SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 7, NO. 5
ISSN: 2054 -7420

Implementation and Verification of High Data Availability on Database

Yen-Jen Chen , Han Tsai
Ming Chi University of Technology, Taiwan, R.O.C.
yjchen@mail.mcut.edu.tw; alsontsai@gmail.com

ABSTRACT

This study provides a low-cost and high-availability database management system architecture for general Small/Medium Enterprises (SMEs) to backup database data access. To prove that the proposed architecture can support the high availability of the database, and can effectively avoid data loss in memory caused by *failovers*, this study applies the main test method of powering off the virtual machine and verified three cases on two commonly used databases MySQL and PostgreSQL: Case 1 proves that this study combines the database native disaster recovery mechanism to effectively achieve high availability of the database. Case 2 proves that it effectively controls the WAL (Write Ahead Log) of the PostgreSQL database and Redo log mechanism of the MySQL database, so that data correctness is maintained during failovers. Case 3 proves that it can analyze and control the timing of the database in writing data in the cache memory to the hard disk. This study also designed a failover process to avoid data loss during failovers due to no enough time to write the data in the cache memory back to the hard disk; and finally to realize the high-availability of the database management system architecture in a practical way.

Keywords: Database, DBMS, High Availability, Failover, DRBD

1 Introduction

1.1 Motivation

Information technology infrastructure is an indispensable part of the information environment of modern SMEs. The database (DB) for storing enterprise operational data is the most important part of the information environment because it relates to the lifeblood of a company. In the course of business operation, in order to avoid business interruption, enterprises adopt a backup mechanism so that the network and system services in the information environment will be uninterrupted. Database service is no exception. However, during the operation of the backup mechanism of the database, since the database software has a mechanism for storing data in the memory to enhance the operational efficiency of the system; this mechanism will also lead to data loss due to the switching of the database server while the backup mechanism is in operation.

Therefore, the subject of this study is data loss caused by the operation of database backup mechanism, and we discuss about high data availability mechanism on database management systems.

DOI: 10.14738/tnc.75.7292
Publication Date: 17th November 2019
URL: http://dx.doi.org/10.14738/tnc.75.7292

1.2 Objective

In the past, the high availability mechanism of the database was divided into server parts and database segments, and the two segments need to be separately constructed and maintained. Considering the data consistency of the database, it is necessary to modify the internal source code or use the database communication protocol. In order to achieve this effect, in addition to a certain understanding of the source code, it is necessary to do a certain degree of secondary development, and the complexity is relatively improved. Nowadays, the database of the high availability mechanism is not commonly. One architecture can only achieve a database high availability mechanism for a database, and it needs to pay a high amount of software authorization. Therefore, the architecture flexibility and scalability of the enterprise at this part are in limited This study uses the Clustering and Virtualization technologies of open source licensed Linux to achieve a system platform of high availability, scalability, flexibility, and efficiency, in addition to support commercial software operations for business operations. Clustering is mean that at least two homogeneous devices share a service, and the devices can support each other or share the work. Clustering technology can make the platform have high availability and scalability; On the other hand, Virtualization refers to virtualizing a physical machine into multiple virtual machines, so that hardware resources can be fully utilized to achieve so-called efficiency. Since virtual machines can be transferred between different physical machines, managers can flexibly schedule them. Thus achieving the so-called flexibility. According to DB-Engines, the first global database ranking list [1] in the second half of 2018 was released. The top 4 were found to be 1. "Oracle", 2. "MySQL", 3. "Microsoft SQL Server", and 4. "PostgreSQL". This study uses two high market share databases: PostgreSQL and MySQL to discuss the data high availability mechanism of the database system.

1.3 Organization

This paper is organized as follows. In the second section, the research is explained on the high availability mechanism of the database in recent years. In section 3, the system architecture and research method of proposed scheme is presented. Section 4 shows the analysis of the experimental results and three cases of verifying that the high availability architecture is indeed implemented. Finally, section 5 gives the conclusions of this study.

2 Background

In recent years, among studies on the high availability mechanism of database, C. Jaiswal et al. [2] proposed a mechanism that regardless of whether the backup mechanism is activated, the database system will complete the write operation and write the data to the disk. That mechanism is called Always Ahead Processing (AAP). With this mechanism, high availability on database management system is achieved; RH de Souza et al. [3] proposed an architectural model for assessing the reliability, availability, and maintainability of decentralized databases. The model is based on an approach similar to service quality, and provides database services based on the Service Level Agreement (SLA), to increase the availability of the database.

The architecture design in this study has a high availability mechanism, so there will be two DBMS servers, and the servers are connected to the storage via the ISCSI (Internet Small Computer System Interface) [4][5] protocol. All data is uniformly stored in the storage unit to maintain data consistency. However, in order to back up the data, the data is synchronized to another ISCSI storage by means of DRBD (Distributed Replicated Block Device) [6] distributed mirror. Backup between the two DBMS systems is achieved

through the connection management software Keepalived [7] for the failover function [8][9]. Relevant information such as technologies, systems, and open source software used are as follows:

2.1 DRBD

DRBD (Distributed Replicated Block Device) is based on a distributed storage system on the Linux platform. It is also a technology for instantly synchronizing the data content of the inter-host block storage device through the network. It can be regarded as a kind of network RAID1, that can use storage devices on two independent server hosts through the network as RAID1. Management can be done via the two hosts, and the two sides of RAID1 data are placed respectively on the storage device of these two hosts. Therefore, even if the storage device of one of the hosts is damaged, it will not affect client end usage. After the damaged host is repaired, the data can be recovered by synchronizing the data.

2.2 ISCSI

Internet SCSI (Internet Small Computer System Interface) is a standard developed by the IETF. The principle is to use the network to transport SCSI commands for data transmission. Block-level data transmission is used. The storage device and the host can be divided into two parts. The ISCSI Target, that is the storage device side, is used to store data on the ISCSI disk array. The ISCSI Initiator, that is the client side which uses the Target. The storage space provided by Target can only be used by installing the Initiator.

2.3 Keepalived

A high availability solution is usually used to avoid single-node failures. The mechanism is generally divided into two servers, a primary server and the secondary (i.e. backup) server. To maintain the connection, the primary server sends specific messages to the backup server. When the backup server cannot receive the message, the backup server will take over the virtual IP and continue to provide services to ensure high availability.

2.4 Failover

Whenever a node in a cluster is unavailable, the current service providing node's resources is switched to an available node. A procedure that moves a service from an active node in a cluster to a passive node. It is also a backup. If one or more of the cluster nodes fail, the other nodes will start to take over and continue to provide the service through a procedure called "Failover". There will also be at least one monitoring node responsible for monitoring to confirm that the service is working properly. If the service is not working properly, the user's connection to the service will be switched to another node.

2.5 WAL for PostgreSQL

WAL (Write ahead log) [10] is a technique used in relational database systems to provide Atomicity and Durability, where both are in database ACID properties. The database system software has a mechanism for storing data in the memory to improve the performance of the system. If the database server is unexpectedly shut down, the data in the memory will lost. Then, when the database service is interrupted by power, when the database is restored, it is restored to the hard disk according to the recorded content of the WAL, so that the data remains consistent.

2.6 Redo log for MySQL

MySQL's Redo log [11] and PostgreSQL's WAL have the same effect. If the database server is shut down unexpectedly, the data in the memory is lost. Then, when the database service is interrupted, when the database is restored, the contents of the Redo log are restored to the hard disk to keep the data consistent.

3 Research Methods

In this research, we use the database, MySQL and PostgreSQL, to verify the proposed scheme. The database management systems (DBMS) and databases are stored separately on servers and storage units; the server utilizes ISCSI to connect to storage; for the system backup mechanism, two DBMS servers utilizes high data availability mechanisms to toggle one on active and the other on standby. For the data backup mechanisms, the two storage units are separated as primary and secondary storage with both systems using DRBD mirroring mechanism to achieve data synchronization; in other words, whenever DBMS writes data into primary storage, DRBD synchronizes the data into secondary storage. The further details for the proposed design is shown in Fig. 1.

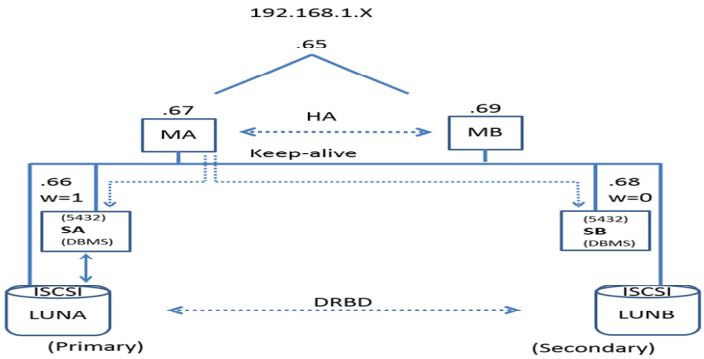


Figure. :1 Database System and Network Structure

MA and MB are master servers while SA and SB are salve servers; the slave is the DBMS server responsible accessing and computing of database. One of the masters (i.e. MA and MB) is active while the other is on standby; the active master always dispatches the database service request to a highly weighted (w=1) slave, and thus the low weighted (w=0) slave is always on standby. The default weight value of SA and SB are 1 and 0 respectively, and when database services of the SA cannot be provided normally, the master will change the weight of SB from 0 to 1 and dispatches the database service request to SB to ensure no data is lost during the failover process. This study was implemented in the PostgreSQL and MySQL database system using its disaster recovery mechanism WAL and Redo log to prevent data loss during failovers. It records all write activities of PostgreSQL and MySQL by recording writing to ensure that the time of WAL and Redo log doesn't fall behind the database. The WAL and Redo log and database are placed in storage together. Since the database cache memory cannot be shut off, DBMS writes data in the memory into database when shutting down normally, aside from the periodic writing of data in the memory into the database. This is used as a reference to write a control program in the masters responsible for activating and shutting off DBMSs in order so that when a failover occurs, the control program will link the database to a new highly weighted slave, say, SB after shutting off DBMS in the other slave, say, SA. If DBMS cannot be shut off, SA will be forced to shut down for flushing the database cache

memory. After shutdown is confirmed, the program activates SB's DBMS to connect to the database and perform the unwritten log in WAL and Redo log to avoid the loss of data in memory having not yet to be written into the database.

The storage DRBD used in this paper adopts the single master mode, which refers to any resource at any specific time. There is only one master node in this two-node cluster. Therefore, the DRBD has two states: Primary and Secondary. As shown in Figure 2, Users' requests will generally be handled by the Primary. The Secondary node is in the Standby state and is responsible as a real-time backup for the Primary. When a hardware failure occurs in the Primary, there is an immediate backup of the data in the Secondary that can be restored.

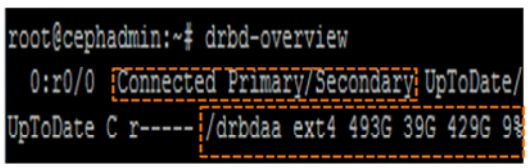


Figure. 2: DRBD status display

In order to achieve High Availability (HA), the cluster monitoring software Keepalive in this study detects the database's udp port. When the SA cannot provide services, the master will trigger the control of weight values through the utility "notify_down" as shown in Figure 3.

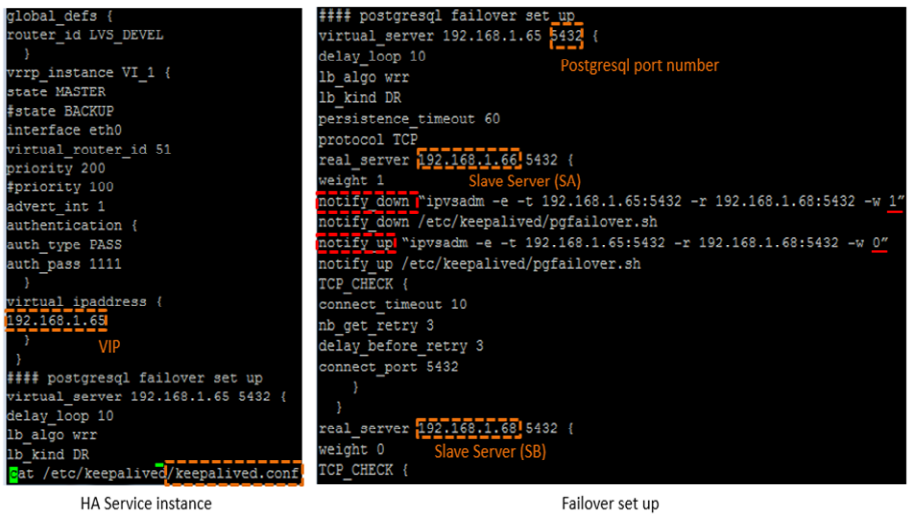


Figure. 3: Keepalived Failover set up

Figure 3 shows the Keepalived profile with control settings of connections and weights. Set the SB weight value from 0 to 1, then the connections are directed to SB, which continues to provide the database service for users. If the SA database returns to normal, with "notify_up" trigger to set the SB weight value from 1 to 0, then connections will then be directed back.

In this study, the architecture integrates multiple open source software and database services (PostgreSQL and MySQL). Usually every project here uses memory (cache). To avoid data loss caused by Failover when the data is in memory, the ISCIS cache has been closed. By changing parameters in the ISCSI Target profile, set the IOMode to wt to change the data writing mode as shown in Figure 4.

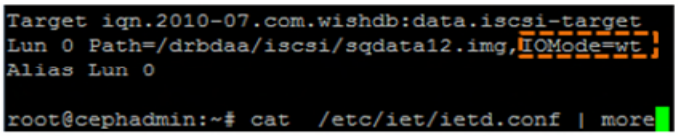


Figure. 4: Data Access Mode Settings in an ISCSI Profile

IOMode is set to wt so that data will not pass through the memory, but will be directly written to the hard disk. Some software including the operating system can not close the cache, such as PostgreSQL and MySQL. PostgreSQL and MySQL does not read and write directly to the hard disk. Instead, it puts the data in the hard disk into the Shared Buffers (cache), and uses the LRU (Least Recently Use) algorithm to make the data access complete in memory to improve the performance. Data will then be flushed to the hard disk at regular intervals.

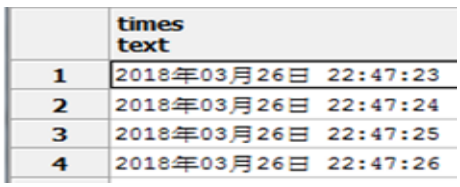
4 Experiment and Verification

The purpose of this study is to ensure the high reliability and the status of the data transferred to the database. There will be three cases in which a failover will be caused by powering off the virtual machines, and the consistency of the data will be compared. The content and principles of each case will be explained below:

Case 1 : Failover data consistency

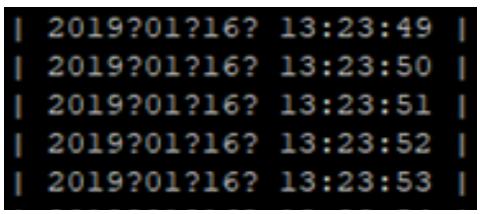
Figures 5 verify the consistency of the data before and after failover. A dataset is transmitted continuously every second, and the serial number and time recorded in the data is used to verify whether the data is consistent. During the data transmission, the physical machine forces the virtual machine to power off via the command “virsh destroy” to cause a Failover. Data in the user end and the database end are compared for consistency.

(a) User end



(PostgreSQL) (b) Database end

(b) User end

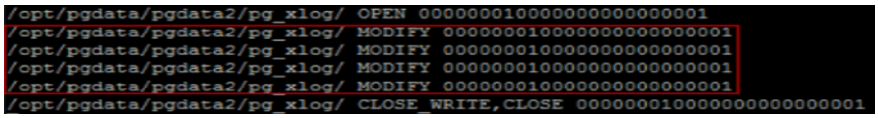


(MySQL) (b) Database end

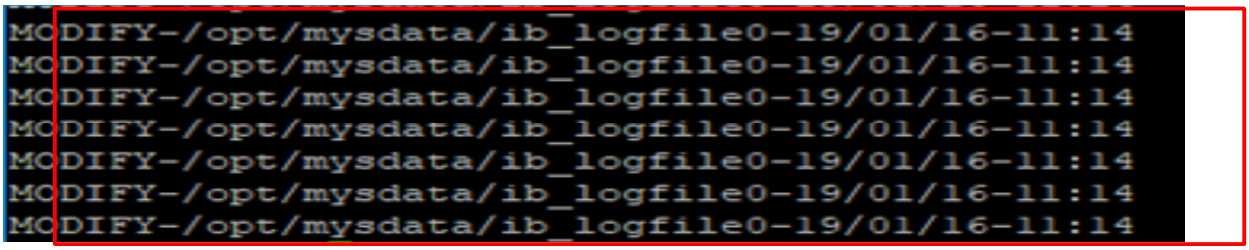
Figure. :5 (a), (b) Comparison of user-end and database-end’s data after a failover

It can be confirmed from Figure 5 that even if an unexpected accident (power outage) occurs, the data can still remain consistent. The key to maintaining consistency is the WAL and Redo log, which records in advance the dataset transmitted by the user into the database. Even if the data is lost in the memory, data can still be recovered through the records in the WAL and Redo log. In order to verify the function of

the WAL and Redo log, this study uses the file system monitoring software Inotifywait [12] to monitor the directory where the WAL and Redo log are located as shown in Figure 6.



(PostgreSQL)



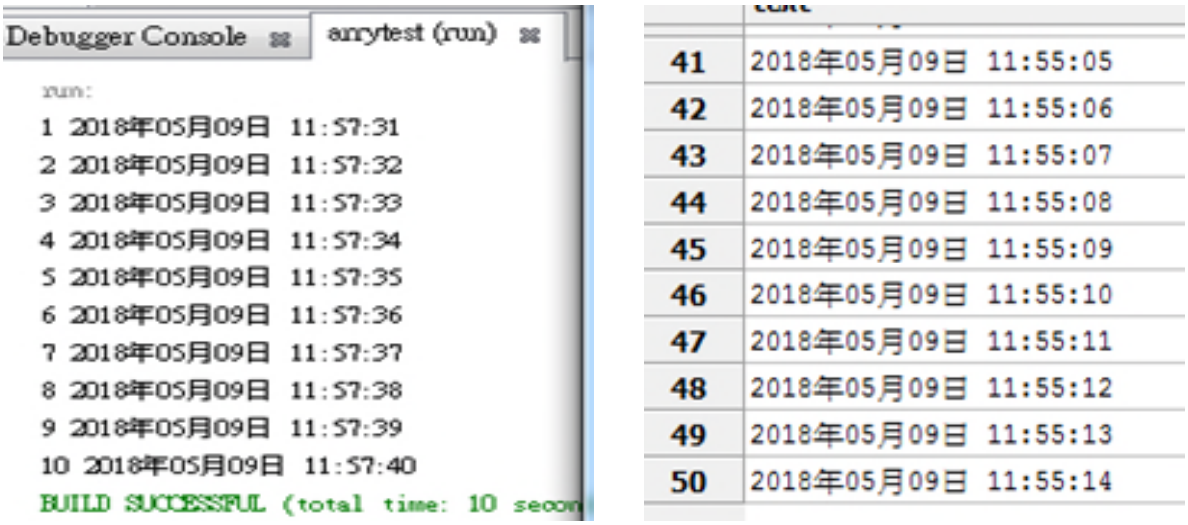
(MySQL)

Figure. :6 Inotifywait WAL and Redo log record screen

It can be confirmed from Figure 6 that when the user transmits a dataset, a modification record will be recorded in the WAL and Redo log file. Therefore, although the data has not been stored in the hard disk, when a Failover occurs, master will connect the database to SB. SB can also recover data through WAL and Redo log.

Case 2: Failover data recovery verification

In Case 1, during continuous data transfer, the virtual machine is powered off to cause a Failover and the data consistency is verified, so as to confirm that a data loss can be recovered by WAL and Redo log. However, in Case 2, only PostgreSQL can be tested because MySQL writes memory data back to its database with an interval not bigger than 1 second. Before transferring the data, copy WAL to the temporary folder first. The WAL saved in this folder has not recorded the data transferred yet. Then 10 datasets are transferred to the database. At this time, because there is a record in WAL about the 10 datasets just transferred, even after the failover, the data will remain consistent. Therefore, after the data is transferred, copy the WAL in the temporary folder to overwrite the WAL in PostgreSQL. Then the virtual machine is powered off to cause a Failover. Afterwards, verify again the state of the data on the user end and database end as shown in Figure 7.



(a) User end (b) Database end

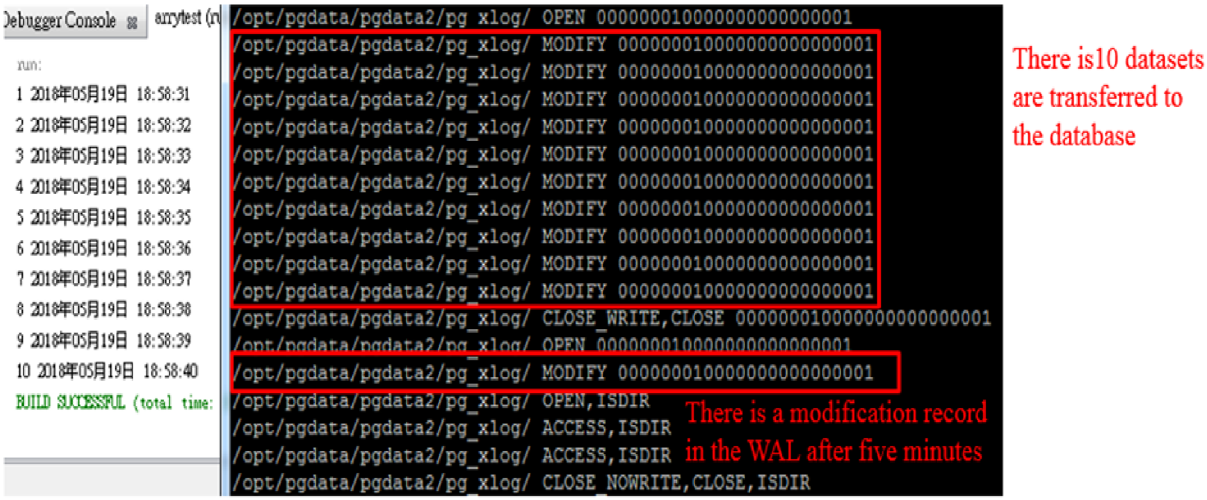
Figure. :7 (a), (b) Comparison of user-end and database-end data after a Failover

It can be seen from Figure 7 that after the failover, the data in the database-end is inconsistent with the user-end. The reason is that when the data is transferred to the database, the system will store the frequently used data in the memory for access without directly storing it into the database, and copy the WAL that has not recorded the incoming data to overwrite the WAL in PostgreSQL. At this time, if Failover occurs, since the data is not stored into the database, nor has WAL recorded the incoming data, a data loss will be caused. After verifying a Failover by replacing WAL, PostgreSQL uses WAL to recover lost data in memory.

Case 3 : Checkpoint function data is written to the hard disk for verification

Checkpoint [13] is one of PostgreSQL's and MySQL's checkpoints for writing data back to the hard disk. It will write the data in the memory back to the hard disk at intervals, based on the time set by the user. The PostgreSQL default is every five minutes, while the MySQL default is every second. Since the MySQL setting is forced to one second, MySQL is hard to be verified in Case2. After the data is written back to the hard disk, the checkpoint record will be written to the WAL and Redo log. After the database service is restarted, the latest checkpoint will be loaded, and the write operations after this checkpoint will be re-executed to ensure that the data will not be lost. While the data before this point has been written back from the memory to the hard disk by Checkpoint, therefore no action is required.

This is a recovery mechanism that PostgreSQL and MySQL does to avoid data loss. In order to verify that the data is actually written to the hard disk after the checkpoint, the verification steps is as below. The software Inotifywait monitors the folder where the WAL and Redo log are located to observe its status. Before the data is transferred, copy the WAL and Redo log to the temporary folder, and then transfer 10 datasets to the database. Wait for five minutes until Checkpoint is executed. After that, the execution record is written into the WAL and Redo log as shown in Figure 8.



(PostgreSQL)

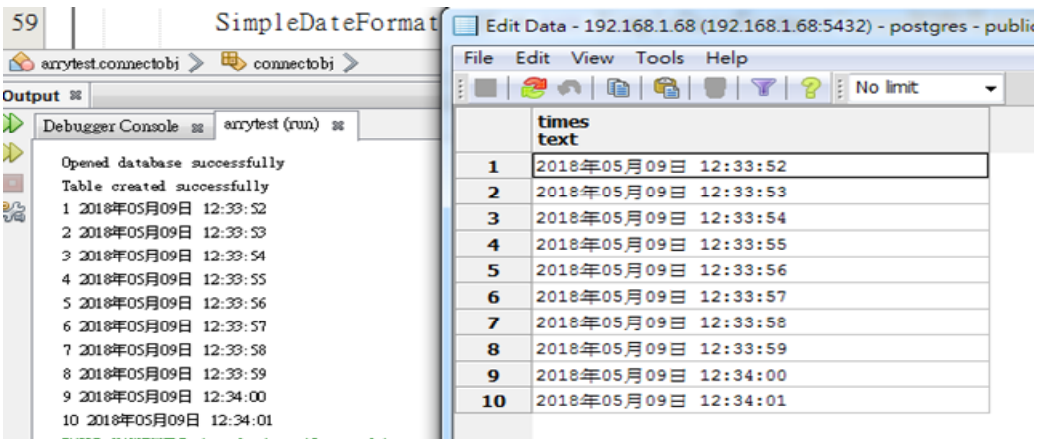
(MySQL)

Figure. :8 Inotifywait Checkpoint record screen

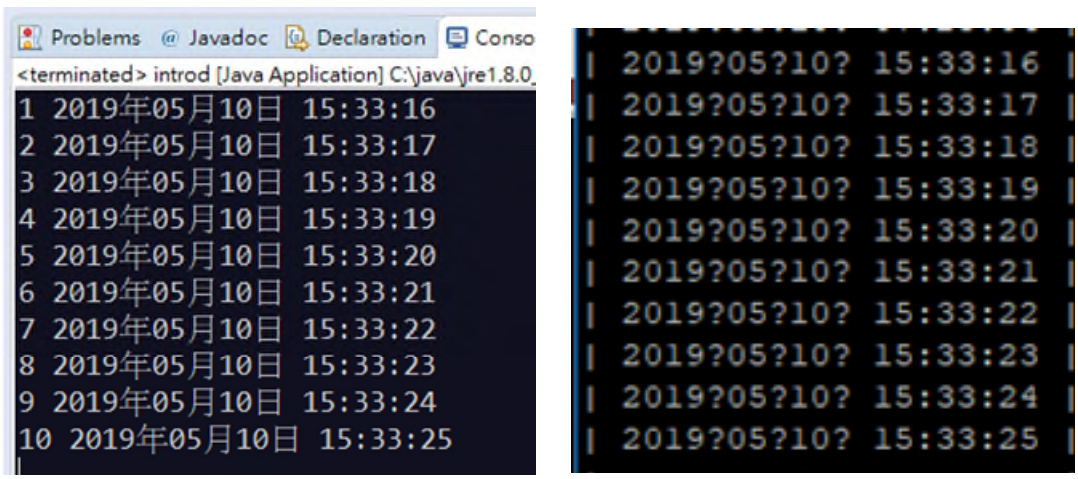
For PostgreSQL, it can be observed from Figure 8 that there is a modification record in the WAL after five minutes. This modification is to record the data in the memory being written back to the WAL after the

Checkpoint execution is completed. In order to verify that the data in the memory has been written back to the hard disk by Checkpoint, again, use the WAL which has not yet recorded the transferred data to overwrite the WAL in PostgreSQL. Power off the virtual machine to cause a Failover. Afterwards, the master will direct the database connection to SB to reconfirm the status of the data at user-end and the database-end as shown in Figure 9.

For MySQL, it can be observed that there are twenty modified records in the Redo log, ten of which are incoming data, and the other ten are completed by Checkpoint. The data in the memory is written back to the hard disk and recorded to the Redo log. In order to verify that the data in the memory has been written back to the hard disk by Checkpoint. Use the Redo log which has not yet recorded the transferred data to overwrite the Redo log in MySQL. Power off the virtual machine to cause a Failover. Afterwards, the master will direct the database connection to SB to reconfirm the status of the data at user-end and the database-end as shown in Figure 9



(a) User-end (PostgreSQL) (b) Database-end



(a) User-end (MySQL) (b) Database-end

Figure. :9 (a),(b) Comparison of user-end and database-end data after a failover

From Figure 9 it can be observed that the data on the user-end and on the database-end is consistent. PostgreSQL compared with Case 2, there is 5 minutes more waiting time for Checkpoint to complete and

to record the time in the WAL. No matter is PostgreSQL or MySQL. The result data of this experiment is consistent.

Data still exists in memory and is not yet saved to the hard drive. Also, use the WAL before the data is entered to overwrite the WAL in PostgreSQL; use the Redo log before the data is entered to overwrite the Redo log in MySQL. Therefore, WAL and Redo log did not record the data entry. The data was lost in the memory and WAL and Redo log did not record the written data. The data could not be recovered via the WAL and Redo log. However, after the Failover, it was observed that the data still existed. This can prove that Checkpoint writes data from the memory into the hard disk, and SB can display the ten datasets that the user just entered after the failover.

5 Conclusions

This study proposes a high-availability database management system architecture. For the universality of this research, two databases, MySQL and PostgreSQL, are used to increase the chances of combining with various systems. In order to achieve the capabilities of data backup and system backup at the same time, there are three system operating mechanisms proposed. First, the architecture combines the WAL and Redo log mechanism of the database to implement a database disaster recovery mechanism. Next, this study can control the WAL and Redo log mechanism of the database, so that the correctness of the data is maintained even during a database failover. Finally, this study can analyze and control the timing when the database system writes cached data to the hard disk, and to verify with three experimental cases that the operation of the above three mentioned mechanisms are feasible. Based on the above three mechanisms, this study designs a failover process to avoid data loss in the cache memory due to a failover, so as to achieve a high availability database management system architecture.

REFERENCES

[1] DB-Engines. (2019). The first global database ranking list in the second half of the 2018, https://buzzorange.com/techorange/2018/08/02/2018-july-database-ranking/

[2] C. Jaiswal and V. Kumar, "DbHAaaS: Database High Availability as a Service," 2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Bangkok, 2015, pp. 725-732.doi: 10.1109/SITIS.2015.25

[3] R. H. de Souza, P. A. Flores, M. A. R. Dantas and F. Siqueira, "Architectural recovering model for Distributed Databases: A reliability, availability and serviceability approach," 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, 2016, pp. 575-580.doi: 10.1109/ISCC.2016.7543799

[4] Adnan, A. A. Ilham and S. Usman, "Performance analysis of extract, transform, load (ETL) in apache Hadoop atop NAS storage using ISCSI," 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), Kuta Bali, 2017, pp. 1-5.doi: 10.1109/CAIPT.2017.8320716

[5] A. Elghazi, M. Berrezzouq and Z. Abdelali, "New version of iSCSI protocol to secure Cloud data storage," 2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech), Marrakech, 2016, pp. 141-145. doi: 10.1109/CloudTech.2016.7847690

[6] M. Riasetiawan, A. Ashari and I. Endrayanto, "Distributed Replicated Block Device (DRDB) implementation on cluster storage data migration," 2015 International Conference on Data and Software Engineering (ICoDSE), Yogyakarta, 2015, pp. 93-97.doi: 10.1109/ICODSE.2015.7436978

[7] Rahul k.(2018). How to Setup IP Failover with KeepAlived on Ubuntu & Debian. Retrieved from https://tecadmin.net/setup-ip-failover-on-ubuntu-with-keepalive/

[8] Yen-Jen Chen & An-Liang Lo. Design and implementation of construction system with high-availability application service environment, IJCEE 2015 Vol.7(6): 357-369 ISSN: 1793-8163 DOI: 10.17706/IJCEE.2015.7.6.357-369

[9] R. F. Gibadullin, I. S. Vershinin and R. S. Minyazev, "Realization of replication mechanism in PostgreSQL DBMS," 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), St. Petersburg, 2017, pp. 1-6. doi: 10.1109/ICIEAM.2017.8076380

[10] S. Ryu, K. Lee and H. Han, "In-memory write-ahead logging for mobile smart devices with NVRAM," in IEEE Transactions on Consumer Electronics, vol. 61, no. 1, pp. 39-46, February 2015.doi: 10.1109/TCE.2015.7064109

[11] Peter Frühwirt , Peter Kieseberg, Sebastian Schrittwieser, Markus Huber, and Edgar Weippl, "InnoDB Database Forensics: Reconstructing Data Manipulation Queries from Redo Logs," in Seventh International Conference on Availability, Reliability and Security, 2012, doi:10.1109/ARES.2012.50

[12] Rohan McGovern.(2018). Home · rvoicilas/inotify-tools Wiki · GitHub. Retrieved from https://github.com/rvoicilas/inotify-tools/wiki

[13] S. i. Sou and Y. b. Lin, "Modeling mobility database failure restoration using checkpoint schemes," in IEEE Transactions on Wireless Communications, vol. 6, no. 1, pp. 313-319, Jan. 2007.doi: 10.1109/TWC.2007.05200