PostgresSQL with two different replication managers, same-containment Keepalived-RepMgr and cross-containment HAProxy-PgBouncer. They defined different workloads for performance testing using Apache JMeter and reported results for throughput, and response time. They concluded that the performance of HAProxy-PgBouncer is better for throughput with load balancing abilities as compared to Keepalived-RepMgr.

Literature study showed that most of the previous studies on HADB solutions use one particular database management system and they mainly use quantitative metrics in their evaluation. In this paper, we tried to address this through a systematic evaluation of HADB solutions both quantitatively and qualitatively, and at the same time focusing on modern cluster-based solutions.

## 3 OVERVIEW OF HIGH AVAILABILITY DATABASE SOLUTIONS

Among open-source databases, MySQL and MariaDB are embedded with solutions such as data replication for data backup, clustering with shared storage, shared-nothing architecture, and clusters replicated in geographical locations. Master-slave architecture has been in the ecosystem for a long time as a HADB solution in databases such as MariaDB, MySQL, and Microsoft SQL Server. MongoDB provides master-slave replication through replica sets, which use elections to support high availability (Gu et al., 2015). Elections occur when the primary becomes unavailable and replica set autonomously select a new primary. MongoDB can also use sharding to distribute large data to multiple instances.

Most of the modern DBMSs support cluster-based multi-master replication for highly available databases. Microsoft SQL Server supports multi-master Peer-to-Peer (P2P) replication[1], in which each node is both a publisher and a subscriber. Data is replicated in multiple nodes and apply the changes to the destination in near real-time. It is a good scaling solution for reads as they can be serviced through multiple nodes. High availability is achieved by routing the writes from the failed node to the live node. P2P replication has an issue in that there could be some latency while applying data changes to other nodes and hence data loss can occur (Mishra et al., 2008). Microsoft SQL Server offers a new high availability solution, called Failover Cluster Instances (FCI) with always-on Availability Groups (AGs). FCI leverages Windows Server Failover Cluster (WSFC), which is a group of independent servers that work together to increase the availability of applications and services to support always-on availability groups (Parui and Sanil, 2016). The always-on availability groups feature is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. An FCI is used to host an availability replica for an availability group.

PostgresSQL supports bidirectional replication (BDR) as an enhanced feature for high availability multi-master solutions (Raja and Celentano, 2022). It uses a standby replica for each master, with a master replica forming a cluster. Data flows from both master to standby and from standby to master with streaming logical replication. BDR, thus, has both physical and logical replication providing strong resiliency. BDRv3 supports fully automatic replication of both Data Manipulation Language (DML)) and Data Definition Language (DDL). Commit At Most Once (CAMO) ensures that any in-flight transactions with unknown states are fully resolved.

Oracle offers various architectures for different level of high availability[2]. Among them, cluster-based Oracle Real Application Clusters (RAC) and Oracle Data Guard is the most comprehensive architecture, which reduces downtime for scheduled outages with premention, detection, and recovering mechanisms from unscheduled outages, and provides the highest level of availability. Oracle RAC is a shared everything database, which uses Oracle Clusterware, a cluster management solution that binds a pool of servers into the cluster, providing maximum write-scalability. It is recommended that Oracle RAC and Oracle Data Guard reside on separate clusters and data centers. Oracle Data Guard supports both synchronous mode for maximum protection and availability of data, and asynchronous mode for better response time in systems with a network bottleneck.

Among others, MariaDB Galera Cluster (MariaDB, 2022), Percona XtraDB Cluster (Percona, 2022), and MySQL NDB Cluster (MySQL, 2022), are three widely used open-source cluster-based multi-master HADB solutions. Percona XtraDB Cluster is a variant of Galera Cluster[3]. We used Percona XtraDB Cluster and MySQL NDB Cluster as a case study in our comparative study in this paper because they are free, modern, and open source and they use different storage engines. The two technologies are described briefly below.

**Percona XtraDB Cluster (PXDB)** is a synchronous multi-master clustering technology, which uses certification-based replication. It uses the Percona server for MySQL and Galera library with write set replication (wsrep). Recommended configuration is to have at least 3 nodes, but one can make it run with 2 nodes as well (Tkachenko, 2012). For data consistency, the XtraDB storage engine, a drop-in equivalent to the InnoDB storage engine with an enhanced set of features, is used and data is kept synced in all the nodes through synchronous replication of Data Defination Language (DDL) and Data Manipulation Language (DML) actions. PXDB uses Galera replication protocol for DML actions and total order isolation for DDL actions to ensure consistency in the cluster (Krunal, 2016). The binary log format is set to row for row-level replication. Certification-based replication is used to avoid transaction conflicts. Figure 1 shows the architecture of Percona XtraDB Cluster.

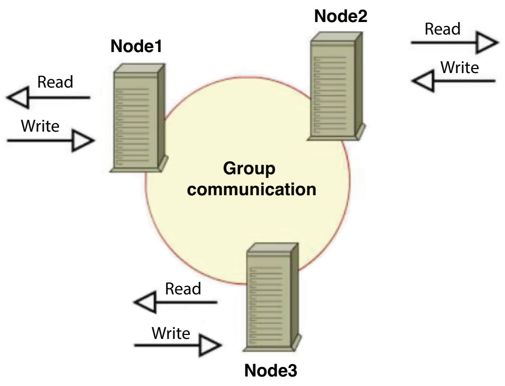

Figure 1: Percona XtraDB Cluster architecture (Tkachenko, 2012).

[1]Microsoft SQL Sever Peer-to-Peer Transactional Replication, https://learn.microsoft.com/en-us/sql/relational-databases/replication/transactional/peer-to-peer-transactional-replication

[2]Oracle High Availability Architectures and Solutions, https://docs.oracle.com/cd/E11882_01/server.112/e17157/architectures.htm

[3]Galera Cluster - A new type of highly consistent MySQL Cluster architecture, https://www.programmersought.com/article/6193909457

**MySQL NDB Cluster (MNDB)** is a HADB solution for MySQL database system with an underlying storage engine used is NDB (Network DataBase), which is implemented using a distributed, shared-nothing architecture with automatic failover and self-healing, providing high levels of availability (MySQL, 2022). It has three different types of nodes: management node – used to monitor, maintain, and manage the cluster, data node – used to store data, and SQL node – used to access data in data nodes. MySQL Servers provide an SQL interface to access the data stored in data nodes. The minimum recommended number of nodes is four: one management, one SQL node, and two data nodes. There can be several node groups; each node group consists of one or more data nodes. However, one can make it work with one data node as well. The cluster is available till one node in each node group is alive. MNDB has a built-in fault tolerance with a heartbeat mechanism, where each table created gets partitioned into data nodes, each data node holding its own primary replica and also a copy of secondary replica. Replication in data nodes is performed synchronously using a protocal called a two-phase commit. (Mether, 2013; King, 2015). Figure 2 shows the MNDB Cluster architecture.

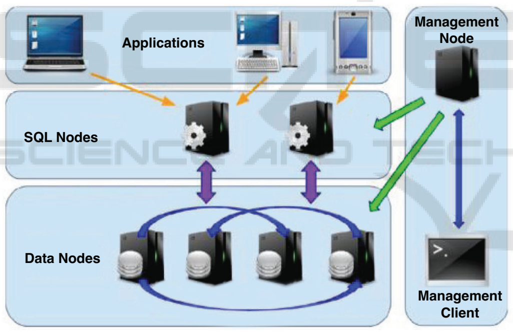

Figure 2: MySQL NDB Cluster architecture (Mether, 2013).

## 4 EVALUATON METHOD

The proposed method for evaluating and comparing HADB technologies or solutions consists of qualitative and quantitative evaluations based on various attributes and metrics. They are described in the following subsections.

### 4.1 Qualitative Evaluation

HADB solutions can be compared qualitatively based on their type and support in terms of five different attributes. They are *architecture*, *replication type*, *scale-out*, level of *availability*, and *automatic failover* support.

**Architecture:** Basically, architecture could be either master-slave or multi-master. Some DBMSs use the term primary-secondary for the master-slave. Architecture also defines the minimum or recommended number of nodes required for a HADB setup.

**Replication Type:** A HADB could offer either asynchronous or synchronous or both.

**Scale-Out:** Scale-out support is an essential feature in a cloud infrastructure. The scale-out attribute defines whether a HADB supports scaling of read-only or both read and write.

**Availability:** Availability attribute indicates the level of database availability. We have defined three levels as low (90% or below), medium (90-99%), and high (above 99%).

**Automatic Failover:** Automatic failover is yet another important feature in a HADB. Some HADBs do not support failover at all, some support automatic failover, while some other offer automatic failover.

Depending on the needs of the application at hand, these attributes help make decisions in the selection of an appropriate HADB solution.

### 4.2 Quantitative Evaluation

A quantitative comparison is done based on performance evaluation of the HADB solutions under benchmark tests with standard CRUD (Create, Read, Update, and Delete) database operations. Most of the previous studies (Ha et al., 2016; Shrestha, 2020) used mainly two performance metrics, *throughput* and *response time* in their performance evaluation. Since resource usage such as CPU and memory are also important factors that are commonly considered while comparing different systems (Birke et al., 2012; Kozhirbayev and Sinnott, 2017), we have added two more metrics, *CPU use* and *Memory use* in our quantitative comparison. The four metrics used are defined as follows.

**Throughput:** Throughput is defined as the number of transactions executed per second (tps). Higher the throughput, the better the performance of the HADB (Krzysztof, 2021).

**Response Time:** Response time (or latency) is the time taken to execute a transaction. A lower response time means better performance.

**CPU Use:** It is the percentage utilization of CPUs. The lower the CPU use with similar setups, the better the system is.

**Memory Use:** It is the percentage utilization of system memory. Just like CPU use, a lower Memory use with similar setups means a better system.

Based on these performance metrics obtained through experiments with the two HADB setups using MySQL NDB Cluster and Percona XtraDB Cluster, the two HADB solutions are evaluated and compared.

## 5 EXPERIMENTAL SETUP AND TESTS

This section describes the implementation and setups of the two HADBs, and benchmark and failover tests conducted to evaluate and compare them.

**HADB Setups:** Percona XtraDB Cluster (PXDB) and MySQL NDB Cluster (MNDB) based HADBs are set up on the OpenStack cloud at the university (see Figures 3 and 4). Nodes in both setups are implemented with a virtual machine (VM), created in a private network. Necessary security rules and ports are opened to allow TCP/IP communication between them. The same specification of the operating system: Ubuntu 18.04, CPU: 1 VCPU, Memory: 2 GB, and Storage: 20GB was used for all the database nodes. For a fair comparison, the same total number of VMs (four) were used in the two setups. The PXDB setup has four database nodes and the MNDB setup has two SQL nodes and one node group consisting of two data nodes. MySQL NDB Cluster version 8 and Percona XtraDB version 8 were used in the experimental setups. In both setups, a server with HAProxy[4] was used as a load balancer as well as an application server. ClusterControl[5], a database management tool, was used to deploy, manage, and monitor clusters.

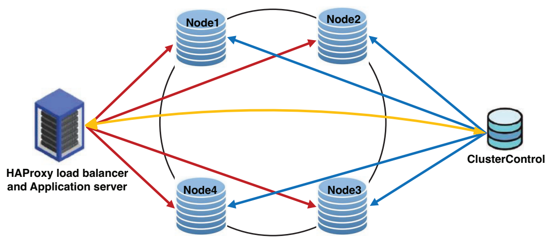

Figure 3: Percona XtraDB Cluster (PXDB) setup.

**Benchmark Tests:** Sysbench v1.0.20[6], a popular and powerful database benchmarking tool, was used to perform various benchmark tests under intensive

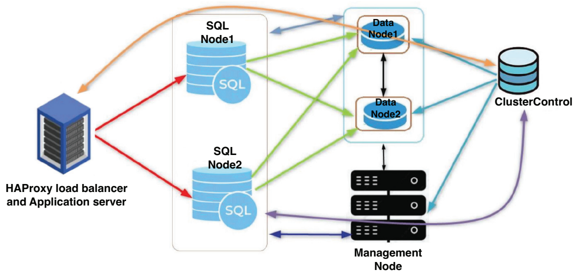

Figure 4: MySQL NDB Cluster (MNDB) setup.

loads. Four different benchmark tests, ReadOnly, ReadWrite, Update, and Delete, were performed using the Online Transaction Processing (OLTP)) benchmark that comes with Sysbench. OLTP was used as it is close to common real-world dynamic database-driven web applications. OLTP ReadWrite consists of a mix of about 70% read (SELECT) and 30% write (INSERT, UPDATE and DELETE). In these tests, a test database was prepared with 50 tables of size 10000 rows, which took about 134MB of disk space. The size was set so that it fits well in the storage buffer pool in Percona XtraDB and data memory in MySQL NDB.

Benchmark tests were run for different numbers of threads (8, 16, 32, 64, 72) and each test was run for 120 seconds. Tests were repeated for five times and average throughput and response time values were used in the results and analysis. CPU and Memory utilization were obtained using `s9s` ClusterControl CLI command (Ashraf, 2017). For each test run, average CPU and Memory usage were computed for values taken every 30 seconds for two minutes. CPU and Memory use amount to the resource consumption by database engines, connection, communication, data replication and synchronization between the nodes, and operating system.

**Failover Test:** The two HADBs were also tested for failover scenarios to see how they behave in case of failures. PXDB was tested by abruptly disconnecting one and two nodes from the network, while MNDB was tested by disconnecting one SQL node and one data node.

## 6 RESULTS AND DISCUSSION

Results from the study are presented and discussed here in three different parts: benchmark test results, failover test results, and qualitative evaluation.

**Benchmark Test Results:** Figures 5 and 6 show benchmark test results in terms of throughput and response time respectively.

[4]HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer, http://www.haproxy.org

[5]ClusterControl, https://docs.severalnines.com/docs/clustercontrol

[6]Sysbench, https://github.com/akopytov/sysbench

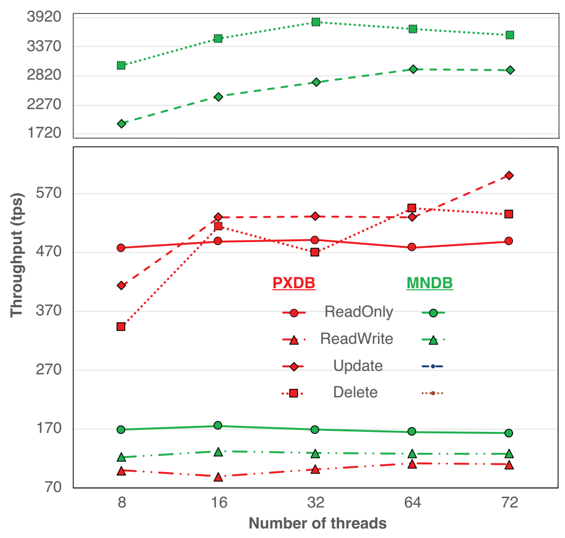

Figure 5: Benchmark test results in terms of throughput.

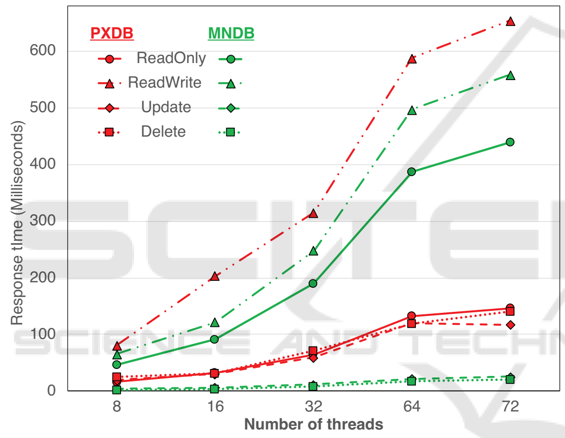

Figure 6: Benchmark test reults in terms of response time.

We observed that in ReadOnly and ReadWrite tests on both HADBs, throughput didn't increase but rather decrease in some cases after increasing the number of threads beyond 16 (see Figure 5). This is because of row locking and internal resource contention with the further increase in the threads. In the case of Update and Delete tests, throughput still increases slightly beyond threads 16, possibly due to lesser resource contention. Since ReadOnly and ReadWrite are the two most common database operations in OLTP applications, we consider 16 threads as optimal in terms of throughput in our setups. In terms of response time, as expected, response time increases with the increase in the number of threads in all the tests, though with different degrees. ReadWrite tests have the lowest throughput and highest response time in both HADBs. This is because when data is written, it needs to be replicated and synchronized with the other nodes in a cluster which takes more time and hence lowers throughput and increased response time.

Among the two HADBs, PXDB had significantly higher throughput in ReadOnly tests compared to MNDB, but the scenario was opposite in Update and Delete tests. Both HADB performed more or less the same in ReadWrite tests. Similar performance was observed from the two HADBs in terms of response time also.

Figures 7 and 8 show CPU and Memory usage in the two HADBs under four benchmark tests. As anticipated, CPU use increased gradually with the increase in the number of threads in both HADBs under all four tests. Similar behavior was observed with PXDB in ReadWrite and Delete tests. While memory use remained more or less constant in other tests in both HADBs.

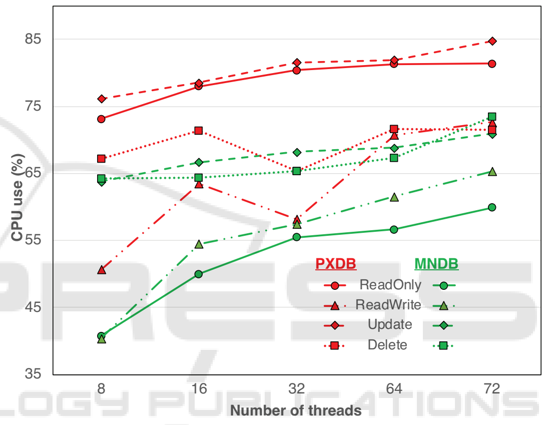

Figure 7: CPU use in different benchmark tests.

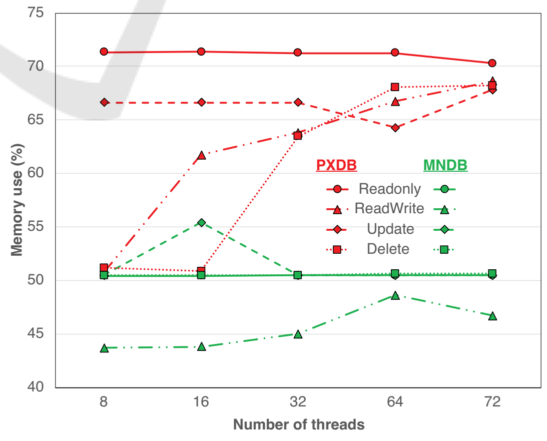

Figure 8: Memory use in different benchmark tests.

MNDB used less or similar CPU and Memory compared to PXDB in all the tests, indicating its superiority in terms of both CPU and Memory use.

To provide a general overview of the performance of the two HADBs, Table 1 gives benchmark test results in terms of all four metrics when the number of threads was 16. From the table, we see that MNDB outperforms PXDB in all the tests except in ReadOnly test, where PXDB has higher throughput and lower response time.

Table 1: Comparative results from benchmark tests with the two HADBs when the number of threads was 16. Numbers shown in green indicate better performance over the corresponding ones shown in red.

| HADB | Benchmark test / Metric | ReadOnly | ReadWrite | Update | Delete |
|---|---|---|---|---|---|
| PXDB | Throughput (tps) | 489 | 89 | 530 | 515 |
| | Response time (ms) | 33 | 204 | 30 | 32 |
| | CPU use (%) | 78 | 64 | 79 | 71 |
| | Memory use (%) | 71 | 62 | 67 | 51 |
| MNDB | Throughput (tps) | 175 | 131 | 2434 | 3529 |
| | Response time (ms) | 91 | 122 | 7 | 5 |
| | CPU use (%) | 50 | 54 | 67 | 64 |
| | Memory use (%) | 50 | 44 | 55 | 50 |

**Failover Test Results:** Failover tests show that PXDB continued to be operational and was available even when two random nodes were disconnected, keeping the three minimum nodes as required by PXDB. No failure-related hiccups were observed. MNDB was also available and working normally when one SQL node and one data node were disconnected. This test confirmed that both HADBs supported automatic failover making them highly available as long as the minimal number of nodes are there.

**Qualitative evaluation:** Comparing the two HADBs in terms of the qualitative attributes described in Section 4, both PXDB and MNDB are based on multi-master architecture, with the former requiring three recommended nodes while the latter requires four recommended nodes (1 management node, 1 SQL node, and 2 data nodes). Both HADBs use synchronous replication, support both read and write scaling, with built-in automatic failover, thus providing higher level of availability.

Based on the qualitative and quantitative evaluation of the two HADBs, one can select an appropriate solution according to the need for the application or service at hand. For a side-by-side comparison between different solutions (PXDB and MNDB in our case study), we can create a simplistic comparative table as shown in Table 2, where numeric metric values are translated as low, moderate, and high.

The experiments conducted in the case study with two HADB solutions, PXDB and MNDB, were based on the setups in the OpenStack cloud setup at the university with certain resource limitations. However,

Table 2: Comparion table between two HADBs. Better indicators are shown in green, while relatively worse indicators are shown in red.

| Attributes | PXDB | MNDB |
|---|---|---|
| Architecture | Multi-master; Min. 3 nodes | Multi-master; Min. 4 nodes |
| Replication type | Synchronous | Synchronous |
| Scale-out | Both read write | Both read write |
| Data consistency | Yes | Yes |
| Availability | High | High |
| Throughput for Read only | High | Low |
| Throughput for ReadWrite | Low | High |
| Throughput for Update & Delete | Low | High |
| Response time for ReadOnly | Low | High |
| Response time for ReadWrite | High | Low |
| Response time for Update & Delete | High | Low |
| CPU use | High | Moderate |
| Memory use | High | Moderate |

the results from the benchmark tests provide a general idea about the performance of the two HADBs.

Since none of the two HADBs shows superior performance in all aspects, one should make a wise decision while selecting a HADB. For example, if an application will have significantly much more read than write, update, and delete in its database, then it'd be wise to select PXDB over MNDB as it has significantly better throughput and response time. Otherwise, MNDB would be a better choice.

## 7 CONCLUSION

The proposed evaluation method comprising both qualitative and quantitative evaluation provides a systematic and comprehensive evaluation of high availability database technologies and solutions. Both the modern cluster-based high availability database solutions tested and compared in our case study, Percona XtraDB Cluster and MySQL NDB Cluster use multi-master architecture and synchronous replication enabling data consitency and high availability and support both read and write scaling. Performance-wise Percona XtraDB Cluster has been found to offer high throughput and low response time for ReadOnly tests while MySQL NDB Cluster has shown superior performance with high throughput and low response time and lower CPU and Memory use in ReadWrite, Update, and Delete tests. By following the proposed evaluation method, one can evaluate, compare, and select a suitable solution from among different solutions available at one's disposal, for a given application or service.

## REFERENCES

Ashraf, S. (2017). The Command Line Interface to ClusterControl. Serverlines, Online: https://severalnines.com/blog/command-line-interface-clustercontrol/.

Birke, R., Chen, L. Y., and Smirni, E. (2012). Data centers in the cloud: A large scale performance study. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 336–343.

Brewer, E. (2012). Pushing the CAP: Strategies for Consistency and Availability. *Computer*, 45(2):23–29.

Curino, C., Jones, E., Zhang, Y., and Madden, S. (2010). Schism: A workload-driven approach to database replication and partitioning. *Proc. VLDB Endow.*, 3(1-2):48–57.

Dahiya, S. and Ahlawat, P. (2015). Choosing between high availability solutions in Microsoft SQL Server. 4:387–391.

Dimitri, V. (2017). Different types of MySQL replication solutions. Percona Database Performance Blog, Online: https://www.percona.com/blog/2017/02/07/overview-of-different-mysql-replication-solutions.

Endo, P. T., Rodrigues, M., Gonçalves, G. E., Kelner, J., Sadok, D. H., and Curescu, C. (2016). High availability in clouds: Systematic review and research challenges. *Journal of Cloud Computing*, 5.

Ezéchiel, K. K., Agarwal, R., and Kaushik, B. N. (2017). Synchronous and asynchronous replication. In *International Conference On Machine Learning and Computational Intelligence*.

Gopinath, S. and Sherly, E. (2018). A comprehensive survey on data replication techniques in cloud storage systems. *International Journal of Applied Engineering Research*, 13:15926–15932.

Gu, Y., Wang, X., Shen, S., Ji, S., and Wang, J. (2015). Analysis of data replication mechanism in NoSQL database MongoDB. In *Proeedings of the IEEE International Conference on Consumer Electronics (ICCE)*, pages 66–67, Taiwan. IEEE.

Ha, L. Q., Xie, J., Millington, D., and Waniss, A. (2016). Comparative performance analysis of PostgreSQL high availability database clusters through containment.

Hvasshovd, S. O., Torbjørnsen, O., Bratsberg, S. E., and Holager, P. (1995). The ClustRa Telecom Database: High availability, high throughput, and real-time response. In *Proceedings of the 21st International Conference on Very Large Data Bases*, VLDB '95, pages 469–477, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

King, C. (2015). The trade-offs between MySQL cluster and Master/Slave content. White paper, https://www.datavail.com/resources/the-trade-offs-between-mysql-cluster-masterslave.

Kozhirbayev, Z. and Sinnott, R. O. (2017). A performance comparison of container-based technologies for the cloud. *Future Generation Computer Systems*, 68:175 – 182.

Krunal, B. (2016). How Percona XtraDB cluster certification works. Percona Database Performance Blog, Online: https://www.percona.com/blog/2016/04/17/how-percona-xtradb-cluster-certification-works.

Krzysztof, K. (2021). How to measure database performance. Online: https://severalnines.com/database-blog/how-measure-database-performance.

Liu, Y. and Vlassov, V. (2013). Replication in distributed storage systems: State of the art, possible directions, and open issues. In *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 225–232.

MariaDB (2022). What is MariaDB Galera Cluster? Online: https://mariadb.com/kb/en/what-is-mariadb-galera-cluster/.

Mether, M. (2013). MySQL cluster internal architecture. Online: https://mariadb.com/files/MySQL_Cluster_Internal_Architecture_-_MariaDB_White_Paper_-_08-26-13-001_1.pdf.

Milani, B. A. and Navimipour, N. J. (2017). A systematic literature review of the data replication techniques in the cloud environments. *Big Data Research*, 10:1–7.

Mishra, S., Mehra, P., Rao, V., and Ashok, G. (2008). Proven SQL Server architectures for high availability and disaster recovery.

MySQL (2022). MySQL NDB cluster 8.0. Online: https://dev.mysql.com/doc/mysql-cluster-excerpt/8.0/en/mysql-cluster-overview.html.

Parui, U. and Sanil, V. (2016). Introduction to Windows Server Failover Clustering. *Pro SQL Server Always On Availability Groups*, pages 45–52.

Percona (2022). Percona XtraDB cluster 8.0 documentation. Online: https://www.percona.com/doc/percona-xtradb-cluster/LATEST/index.html.

Piedad, F. and Hawkins, M. (2008). *High availability: Design, techniques, and processes*. Harris Kern's Enterprise Computing Institute Series. Prentice Hall.

Raja, Y. and Celentano, P. (2022). PostgreSQL bi-directional replication using pglogical. Online: https://aws.amazon.com/blogs/database/postgresql-bi-directional-replication-using-pglogical.

Shrestha, R. (2017). High availability and performance of database in the cloud: Traditional Master-slave replication versus modern Cluster-based solutions. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, CLOSER, pages 413–420, Portugal. SciTePress.

Shrestha, R. (2020). Performance of cluster-based high availability database in cloud containers. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science*, CLOSER, pages 320–327. SciTePress.

Tkachenko, V. (2012). Percona XtraDB Cluster Feature 1: High availability. Online: https://www.percona.com/blog/2012/01/17/xtradb-cluster-feature-1-high-availability/.

Wiesmann, M. and Schiper, A. (2005). Comparison of database replication techniques based on total order broadcast. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):551–566.