## **前言**

我们试着想一想, 在生产环境中什么最重要？如果我们服务器的硬件坏了可以维修或者换新, 软件问题可以修复或重新安装, 但是如果数据没了呢？这可能是最恐怖的事情了吧, 我感觉在生产环境中应该没有什么比数据跟更为重要. 那么我们该如何保证数据不丢失、或者丢失后可以快速恢复呢？只要看完这篇, 大家应该就能对MySQL中实现数据备份和恢复能有一定的了解。

## **为什么需要备份数据？**

其实在前言中也大概说明了为什么要备份数据, 但是我们还是应该具体了解一下为什么要备份数据

在生产环境中我们数据库可能会遭遇各种各样的不测从而导致数据丢失, 大概分为以下几种.

硬件故障

软件故障

自然灾害

黑客攻击

误操作 (占比最大)

所以, 为了在数据丢失之后能够恢复数据, 我们就需要定期的备份数据, 备份数据的策略要根据不同的应用场景进行定制, 大致有几个参考数值, 我们可以根据这些数值从而定制符合特定环境中的数据备份策略

能够容忍丢失多少数据

恢复数据需要多长时间

需要恢复哪一些数据

## **数据的备份类型**

数据的备份类型根据其自身的特性主要分为以下几组

完全备份

部分备份

完全备份指的是**备份整个数据集( 即整个数据库 )**、部分备份指的是**备份部分数据集(例如: 只备份一个表)**

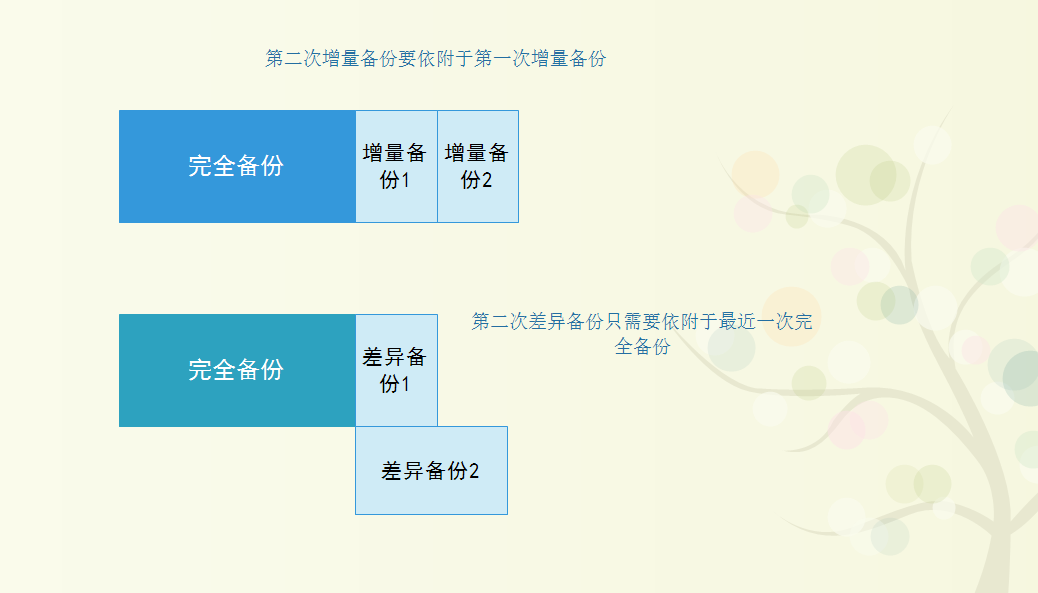
而部分备份又分为以下两种

增量备份

差异备份

增量备份指的是**备份自上一次备份以来(增量或完全)以来变化的数据**; 特点: 节约空间、还原麻烦   
差异备份指的是**备份自上一次完全备份以来变化的数据** 特点: 浪费空间、还原比增量备份简单

示意图



## ****MySQL备份数据的方式****

在MySQl中我们备份数据一般有几种方式

热备份

温备份

冷备份

热备份指的是当数据库进行备份时, **数据库的读写操作均不是受影响**  
温备份指的是当数据库进行备份时, **数据库的读操作可以执行, 但是不能执行写操作**   
冷备份指的是当数据库进行备份时, **数据库不能进行读写操作, 即数据库要下线**

MySQL中进行不同方式的备份还要考虑存储引擎是否支持

MyISAM

 热备 ×

 温备 √

 冷备 √

InnoDB

 热备 √

 温备 √

 冷备 √

我们在考虑完数据在备份时, 数据库的运行状态之后还需要考虑对于MySQL数据库中数据的备份方式

物理备份一般就是**通过**tar**,**cp**等命令直接打包复制数据库的数据文件**达到备份的效果   
逻辑备份一般就是**通过特定工具从数据库中导出数据并另存备份**(逻辑备份会丢失数据精度)

物理备份

逻辑备份

## **备份需要考虑的问题**

定制备份策略前, 我们还需要考虑一些问题

**我们要备份什么?**

一般情况下, 我们需要备份的数据分为以下几种

数据

二进制日志, InnoDB事务日志

代码(存储过程、存储函数、触发器、事件调度器)

服务器配置文件

**备份工具**

这里我们列举出常用的几种备份工具   
mysqldump : 逻辑备份工具, 适用于所有的存储引擎, 支持温备、完全备份、部分备份、对于**InnoDB**存储引擎支持热备   
cp, tar 等归档复制工具: 物理备份工具, 适用于所有的存储引擎, 冷备、完全备份、部分备份   
lvm2 snapshot: 几乎热备, 借助文件系统管理工具进行备份   
mysqlhotcopy: 名不副实的的一个工具, 几乎冷备, 仅支持**MyISAM**存储引擎   
xtrabackup: 一款非常强大的**InnoDB/XtraDB**热备工具, 支持完全备份、增量备份, 由percona提供

## **设计合适的备份策略**

针对不同的场景下, 我们应该制定不同的备份策略对数据库进行备份, 一般情况下, 备份策略一般为以下三种

**直接cp,tar复制数据库文件**

**mysqldump+复制BIN LOGS**

**lvm2快照+复制BIN LOGS**

**xtrabackup**

以上的几种解决方案分别针对于不同的场景

如果数据量较小, 可以使用第一种方式, 直接复制数据库文件

如果数据量还行, 可以使用第二种方式, 先使用mysqldump对数据库进行完全备份, 然后定期备份BINARY LOG达到增量备份的效果

如果数据量一般, 而又不过分影响业务运行, 可以使用第三种方式, 使用lvm2的快照对数据文件进行备份, 而后定期备份BINARY LOG达到增量备份的效果

如果数据量很大, 而又不过分影响业务运行, 可以使用第四种方式, 使用xtrabackup进行完全备份后, 定期使用xtrabackup进行增量备份或差异备份

## **实战演练**

### **使用cp进行备份**

我们这里使用的是使用yum安装的mysql-5.1的版本, 使用的数据集为从网络上找到的一个员工数据库

**查看数据库的信息**

mysql> SHOW DATABASES;    #查看当前的数据库, 我们的数据库为employees  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
4 rows in set (0.00 sec)  
  
mysql> USE employees;  
Database changed  
mysql> SHOW TABLES;         #查看当前库中的表  
+---------------------+  
| Tables\_in\_employees |  
+---------------------+  
| departments         |  
| dept\_emp            |  
| dept\_manager        |  
| employees           |  
| salaries            |  
| titles              |  
+---------------------+  
6 rows in set (0.00 sec)  
  
mysql> SELECT COUNT(\*) FROM employees;   #由于篇幅原因, 我们这里只看一下employees的行数为300024  
+----------+  
| COUNT(\*) |  
+----------+  
|   300024 |  
+----------+  
1 row in set (0.05 sec)

**向数据库施加读锁**

mysql> FLUSH TABLES WITH READ LOCK;    #向所有表施加读锁  
Query OK, 0 rows affected (0.00 sec)

**备份数据文件**

[root@node1 ~]# mkdir /backup   #创建文件夹存放备份数据库文件  
[root@node1 ~]# cp -a /var/lib/mysql/\* /backup     #保留权限的拷贝源数据文件  
[root@node1 ~]# ls /backup   #查看目录下的文件  
employees  ibdata1  ib\_logfile0  ib\_logfile1  mysql  mysql.sock  test

**模拟数据丢失并恢复**

[root@node1 ~]# rm -rf /var/lib/mysql/\*    #删除数据库的所有文件  
[root@node1 ~]# service mysqld restart   #重启MySQL, 如果是编译安装的应该不能启动, 如果rpm安装则会重新初始化数据库  
  
  
mysql> SHOW DATABASES;    #因为我们是rpm安装的, 连接到MySQL进行查看, 发现数据丢失了！  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| mysql              |  
| test               |  
+--------------------+  
3 rows in set (0.00 sec)  
  
[root@node1 ~]# rm -rf /var/lib/mysql/\*    #这一步可以不做  
[root@node1 ~]# cp -a /backup/\* /var/lib/mysql/    #将备份的数据文件拷贝回去  
[root@node1 ~]# service mysqld restart  #重启MySQL  
  
  
#重新连接数据并查看  
  
mysql> SHOW DATABASES;    #数据库已恢复  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
4 rows in set (0.00 sec)  
  
mysql> USE employees;        
  
mysql> SELECT COUNT(\*) FROM employees;    #表的行数没有变化  
+----------+  
| COUNT(\*) |  
+----------+  
|   300024 |  
+----------+  
1 row in set (0.06 sec)  
  
  
##完成

### **使用mysqldump+复制BINARY LOG备份**

我们这里使用的是使用yum安装的mysql-5.1的版本, 使用的数据集为从网络上找到的一个员工数据库

我们通过mysqldump进行一次完全备份, 再修改表中的数据, 然后再通过binary log进行恢复 **二进制日志需要在mysql配置文件中添加 log\_bin=on 开启**

mysqldump**命令介绍**

mysqldump是一个客户端的逻辑备份工具, 可以生成一个重现创建原始数据库和表的SQL语句, 可以支持所有的存储引擎, 对于InnoDB支持热备

[官方文档介绍](http://dev.mysql.com/doc/refman/5.7/en/mysqldump.html" \t "/root/文档\\x/_blank)

#基本语法格式  
  
shell> mysqldump [options] db\_name [tbl\_name ...]    恢复需要手动CRATE DATABASES  
shell> mysqldump [options] --databases db\_name ...   恢复不需要手动创建数据库  
shell> mysqldump [options] --all-databases           恢复不需要手动创建数据库  
  
  
其他选项:  
     -E, --events: 备份事件调度器  
     -R, --routines: 备份存储过程和存储函数  
     --triggers: 备份表的触发器; --skip-triggers   
     --master-date[=value]    
         1: 记录为CHANGE MASTER TO 语句、语句不被注释  
         2: 记录为注释的CHANGE MASTER TO语句  
         基于二进制还原只能全库还原  
  
     --flush-logs: 日志滚动  
         锁定表完成后执行日志滚动

**查看数据库的信息**

mysql> SHOW DATABASES;    #查看当前的数据库, 我们的数据库为employees  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
4 rows in set (0.00 sec)  
  
mysql> USE employees;  
Database changed  
mysql> SHOW TABLES;         #查看当前库中的表  
+---------------------+  
| Tables\_in\_employees |  
+---------------------+  
| departments         |  
| dept\_emp            |  
| dept\_manager        |  
| employees           |  
| salaries            |  
| titles              |  
+---------------------+  
6 rows in set (0.00 sec)  
  
mysql> SELECT COUNT(\*) FROM employees;   #由于篇幅原因, 我们这里只看一下employees的行数为300024  
+----------+  
| COUNT(\*) |  
+----------+  
|   300024 |  
+----------+  
1 row in set (0.05 sec)

**使用**mysqldump**备份数据库**

[root@node1 ~]# mysql -uroot -p -e 'SHOW MASTER STATUS'   #查看当前二进制文件的状态, 并记录下position的数字  
+------------------+----------+--------------+------------------+  
| File             | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |  
+------------------+----------+--------------+------------------+  
| mysql-bin.000003 |      106 |              |                  |  
+------------------+----------+--------------+------------------+  
  
[root@node1 ~]# mysqldump --all-databases --lock-all-tables  > backup.sql   #备份数据库到backup.sql文件中  
  
mysql> CREATE DATABASE TEST1;   #创建一个数据库  
Query OK, 1 row affected (0.00 sec)  
  
mysql> SHOW MASTER STATUS;   #记下现在的position  
+------------------+----------+--------------+------------------+  
| File             | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |  
+------------------+----------+--------------+------------------+  
| mysql-bin.000003 |      191 |              |                  |  
+------------------+----------+--------------+------------------+  
1 row in set (0.00 sec)  
  
[root@node1 ~]# cp /var/lib/mysql/mysql-bin.000003 /root  #备份二进制文件  
[root@node1 ~]# service mysqld stop   #停止MySQL  
[root@node1 ~]# rm -rf /var/lib/mysql/\*   #删除所有的数据文件  
[root@node1 ~]# service mysqld start    #启动MySQL, 如果是编译安装的应该不能启动(需重新初始化), 如果rpm安装则会重新初始化数据库  
  
  
mysql> SHOW DATABASES;   #查看数据库, 数据丢失!  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| mysql              |  
| test               |  
+--------------------+  
3 rows in set (0.00 sec)  
  
mysql> SET sql\_log\_bin=OFF;   #暂时先将二进制日志关闭    
Query OK, 0 rows affected (0.00 sec)  
  
  
mysql> source backup.sql  #恢复数据，所需时间根据数据库时间大小而定  
  
mysql> SET sql\_log\_bin=ON; 开启二进制日志  
  
mysql> SHOW DATABASES;   #数据库恢复, 但是缺少TEST1  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
4 rows in set (0.00 sec)  
  
[root@node1 ~]# mysqlbinlog --start-position=106 --stop-position=191 mysql-bin.000003 | mysql employees #通过二进制日志增量恢复数据  
  
mysql> SHOW DATABASES;    #现在TEST1出现了！  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| TEST1              |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
5 rows in set (0.00 sec)  
  
  
  
#完成

### **使用lvm2快照备份数据**

做实验之前我们先回顾一下lvm2-snapshot的知识

LVM快照简单来说就是将所快照源分区一个时间点所有文件的元数据进行保存，如果源文件没有改变，那么访问快照卷的相应文件则直接指向源分区的源文件，如果源文件发生改变，则快照卷中与之对应的文件不会发生改变。快照卷主要用于辅助备份文件。 这里只简单介绍，[点击查看详细介绍](http://www.360doc.com/content/13/0522/16/11801283_287305129.shtml" \t "/root/文档\\x/_blank)

**部署lvm环境**

添加硬盘; 这里我们直接实现SCSI硬盘的热插拔, 首先在虚拟机中添加一块硬盘, 不重启  
  
[root@node1 ~]# ls /dev/sd\*   #只有以下几块硬盘, 但是我们不重启可以让系统识别新添加的硬盘  
/dev/sda  /dev/sda1  /dev/sda2  
  
[root@node1 ~]# echo '- - -' > /sys/class/scsi\_host/host0/scan   
[root@node1 ~]# echo '- - -' > /sys/class/scsi\_host/host1/scan   
[root@node1 ~]# echo '- - -' > /sys/class/scsi\_host/host2/scan   
  
[root@node1 ~]# ls /dev/sd\*    #看！sdb识别出来了  
/dev/sda  /dev/sda1  /dev/sda2  /dev/sdb  
  
  
[root@node1 ~]# fdisk /dev/sdb   #分区  
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel  
Building a new DOS disklabel with disk identifier 0xd353d192.  
Changes will remain in memory only, until you decide to write them.  
After that, of course, the previous content won't be recoverable.  
  
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)  
  
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to  
         switch off the mode (command 'c') and change display units to  
         sectors (command 'u').  
  
Command (m for help): n  
Command action  
   e   extended  
   p   primary partition (1-4)  
p  
Partition number (1-4): 1  
First cylinder (1-2610, default 1):   
Using default value 1  
Last cylinder, +cylinders or +size{K,M,G} (1-2610, default 2610): +15G  
  
Command (m for help): t  
Selected partition 1  
Hex code (type L to list codes): 8e  
Changed system type of partition 1 to 8e (Linux LVM)  
  
Command (m for help): w  
The partition table has been altered!  
  
Calling ioctl() to re-read partition table.  
Syncing disks.  
You have new mail in /var/spool/mail/root  
[root@node1 ~]# partx -a /dev/sdb  
BLKPG: Device or resource busy  
error adding partition 1  
  
##创建逻辑卷  
[root@node1 ~]# pvcreate /dev/sdb1  
  Physical volume "/dev/sdb1" successfully created  
[root@node1 ~]# vgcreate myvg /dev/sdb1   
  Volume group "myvg" successfully created  
[root@node1 ~]# lvcreate -n mydata -L 5G myvg   
  Logical volume "mydata" created.  
  
[root@node1 ~]# mkfs.ext4 /dev/mapper/myvg-mydata   #格式化  
[root@node1 ~]# mkdir /lvm\_data  
[root@node1 ~]# mount /dev/mapper/myvg-mydata /lvm\_data  #挂载到/lvm\_data  
  
  
[root@node1 ~]# vim /etc/my.cnf    #修改mysql配置文件的datadir如下  
  
datadir=/lvm\_data  
  
[root@node1 ~]# service mysqld restart  #重启MySQL  
  
####重新导入employees数据库########略过####

**查看数据库的信息**

mysql> SHOW DATABASES;    #查看当前的数据库, 我们的数据库为employees  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
4 rows in set (0.00 sec)  
  
mysql> USE employees;  
Database changed  
mysql> SHOW TABLES;         #查看当前库中的表  
+---------------------+  
| Tables\_in\_employees |  
+---------------------+  
| departments         |  
| dept\_emp            |  
| dept\_manager        |  
| employees           |  
| salaries            |  
| titles              |  
+---------------------+  
6 rows in set (0.00 sec)  
  
mysql> SELECT COUNT(\*) FROM employees;   #由于篇幅原因, 我们这里只看一下employees的行数为300024  
+----------+  
| COUNT(\*) |  
+----------+  
|   300024 |  
+----------+  
1 row in set (0.05 sec)

**创建快照卷并备份**

mysql> FLUSH TABLES WITH READ LOCK;     #锁定所有表  
Query OK, 0 rows affected (0.00 sec)  
  
[root@node1 lvm\_data]# lvcreate -L 1G -n mydata-snap -p r -s /dev/mapper/myvg-mydata   #创建快照卷  
  Logical volume "mydata-snap" created.  
  
mysql> UNLOCK TABLES;  #解锁所有表  
Query OK, 0 rows affected (0.00 sec)  
  
[root@node1 lvm\_data]# mkdir /lvm\_snap  #创建文件夹  
[root@node1 lvm\_data]# mount /dev/myvg/mydata-snap /lvm\_snap/  #挂载snap  
mount: block device /dev/mapper/myvg-mydata--snap is write-protected, mounting read-only  
  
[root@node1 lvm\_data]# cd /lvm\_snap/  
[root@node1 lvm\_snap]# ls  
employees  ibdata1  ib\_logfile0  ib\_logfile1  mysql  mysql-bin.000001  mysql-bin.000002  mysql-bin.000003  mysql-bin.index  test  
[root@node1 lvm\_snap]# tar cf /tmp/mysqlback.tar \*  #打包文件到/tmp/mysqlback.tar  
  
[root@node1 ~]# umount /lvm\_snap/  #卸载snap  
[root@node1 ~]# lvremove myvg mydata-snap  #删除snap

**恢复数据**

[root@node1 lvm\_snap]# rm -rf /lvm\_data/\*  
[root@node1 ~]# service mysqld start    #启动MySQL, 如果是编译安装的应该不能启动(需重新初始化), 如果rpm安装则会重新初始化数据库  
  
  
mysql> SHOW DATABASES;   #查看数据库, 数据丢失!  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| mysql              |  
| test               |  
+--------------------+  
3 rows in set (0.00 sec)  
  
[root@node1 ~]# cd /lvm\_data/  
[root@node1 lvm\_data]# rm -rf \* #删除所有文件  
[root@node1 lvm\_data]# tar xf /tmp/mysqlback.tar     #解压备份数据库到此文件夹   
[root@node1 lvm\_data]# ls  #查看当前的文件  
employees  ibdata1  ib\_logfile0  ib\_logfile1  mysql  mysql-bin.000001  mysql-bin.000002  mysql-bin.000003  mysql-bin.index  test  
  
mysql> SHOW DATABASES;  #数据恢复了  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| employees          |  
| mysql              |  
| test               |  
+--------------------+  
4 rows in set (0.00 sec)  
  
  
##完成

### **使用Xtrabackup备份**

为了更好地演示, 我们这次使用mariadb-5.5的版本, 使用xtrabackup使用InnoDB能够发挥其最大功效, 并且InnoDB的每一张表必须使用单独的表空间, 我们需要在配置文件中添加 innodb\_file\_per\_table = ON 来开启

**下载安装**xtrabackup

我们这里通过wget percona官方的rpm包进行安装  
[root@node1 ~]# wget https://www.percona.com/downloads/XtraBackup/Percona-XtraBackup-2.3.4/binary/redhat/6/x86\_64/percona-xtrabackup-2.3.4-1.el6.x86\_64.rpm     
[root@node1 ~]# yum localinstall percona-xtrabackup-2.3.4-1.el6.x86\_64.rpm   #需要EPEL源

xtrabackup**介绍**

Xtrabackup是由percona提供的mysql数据库备份工具，据官方介绍，这也是世界上惟一一款开源的能够对innodb和xtradb数据库进行热备的工具。特点：

备份过程快速、可靠；lvm2 snapshot

备份过程不会打断正在执行的事务；

能够基于压缩等功能节约磁盘空间和流量；

自动实现备份检验；

还原速度快；

**摘自马哥的文档**

xtrabackup**实现完全备份**

我们这里使用xtrabackup的前端配置工具innobackupex来实现对数据库的完全备份

使用innobackupex备份时, 会调用xtrabackup备份所有的**InnoDB**表, 复制所有关于表结构定义的相关文件(.frm)、以及**MyISAM**、**MERGE**、**CSV**和**ARCHIVE**表的相关文件, 同时还会备份触发器和数据库配置文件信息相关的文件, 这些文件会被保存至一个以时间命名的目录.

**备份过程**

[root@node1 ~]# mkdir /extrabackup  #创建备份目录  
[root@node1 ~]# innobackupex --user=root /extrabackup/ #备份数据  
###################提示complete表示成功\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
[root@node1 ~]# ls /extrabackup/  #看到备份目录  
2016-04-27\_07-30-48

一般情况, 备份完成后, 数据不能用于恢复操作, 因为备份的数据中可能会包含尚未提交的事务或已经提交但尚未同步至数据文件中的事务。因此, 此时的数据文件仍不一致, 所以我们需要”准备”一个完全备份

[root@node1 ~]# innobackupex --apply-log /extrabackup/2016-04-27\_07-30-48/  #指定备份文件的目录  
  
#一般情况下下面三行结尾代表成功\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
InnoDB: Starting shutdown...  
InnoDB: Shutdown completed; log sequence number 369661462  
160427 07:40:11 completed OK!  
  
[root@node1 ~]# cd /extrabackup/2016-04-27\_07-30-48/  
[root@node1 2016-04-27\_07-30-48]# ls -hl  #查看备份文件lvm2 snapshot  
total 31M  
-rw-r----- 1 root root  386 Apr 27 07:30 backup-my.cnf  
drwx------ 2 root root 4.0K Apr 27 07:30 employees  
-rw-r----- 1 root root  18M Apr 27 07:40 ibdata1  
-rw-r--r-- 1 root root 5.0M Apr 27 07:40 ib\_logfile0  
-rw-r--r-- 1 root root 5.0M Apr 27 07:40 ib\_logfile1  
drwx------ 2 root root 4.0K Apr 27 07:30 mysql  
drwx------ 2 root root 4.0K Apr 27 07:30 performance\_schema  
drwx------ 2 root root 4.0K Apr 27 07:30 test  
-rw-r----- 1 root root   27 Apr 27 07:30 xtrabackup\_binlog\_info  
-rw-r--r-- 1 root root   29 Apr 27 07:40 xtrabackup\_binlog\_pos\_innodb  
-rw-r----- 1 root root  117 Apr 27 07:40 xtrabackup\_checkpoints  
-rw-r----- 1 root root  470 Apr 27 07:30 xtrabackup\_info  
-rw-r----- 1 root root 2.0M Apr 27 07:40 xtrabackup\_logfile

**恢复数据**

[root@node1 ~]# rm -rf /data/\*   #删除数据文件  
  
\*\*\*不用启动数据库也可以还原\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
[root@node1 ~]# innobackupex --copy-back /extrabackup/2016-04-27\_07-30-48/   #恢复数据, 记清使用方法  
  
#########我们这里是编译安装的mariadb所以需要做一些操作##########  
[root@node1 data]# killall mysqld  
  
[root@node1 ~]# chown -R mysql:mysql ./\*   
[root@node1 ~]# ll /data/      #数据恢复  
total 28704  
-rw-rw---- 1 mysql mysql    16384 Apr 27 07:43 aria\_log.00000001  
-rw-rw---- 1 mysql mysql       52 Apr 27 07:43 aria\_log\_control  
-rw-rw---- 1 mysql mysql 18874368 Apr 27 07:43 ibdata1  
-rw-rw---- 1 mysql mysql  5242880 Apr 27 07:43 ib\_logfile0  
-rw-rw---- 1 mysql mysql  5242880 Apr 27 07:43 ib\_logfile1  
-rw-rw---- 1 mysql mysql      264 Apr 27 07:43 mysql-bin.000001  
-rw-rw---- 1 mysql mysql       19 Apr 27 07:43 mysql-bin.index  
-rw-r----- 1 mysql mysql     2166 Apr 27 07:43 node1.anyisalin.com.err  
  
  
[root@node1 data]# service mysqld restart  
MySQL server PID file could not be found!                  [FAILED]  
Starting MySQL..                                           [  OK  ]  
  
MariaDB [(none)]> SHOW DATABASES;  #查看数据库, 已经恢复  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |lvm2 snapshot  
| employees          |  
| mysql              |  
| performance\_schema |  
| test               |  
+--------------------+  
5 rows in set (0.00 sec

**增量备份**

#########创建连两个数据库以供测试#####################  
MariaDB [(none)]> CREATE DATABASE TEST1;  
Query OK, 1 row affected (0.00 sec)  
  
MariaDB [(none)]> CREATE DATABASE TEST2;  
Query OK, 1 row affected (0.00 sec)  
  
[root@node1 ~]# innobackupex --incremental /extrabackup/ --incremental-basedir=/extrabackup/2016-04-27\_07-30-48/   
  
[root@node1 ~]# ls /extrabackup/2016-04-27\_07-57-22/ #查看备份文件  
total 96  
-rw-r----- 1 root root   386 Apr 27 07:57 backup-my.cnf  
drwx------ 2 root root  4096 Apr 27 07:57 employees  
-rw-r----- 1 root root 49152 Apr 27 07:57 ibdata1.delta  
-rw-r----- 1 root root    44 Apr 27 07:57 ibdata1.meta  
drwx------ 2 root root  4096 Apr 27 07:57 mysql  
drwx------ 2 root root  4096 Apr 27 07:57 performance\_schema  
drwx------ 2 root root  4096 Apr 27 07:57 test  
drwx------ 2 root root  4096 Apr 27 07:57 TEST1  
drwx------ 2 root root  4096 Apr 27 07:57 TEST2  
-rw-r----- 1 root root    21 Apr 27 07:57 xtrabackup\_binlog\_info  
-rw-r----- 1 root root   123 Apr 27 07:57 xtrabackup\_checkpoints  
-rw-r----- 1 root root   530 Apr 27 07:57 xtrabackup\_info  
-rw-r----- 1 root root  2560 Apr 27 07:57 xtrabackup\_logfile

BASEDIR指的是完全备份所在的目录，此命令执行结束后，innobackupex命令会在/extrabackup目录中创建一个新的以时间命名的目录以存放所有的增量备份数据。另外，在执行过增量备份之后再一次进行增量备份时，其--incremental-basedir应该指向上一次的增量备份所在的目录。

需要注意的是，增量备份仅能应用于InnoDB或XtraDB表，对于MyISAM表而言，执行增量备份时其实进行的是完全备份。

**整理增量备份**

[root@node1 ~]# innobackupex --apply-log --redo-only /extrabackup/2016-04-27\_07-30-48/  
[root@node1 ~]# innobackupex --apply-log --redo-only /extrabackup/2016-04-27\_07-30-48/ --incremental-dir=/extrabackup/2016-04-27\_07-5  
7-22/

**恢复数据**

[root@node1 ~]# rm -rf /data/\*   #删除数据  
  
[root@node1 ~]# innobackupex --copy-back /extrabackup/2016-04-27\_07-30-48/     #整理增量备份之后可以直接通过全量备份还原  
  
[root@node1 ~]# chown -R mysql.mysql /data/  
[root@node1 ~]# ls /data/ -l  
total 28732  
-rw-rw---- 1 mysql mysql     8192 Apr 27 08:05 aria\_log.00000001  
-rw-rw---- 1 mysql mysql       52 Apr 27 08:05 aria\_log\_control  
drwx------ 2 mysql mysql     4096 Apr 27 08:05 employees  
-rw-r----- 1 mysql mysql 18874368 Apr 27 08:05 ibdata1  
-rw-r----- 1 mysql mysql  5242880 Apr 27 08:05 ib\_logfile0  
-rw-r----- 1 mysql mysql  5242880 Apr 27 08:05 ib\_logfile1  
drwx------ 2 mysql mysql     4096 Apr 27 08:05 mysql  
-rw-rw---- 1 mysql mysql      245 Apr 27 08:05 mysql-bin.000001  
-rw-rw---- 1 mysql mysql       19 Apr 27 08:05 mysql-bin.index  
-rw-r----- 1 mysql mysql     1812 Apr 27 08:05 node1.anyisalin.com.err  
-rw-rw---- 1 mysql mysql        5 Apr 27 08:05 node1.anyisalin.com.pid  
drwx------ 2 mysql mysql     4096 Apr 27 08:05 performance\_schema  
drwx------ 2 mysql mysql     4096 Apr 27 08:05 test  
drwx------ 2 mysql mysql     4096 Apr 27 08:05 TEST1  
drwx------ 2 mysql mysql     4096 Apr 27 08:05 TEST2  
-rw-r----- 1 mysql mysql       29 Apr 27 08:05 xtrabackup\_binlog\_pos\_innodb  
-rw-r----- 1 mysql mysql      530 Apr 27 08:05 xtrabackup\_info  
  
MariaDB [(none)]> SHOW DATABASES;  #数据还原  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| TEST1              |  
| TEST2              |  
| employees          |  
| mysql              |  
| performance\_schema |  
| test               |  
+--------------------+  
7 rows in set (0.00 sec)  
  
#关于xtrabackup还有很多强大的功能没有叙述、有兴趣可以去看官方文档

## **总结**

| **备份方法** | **备份速度** | **恢复速度** | **便捷性** | **功能** | **一般用于** |
| --- | --- | --- | --- | --- | --- |
| cp | 快 | 快 | 一般、灵活性低 | 很弱 | 少量数据备份 |
| mysqldump | 慢 | 慢 | 一般、可无视存储引擎的差异 | 一般 | 中小型数据量的备份 |
| lvm2快照 | 快 | 快 | 一般、支持几乎热备、速度快 | 一般 | 中小型数据量的备份 |
| xtrabackup | 较快 | 较快 | 实现innodb热备、对存储引擎有要求 | 强大 | 较大规模的备份 |

其实我们还可以通过Master-Slave Replication 进行备份。

转自：http://www.cnblogs.com/liangshaoye/p/5464794.html