

## Development of Distributed DBMS based Replication Techniques

Jeong-Joon Kim

Department of Computer Engineering, Korea Polytechnic University,  
Gyeonggi-do, 15073 Siheung-si, South Korea

**Abstract:** The storage virtualization solution's MDS (MetaData Server) uses MySQL to store metadata and prepares for failures using MySQL replication. However, MySQL replication does not provide full synchronous mode, it provides only asynchronous mode and semi-synchronous mode. Therefore when a failure occurs in the active MDS node there is a possibility that data that is not replicated may be generated even if the service is moved to the standby MDS node. To solve this problem, change the redundancy scheme of MySQL using DRBD (Distributed Replicated Block Device) available on Linux.

**Key words:** Distributed DBMS, replication, heart beat, recovery, metadata, standby MDS node

### INTRODUCTION

The MetaData Server (MDS) of the storage virtualization solution uses MySQL to store metadata and is prepared for failure by using MySQL replication (Jang, 2009). However, MySQL replication does not provide full Synchronous mode, only asynchronous mode and semi-synchronous mode. Therefore when an active MDS node fails there is a possibility that unreplicated data will be generated even if the service is moved to the standby MDS node (Park *et al.*, 2016). To solve this problem we change the MySQL replication method using DRBD (Distributed Replicated Block Device) which is available in Linux (Xu *et al.*, 2009).

**Literature review:** Distributed Replicated Block Device (DRBD) an open source linux Kernel block device that provides several types of replication to obtain the same view of data between two systems. In newer Kernels 2.6.33 and above, the DRBD functionality is included in the Kernel and below that you need to install and configure the package. Each device configured in the DRBD cluster may be a primary or secondary DRBD. At both nodes, the DRBD Software creates a link between the DRBD virtual device and the local partition and configures communication information between both nodes. All read/write operations are performed at the primary node (Ellenberg, 2008; Reisner and Ellenberg, 2005).

Figure 1 provides an overview of DRBD in the context of two independent servers that provide independent storage resources. One of the two servers is usually defined as the primary server and the other server is defined as the secondary server. The user accesses the

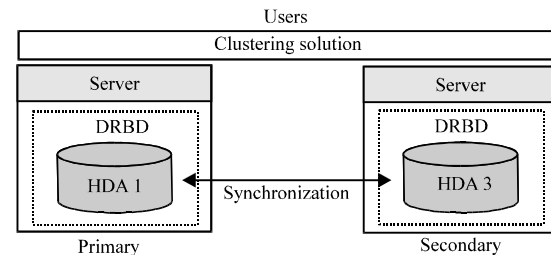


Fig. 1: The default DRBD model for operations

DRBD block device as a traditional local block device or accesses the storage area network or a network attached storage solution. DRBD Software provides synchronization between primary and secondary servers for user-based read and write operations and other synchronization operations.

In the active/passive model, the primary node is used for all user read and write operations. The secondary node is promoted to the primary node when it detects that the primary node is down in the clustering solution. Write operations occur over the primary node and are performed concurrently on local and secondary storage (Fig. 2). DRBD supports two modes of write operations: full synchronous mode and asynchronous mode.

In full synchronous mode, the write operation must be secure on both node's storage before the write transaction is acknowledged to the writer. In asynchronous mode, write transactions are acknowledged before write data is stored on the local node's storage. Data replication to the peer node occurs in the background. Asynchronous mode is the most secure mode for data protection because it is less secure but faster than full synchronous mode because there is a

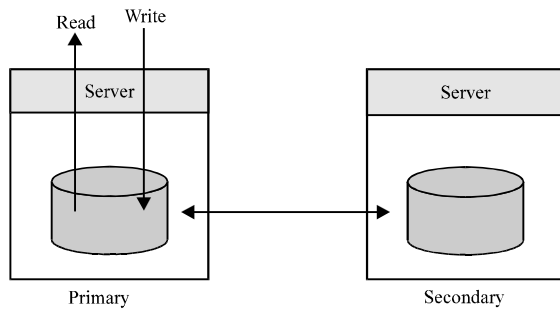


Fig. 2: Read/write operations in DRBD

window about the failure that occurs before data is replicated. Although, fully synchronous mode is recommended asynchronous mode is useful in situations where replication occurs over long distances such as wide area networks for local disaster recovery scenarios. The read operation is performed using local storage unless the local disk fails and accesses the secondary storage through the secondary if the local disk fails.

Because DRBD can also support active/active models, read and write operations can occur simultaneously on both servers in a mode called shared disk mode. The data written in the primary is transferred to the lower-level block device and copied to the secondary node through the network. That is in order to use DRBD, the same DRBD-dedicated partition is created on both nodes and the two partitions are linked to the DRBD virtual device, so that, they can be replicated through the network.

## MATERIALS AND METHODS

### System design

**Considerations and constraints:** In Kernel 2.6.33 or lower, you must register the Kernel module after installing DRBD. In this case, install DRBD together with Linux or Kernel installation. The MySQL account management is stored and managed in a volume or directory accessible only to administrators with specific privileges, so that, user information is not leaked as much as possible.

If the user information is leaked in the worst case, it is managed in an encrypted file format instead of a text file format, so that, the information cannot be confirmed.

If you use MySQL replication, data is not guaranteed but failover can happen instantaneously. However, if you use MySQL+DRBD, fail over time may take longer than MySQL replication because you have to start MySQL in standby in case of active node failure.

**Advantages and disadvantages of DRBD:** The advantage of DRBD is that it supports various modes. Since, it

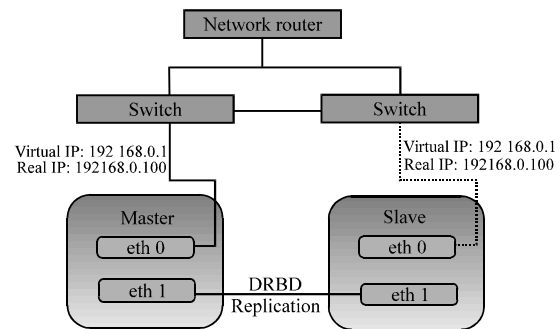


Fig. 3: Redundancy comparison test DRBD and MySQL

supports both full synchronous mode and asynchronous mode, it can be set flexibly according to user's needs. MySQL does not support full synchronous. Since, it has a replication architecture for each device block, any application can be configured to support high availability using DRBD. Inter-node communication can cause errors in the replicated data due to software or firmware bugs or other errors not detected in the checksum of TCP/IP. To provide data integrity, DRBD computes the message integrity code to be provided with data moving between nodes. This allows the receiving node to verify the validity of the received data and the retransmission of the request in the event of an error. DRBD is flexible about the integrity algorithms used because it uses the Linux cryptographic API (Application Programming Interface). It is an open source and complies with the GNU General Public License v2 and is available for free. However, the disadvantage of DRBD is that it only supports Linux OS.

### Advantages and disadvantages of existing method and DRBD application:

The advantage of applying DRBD is that it does not guarantee data because it replicates asynchronously when using MySQL replication but it can guarantee that all data will be replicated when it is replicated in synchronous mode using DRBD. The disadvantage of applying DRBD is that if you are using MySQL, you can have failover instantaneously because MySQL is running on both nodes. However, if you use MySQL+DRBD, failover time may take longer than MySQL replication because you have to start MySQL in standby in case of active node failure (Fig. 3).

### DRBD method and existing MySQL redundancy comparison test implementation plan:

- Identity problems with MySQL replication
- After performing MySQL replication on both nodes insert the data into active node

- Forcibly terminates MySQL on active node
- Terminate MySQL replication receiver on the standby node
- Start MySQL on the active node
- Verify that the data on active and standby nodes are different

#### **Confirmation of DRBD redundancy method:**

- In both nodes, DRBD is used to configure replication and then data is inserted into the active node
- Forcibly terminates MySQL on active node
- Terminate DRBD on the standby node
- Start MySQL on the active node
- Start MySQL on the standby node
- Verify that the data of active and standby nodes are identical

**Operation method:** The flow of the system proceeds in the order of service start, service stop, role change.

#### **Service start flow:**

- Start the service of master host
- mdsdm start
- Start the service of the slave host
- mdsdm start

#### **Service stop flow:**

- Service suspension of slave host
- mdsadm stop
- Service suspension master host
- mdsadm stop

#### **Change role:**

- Perform the following command on the master host
- mdsadm switch
- Role change flow
- Service suspension of master host
- Master switch of slave host
- Service is started from a stopped host to a slave

## **RESULTS AND DISCUSSION**

### **System implementation**

**Installing MySQL:** MySQL installs on the MDS node. MySQL can be downloaded from <http://www.mysql.com> and we recommend using MySQL 5.1.30 which is proven to be reliable. To install MySQL, the following packages must be installed in advance. If you do not have the additional packages mentioned in study, install the individual packages as yum or rpm and then install MySQL (Fig. 4-7):

- gcc
- gcc-c++
- termcap
- libtermcap
- libtermcap-devel

Unpack the downloaded MySQL-5.1.30.tar.gz into an arbitrary directory. Specify the -with-innodb option to use innodb as the storage engine for the configure utility (Algorithm 1).

#### **Algorithm 1; Installing and setting MySQL:**

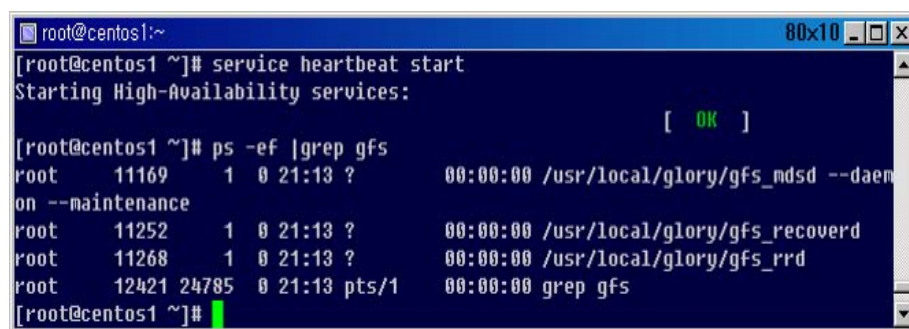
```
[root@mds/] tar xvf mysql-5.1.30.tar.gz
[root@mds/] cd mysql-5.1.30
[root@mds mysql-5.1.30] ./configure --prefix = /usr/local/mysql-with-innodb
[root@mds mysql-5.1.30] make
[root@mds mysql-5.1.30] make install
```

**Installing heartbeat:** Heartbeat is the software that manages the procedure of failover of the MDS service by judging the failure of the correspondent node. The heartbeat is installed in the MDS node and it is installed using yum in the following way (Algorithm 2).

#### **Algorithm 2; Downloading heartbeat by yum:**

```
[root@mds] yum install-y heartbeat
[root@mds] yum install-y heartbeat*
```

If you can not install heartbeat through yum, download the 2.1.4 Version from <http://linux-ha.org> and



```
root@centos1 ~# service heartbeat start
Starting High-Availability services:
[ OK ]

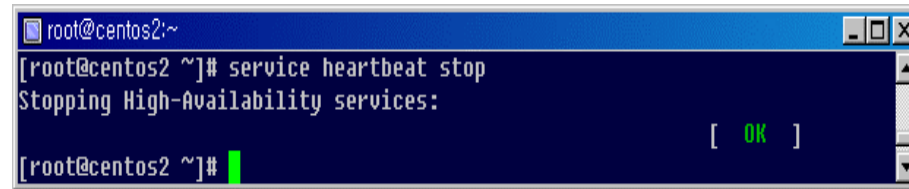
root@centos1 ~# ps -ef | grep gfs
root      11169      1  0 21:13 ?        00:00:00 /usr/local/glory/gfs_mdsc --daemon --maintenance
root      11252      1  0 21:13 ?        00:00:00 /usr/local/glory/gfs_recoverd
root      11268      1  0 21:13 ?        00:00:00 /usr/local/glory/gfs_rrd
root      12421 24785  0 21:13 pts/1    00:00:00 grep gfs
root@centos1 ~#
```

Fig. 4: Master MDS start



```
root@centos2:~  
[root@centos2 ~]# service heartbeat start  
Starting High-Availability services:  
[ OK ]  
[root@centos2 ~]#
```

Fig. 5: Slave MDS start



```
root@centos2:~  
[root@centos2 ~]# service heartbeat stop  
Stopping High-Availability services:  
[ OK ]  
[root@centos2 ~]#
```

Fig. 6: Master MDS stop



```
root@centos1:~  
[root@centos1 ~]# service heartbeat stop  
Stopping High-Availability services:  
[ OK ]  
[root@centos1 ~]# ps -ef |grep gfs  
root 32129 24785 0 21:22 pts/1 00:00:00 grep gfs  
[root@centos1 ~]#
```

Fig. 7: Slave MDS stop

proceed with the installation. The file looks like this. Heartbeat-stable-2-1-stable-2.1.4.tar.gz. Since, heartbeat needs to install libnet before installation, install libnet as follows (Algorithm 3).

**Algorithm 3; Installing libnet:**

```
[root@mids] tar xvfz libnet.tar.gz  
[root@mids] cd libnet  
[root@mids] ./configure; make; install
```

After libnet is installed, install heartbeat as follows (Algorithm 4). Confirmation of DRBD redundancy.

**Algorithm 4; Installing heartbeat:**

```
[root@mids] tar xvfz heartbeat-stable-2-1-stable-2.1.4.tar.gz  
[root@mids] tar xvfz heartbeat-stable-2-1-stable-2.1.4  
[root@mids] ./ConfigureMe configure-syscondir = /etc.  
[root@mids] make; make install
```

**MDS start and shutdown method:**

- Start master MDS
- Service heartbeat start
- Start slave MDS
- Service heartbeat start
- Master MDS Termination
- Service heartbeat stop

- Slave MDS termination
- Service heartbeat stop

**Coping with disability**

**Master OS failure:**

- Reboot-f
- Nf is an option to force shutdown (f) without disk sync (n) (do not call shutdown)
- Test results
- Slave MDS detects failure of master MDS and fails over and service continues after a while
- The OS of the failed node is restored and MDS is started automatically and participated as slave MDS

**Slave OS failure:**

- Reboot-f
- Test results
- Failure of slave MDS does not interfere with service and service is sustained
- The OS of the failed node is restored and MDS is started automatically and participated as slave MDS

**Master node failure:**

- Force resume after power down
- Test results

- Slave MDS detects failure of master MDS and fails over and service continues after a while
- MDS is started automatically as the faulty node is restored and participates as slave MDS

**Slave node failure:**

- Force resume after power down
- Test results
- Failure of the slave node does not interfere with the service and the service continues
- MDS is started automatically as the faulty node is restored and participates as slave MDS

**CONCLUSION**

The MetaData Server (MDS) of the storage virtualization solution used MySQL to store metadata and is prepared for failure using MySQL replication. However, MySQL replication does not provide full synchronous mode, only asynchronous mode and semi-synchronous mode. Therefore when an active MDS node fails there is a possibility that unreplicated data may be generated even if the service is moved to the standby MDS node.

To solve this problem we changed the MySQL replication method using DRBD (Distributed Replicated Block Device) which is available in Linux. In this study, the redundancy is configured as full synchronous to prevent data loss. In addition, in the event of a failure, MySQL automatically starts up on a redundant standby server so that the service can continue.

**ACKNOWLEDGEMENT**

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2017R1A2B4011243).

**REFERENCES**

- Ellenberg, L., 2008. Drbd 9 and device-mapper: Linux block level storage replication. Proceedings of the 15th International Conference on Linux System Technology, October 1-10, 2008, University of Hamburg, Hamburg, Germany, pp: 1-12.
- Jang, T.S., 2009. Understanding disk storage virtualization technology. Korea Inst. Inf. Technol. Mag., 7: 67-74.
- Park, J.Y., H. Park and C. Yoo, 2016. Design and implementation of host-side cache migration engine for high performance storage in a virtualization environment. KIISE. Trans. Comput. Pract., 22: 278-283.
- Reisner, P. and L. Ellenberg, 2005. Replicated storage with shared disk semantics. Proceedings of the 12th International Conference on Linux System Technology (Linux-Kongress), October, 11-14, 2005, University of Hamburg, Hamburg, Germany, pp: 111-119.
- Xu, H., L. Xing and R. Robidoux, 2009. Drbd: Dynamic reliability block diagrams for system reliability modelling. Intl. J. Comput. Appl., 31: 132-141.