Reduxio Tech Note

MySQL Replication

For more information, refer to Reduxio website at http://www.reduxio.com.
If you have comments about this documentation, submit your feedback to docs@reduxio.com.

Reduxio Systems, Inc.
111 Pine Avenue
South San Francisco, CA, 94080
United States
www.reduxio.com

Contents

Overview

Reduxio Tech Note - MySQL Replication describes various tests performed with MySQL Replication and Reduxio Storage.

Setup Details

Hardware Configuration

The following hardware components were used in this reference architecture:

No.	Hardware component	Description
1	Reduxio HX550 Storage System	Dual-controller Cache: 256GB RAM SSD: 8x 800GB MLC SSDs HDD: 16x 2TB 7.2k RPM SATA disks
1	Intel S1600JP Server	VMware vSphere 5.5 Server – used to run the DB VMs Intel® Xeon® CPU E5-1650 v2 @ 3.50GHz RAM: 64GB
1	Mellanox 10GbE switch	Interconnect between ESXi server and Reduxio HX550

Software Configuration

The following software components were used in this reference architecture:

No.	Software component	Description
1	VMware vSphere	ESXi 5.5.0 2302651
2	CentOS 6.6	CentOS release 6.6 (Final) Linux centos1 2.6.32-504.8.1.el6.x86_64 #1 SMP Wed Jan 28 21:11:36 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux iscsi-initiator-utils-6.2.0.873-13.el6.x86_64 device-mapper-multipath-libs-0.4.9-80.el6_6.3.x86_64 device-mapper-multipath-0.4.9-80.el6_6.3.x86_64
2	MySQL Server	VM Hardware: 4 cores, 16GB Percona XtraDB 5.6.23-72.1
1	Windows Server 2012 R2	Load generator running Dell Benchmark Factory V7.1.1

Setup

centos1 – MySQL 5.6 master server configured with row-based replication. Database stored on local disk of the VM.

centos2 – MySQL 5.6 slave server. Datadir stored on a 1TB Reduxio volume.

lg – Windows 2012 R2 running Dell Benchmark Factory – used to create the database and run a workload mix towards the master DB.

Database Configuration

Master – centos1

/etc/my.cnf:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
# eyal@reduxio.com, changed for Benchmark Factory
max_connections=1200
innodb_buffer_pool_size=4294967296
#
server-id=1
log-bin=mysql-bin
binlog_format=row
expire_logs_days=7
sync_binlog=1
innodb_flush_log_at_trx_commit=1
innodb_fast_shutdown=0
#
# new options for 5.6 crash-safe recovery
relay-log-recovery=1
relay-log-info-repository=TABLE
relay-log-purge=1
master-info-repository=TABLE

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Slave – centos2

/etc/my.cnf:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
# eyal@reduxio.com, changed for Benchmark Factory
max_connections=1200
innodb_buffer_pool_size=4294967296
server-id=2
report-host=hostname
binlog_format=row
#
# new options for 5.6 crash-safe recovery
relay-log-recovery=1
relay-log-info-repository=TABLE
relay-log-purge=1
master-info-repository=TABLE

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Tests

Database Recovery using Backdating

Test Description

To validate that Reduxio Backdating works reliably in a MySQL Replication environment, a replication is configured between two servers. Each of the servers is configured with its own Reduxio volume mounted as /var/lib/mysql. A database is generated and an IO workload mix is constantly running towards the master server.

While the slave is pulling changes from the master, 800 clones are created for each second back in time. Another script is then used to loop thru all of the backdated clones, mounting each clone, starting the database, starting the replication slave, and validating that the replication starts as expected.

Test Details

A database is created using Benchmark Factory with the following parameters:

Workload Mix	SELECT 5% INSERT 45% UPDATE 45% DELETE 5%
DB Size	400 GB

The loop_backdate.sh script uses Reduxio Backdating to create a clone for every second (in total ~16 minutes worth of seconds) prior to the current point in time:

```
...
...
==== Started test Wed Mar 18 11:16:44 IST 2015 / Iteration 192 / Creating clone for 03/18/2015-
11:13:33 ====
volumes clone mysqldb_slave1 -name mysqldb_slave1_13_33 -timestamp 03/18/2015-11:13:33
Clone created succesfully
==== Started test Wed Mar 18 11:16:44 IST 2015 / Iteration 193 / Creating clone for 03/18/2015-
11:13:32 ====
volumes clone mysqldb_slave1 -name mysqldb_slave1_13_32 -timestamp 03/18/2015-11:13:32
Clone created successfully
...
...
==== Started test Wed Mar 18 11:16:44 IST 2015 / Iteration 999 / Creating clone for 03/18/2015-11:0:6
====
volumes clone mysqldb_slave1 -name mysqldb_slave1_0_6 -timestamp 03/18/2015-11:0:6
Cannot execute command: Cannot create volume. System has reached the limit for the maximum number of
supported volumes (1000).
Max volumes reached. Exiting.
```

Up to 1,000 volumes (including clones) can be created in the system - ~15 minutes worth of past seconds. Less clones may be created if there are already existing volumes in the system.

Now, the clones are automatically validated using the loop_testclone.sh script:

```
[root@centos2 mysql_demo]# nohup ./loop_testclone.sh > loop_testclone.log 2>&1 &
```

This script mounts the cloned volume, starts MySQL, then starts the replication slave. The replication is validated using “show slave status” – validating that changes to centos1 are replicated to centos2. In addition, the database log is reviewed for any database or replication errors.

Test Results

1. In 100% of test runs (800 out of 800), the database has started successfully from the backdated volume.
   Conclusion: MySQL database can safely be recovered using BackDating - specific seconds back in time are recoverable
2. In 100% of test runs (800 out of 800), the replication slave was recovered using BackDating, and the replication slave was able to restart the replication from the master.
   Conclusion: The MySQL 5.6 slave can be recovered using BackDating

An example of a successful test run for one specific second - The database is stopped, volume cloned and mounted, database restarted, slave started and replication status is polled twice:

```
[root@centos2 mysql_demo]# ./loop_testclone.sh

==== Testing time 04/29/2015-:22:4:44, clone mysqldb_slave1_4_44 ====
+ echo '==== Testing time 04/29/2015-:22:4:44, clone mysqldb_slave1_4_44 ===='
```

First, the database is stopped, the volume is unmounted, and the host is deleted from the Reduxio configuration to avoid any leftover configurations from previous loop run:

```
+ service mysql stop
Shutting down MySQL (Percona Server)..                     [  OK  ]
+ umount /var/lib/mysql
+ iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p 10.46.239.11:3260 --logout
Logging out of session [sid: 1392, target: iqn.2013-12.com.reduxio:ff4032ff0041000e, portal:
10.46.239.11,3260]
Logout of [sid: 1392, target: iqn.2013-12.com.reduxio:ff4032ff0041000e, portal: 10.46.239.11,3260]
successful.
+ ssh rdxadmin@xrtx17-mgmt.il.reduxio 'hosts delete centos2 -force'
Host centos2 deleted
```

Then the host is recreated, and one of the 800 clones is assigned to the host:

```
+ ssh rdxadmin@xrtx17-mgmt.il.reduxio 'hosts add centos2 -iscsi-name iqn.1994-
05.com.redhat:42edf012c766'
Host created successfully
+ ssh rdxadmin@xrtx17-mgmt.il.reduxio 'volumes assign mysqldb_slave1_4_44 -host centos2 -lun 120'
Assignment created successfully
+ iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p 10.46.239.11:3260 --login
Logging in to [iface: default, target: iqn.2013-12.com.reduxio:ff4032ff0041000e, portal:
10.46.239.11,3260] (multiple)
Login to [iface: default, target: iqn.2013-12.com.reduxio:ff4032ff0041000e, portal: 10.46.239.11,3260]
successful.
+ sleep 2
+ lsscsi
[1:0:0:0]    cd/dvd  NECVMWar VMware IDE CDR10 1.00  /dev/sr0
[2:0:0:0]    disk    VMware   Virtual disk     1.0   /dev/sda
[2:0:1:0]    disk    VMware   Virtual disk     1.0   /dev/sdb
[3:0:0:120]  disk    REDUXIO  TCAS             2300  /dev/sdc
[5:0:0:120]  disk    REDUXIO  TCAS             2300  /dev/sde
[6:0:0:120]  disk    REDUXIO  TCAS             2300  /dev/sdf
[1395:0:0:120]disk    REDUXIO  TCAS             2300  /dev/sdd
```

The cloned volume is mounted, and the database is started again:

```
+ echo 'Mounting /dev/mapper/mpatha'
Mounting /dev/mapper/mpatha
+ echo 'Mounting /dev/mapper/mpatha'
+ mount /dev/mapper/mpatha /var/lib/mysql
+ echo 'Starting MySQL...'
Starting MySQL...
+ echo 'Starting MySQL...'
+ service mysql start
Starting MySQL (Percona Server)                            [  OK  ]
```

The script sleeps for 60 seconds to let the database fully start and reach a steady state:

```
+ echo 'sleep 60'
sleep 60
+ echo 'sleep 60'
+ sleep 60
```

The MySQL replication slave is started (note that it is set in /etc/my.cnf to not automatically start):

```
+ echo 'Starting slave...'
Starting slave...
+ echo 'Starting slave...'
+ mysql -e 'start slave;'
```

After an additional 60 seconds sleep, the slave status is validated twice within 60 seconds. A run time cycle in which the slave have shown Slave_IO_Running=Yes and Slave_SQL_Running=Yes is considered a test pass:

```
+ echo 'sleep 60'
sleep 60
+ echo 'sleep 60'
+ sleep 60
+ mysql -e 'show slave status\G'
+ grep Running
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
      Slave_SQL_Running_State: Reading event from the relay log
+ mysql -e 'show slave status\G'
+ grep Running
+ echo 'sleep 60'
sleep 60
+ echo 'sleep 60'
+ sleep 60
+ mysql -e 'show slave status\G'
+ grep Running
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
      Slave_SQL_Running_State: Reading event from the relay log
+ mysql -e 'show slave status\G'
+ grep Running
+ mysql -e 'stop slave;'
```

Scripts

loop_backdate.sh

Creates clones from the current time back in time, one clone for every second.

```
#!/bin/bash

SSH="ssh rdxadmin@xrtx17-mgmt.il.reduxio"
MAXVOLS=999
VOLS=0

DATE=`date`
CUR_MON=`date +%m`
CUR_DAY=`date +%d`
CUR_YEAR=`date +%Y`
CUR_HR=`date +%H`
CUR_MIN=`date +%M`
CUR_SEC=`date +%S`

BASE_TS="${CUR_MON}/${CUR_DAY}/${CUR_YEAR}-${CUR_HR}:${CUR_MIN}:${CUR_SEC}"

echo Current time = ${BASE_TS}

# loop from now back to the beginning of the current minute
for sec in `seq ${CUR_SEC} -1 0`;
do
        let "VOLS=${VOLS}+1"
        if [ "${VOLS}" -ge "1000" ]
        then
                echo "Max volumes reached. Exiting."
                exit
        fi
        TS="${CUR_MON}/${CUR_DAY}/${CUR_YEAR}-${CUR_HR}:${CUR_MIN}:${sec}"
        echo "==== Started test $DATE / Iteration ${VOLS} / Creating clone for ${TS} ===="
        echo "volumes clone mysqldb_slave1 -name mysqldb_slave1_${CUR_MIN}_${sec} -timestamp ${TS}"
        ${SSH} "volumes clone mysqldb_slave1 -name mysqldb_slave1_${CUR_MIN}_${sec} -timestamp ${TS}"
done

let "LAST_MIN=${CUR_MIN}-1"
# loop thru mins
for min in `seq ${LAST_MIN} -1 0`;
do
        # loop thru secs
        for sec in `seq 59 -1 0`;
        do
                let "VOLS=${VOLS}+1"
                if [ "${VOLS}" -ge "1000" ]
                then
                        echo "Max volumes reached. Exiting."
                        exit
                fi
                TS="${CUR_MON}/${CUR_DAY}/${CUR_YEAR}-${CUR_HR}:${min}:${sec}"
                echo "==== Started test $DATE / Iteration ${VOLS} / Creating clone for ${TS} ===="
                echo "volumes clone mysqldb_slave1 -name mysqldb_slave1_${min}_${sec} -timestamp
${TS}"
                ${SSH} "volumes clone mysqldb_slave1 -name mysqldb_slave1_${min}_${sec} -timestamp
${TS}"
        done
done
```

loop_testclone.sh

Mounts every clone, starts the database and replication slave.

```
#!/bin/bash -x

LOG=loop_testclone.log
SSH="ssh rdxadmin@xrtx17-mgmt.il.reduxio"
BASE_TS="04/29/2015-"
CUR_HR=22
CUR_MIN=18
CUR_SEC=22

let "LAST_MIN=${CUR_MIN}-1"
DEV=/dev/mapper/mpatha

service mysql stop
umount /var/lib/mysql
iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p 10.46.239.11:3260 --logout
${SSH} "hosts unassign centos2 -vol mysqldb_slave1"

echo Current time = ${CUR_HR}:${CUR_MIN}:${CUR_SEC}
echo Current time = ${CUR_HR}:${CUR_MIN}:${CUR_SEC} > $LOG

# loop from now back to the beginning of the current minute
for sec in `seq ${CUR_SEC} -1 0`;
do
        TS="${BASE_TS}:${CUR_HR}:${CUR_MIN}:${sec}"
        echo "==== Testing time ${TS}, clone mysqldb_slave1_${CUR_MIN}_${sec} ===="
        echo "==== Testing time ${TS}, clone mysqldb_slave1_${CUR_MIN}_${sec} ====" >> $LOG
        service mysql stop
        umount /var/lib/mysql
        iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p 10.46.239.11:3260 --
logout
        ${SSH} "hosts delete centos2 -force"
        ${SSH} "hosts add centos2 -iscsi-name iqn.1994-05.com.redhat:42edf012c766"
        ${SSH} "volumes assign mysqldb_slave1_${CUR_MIN}_${sec} -host centos2 -lun 120"
        iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p 10.46.239.11:3260 --
login
        sleep 2
        lsscsi
        echo "Mounting $DEV"
        echo "Mounting $DEV" >> $LOG
        mount $DEV /var/lib/mysql
        echo "Starting MySQL..."
        echo "Starting MySQL..." >> $LOG
        service mysql start
        echo "sleep 60"
        echo "sleep 60" >> $LOG
        sleep 60
        echo "Starting slave..."
        echo "Starting slave..." >> $LOG
        mysql -e 'start slave;'
        echo "sleep 60"
        echo "sleep 60" >> $LOG
        sleep 60
        mysql -e "show slave status\G"|grep Running
        mysql -e "show slave status\G"|grep Running >> $LOG
        echo "sleep 60"
        echo "sleep 60" >> $LOG
        sleep 60
        mysql -e "show slave status\G"|grep Running
        mysql -e "show slave status\G"|grep Running >> $LOG
        mysql -e 'stop slave;'
done

# loop thru mins
for min in `seq ${LAST_MIN} -1 0`;
do
        # loop thru secs
        for sec in `seq 59 -1 0`;
        do
                TS="${BASE_TS}:${CUR_HR}:${min}:${sec}"
                echo "==== Testing time ${TS}, clone mysqldb_slave1_${min}_${sec} ===="
                echo "==== Testing time ${TS}, clone mysqldb_slave1_${min}_${sec} ====" >> $LOG
                service mysql stop
                umount /var/lib/mysql
```

```
                iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p
10.46.239.11:3260 --logout
                ${SSH} "hosts delete centos2 -force"
                ${SSH} "hosts add centos2 -iscsi-name iqn.1994-05.com.redhat:42edf012c766"
                ${SSH} "volumes assign mysqldb_slave1_${min}_${sec} -host centos2 -lun 120"
                iscsiadm -m node --targetname iqn.2013-12.com.reduxio:ff4032ff0041000e -p
10.46.239.11:3260 --login
                sleep 2
                lsscsi
                echo "Mounting $DEV"
                echo "Mounting $DEV" >> $LOG
                mount $DEV /var/lib/mysql
                echo "Starting MySQL..."
                echo "Starting MySQL..." >> $LOG
                service mysql start
                echo "sleep 60"
                echo "sleep 60" >> $LOG
                sleep 60
                echo "Starting slave..."
                echo "Starting slave..." >> $LOG
                mysql -e 'start slave;'
                echo "sleep 60"
                echo "sleep 60" >> $LOG
                sleep 60
                mysql -e "show slave status\G"|grep Running
                mysql -e "show slave status\G"|grep Running >> $LOG
                echo "sleep 60"
                echo "sleep 60" >> $LOG
                sleep 60
                mysql -e "show slave status\G"|grep Running
                mysql -e "show slave status\G"|grep Running >> $LOG
                mysql -e 'stop slave;'
        done
done
```

Conclusion

The Reduxio storage solution provides a unique functionality called BackDating, which enables MySQL administrators to recover databases without prior planning or configuration, to any second in the past. BackDating can also be used in conuction with MySQL replication, simplifying the recovery of slave databases without a full copy from the master.

References

Reduxio Documentation

- *Reduxio TimeOS™ Administration Guide*

MySQL Documentation

- *MySQL Documentation portal*