


MYSQL 备份恢复与XTRABACKUP备份

惨绿少年 Linux运维, MySQL, 数据库 0评论 来源: 本站原创 50°C 字体: 小 中 大

 新年贺词



2017年即将过去, 新年的钟声即将敲响。在这辞旧迎新的美好时刻, 我向全国各族人民, 向香港特别行政区同胞、澳门特别行政区同胞, 向台湾同胞和海外侨胞, 向工作在一线的运维工程师们, 向为开源事业做出贡献的朋友们, 向世界各国各地区的朋友们, 致以新年的祝福!

今天是2017的最后一天, 在这样一个特殊的日子, 希望大家都能事事顺心, 快乐常在, 希望在2018年里都能有所成就, 创造不一样的价值。

让我们满怀信心和期待, 一起迎接新年的钟声!

谢谢大家。

1.1 备份的原因

备份是数据安全的最后一道防线, 对于任何数据丢失的场景, 备份虽然不一定能恢复百分之百的数据(取决于备份周期), 但至少能将损失降到最低。衡量备份恢复有两个重要的指标: 恢复点目标(RPO)和恢复时间目标(RTO), 前者重点关注能恢复到什么程度, 而后者则重点关注恢复需要多长时间。

1.1.1 备份的目录

做灾难恢复: 对损坏的数据进行恢复和还原

需求改变: 因需求改变而需要把数据还原到改变以前

测试: 测试新功能是否可用

1.1.2 备份中需要考虑的问题

可以容忍丢失多长时间的数据;

恢复数据要在多长时间内完;

恢复的时候是否需要持续提供服务;

恢复的对象, 是整个库, 多个表, 还是单个库, 单个表。

1.1.3 备份的类型

热备份:

这些动态备份在读取或修改数据的过程中进行, 很少中断或者不中断传输或处理数据的功能。使用热备份时, 系统仍可供读取和修改数据的操作访问。

冷备份:

这些备份在用户不能访问数据时进行，因此无法读取或修改数据。这些脱机备份会阻止执行任何使用数据的活动。这些类型的备份不会干扰正常运行的系统的性能。但是，对于某些应用程序，会无法接受必须在一段较长的时间里锁定或完全阻止用户访问数据。

温备份：

这些备份在读取数据时进行，但在多数情况下，在进行备份时不能修改数据本身。这种中途备份类型的优点是不必完全锁定最终用户。但是，其不足之处在于无法在进行备份时修改数据集，这可能使这种类型的备份不适用于某些应用程序。在备份过程中无法修改数据可能产生性能问题。

1.2 备份的方式

1.2.1 冷备份

最简单的备份方式就是，关闭MySQL服务器，然后将data目录下面的所有文件进行拷贝保存，需要恢复时，则将目录拷贝到需要恢复的机器即可。这种方式确实方便，但是在生产环境中基本没什么作用。因为所有的机器都是要提供服务的，即使是Slave有时候也需要提供只读服务，所以关闭MySQL停服备份是不现实的。与冷备份相对应的一个概念是热备份，所谓热备份是在不影响MySQL对外服务的情况下，进行备份。

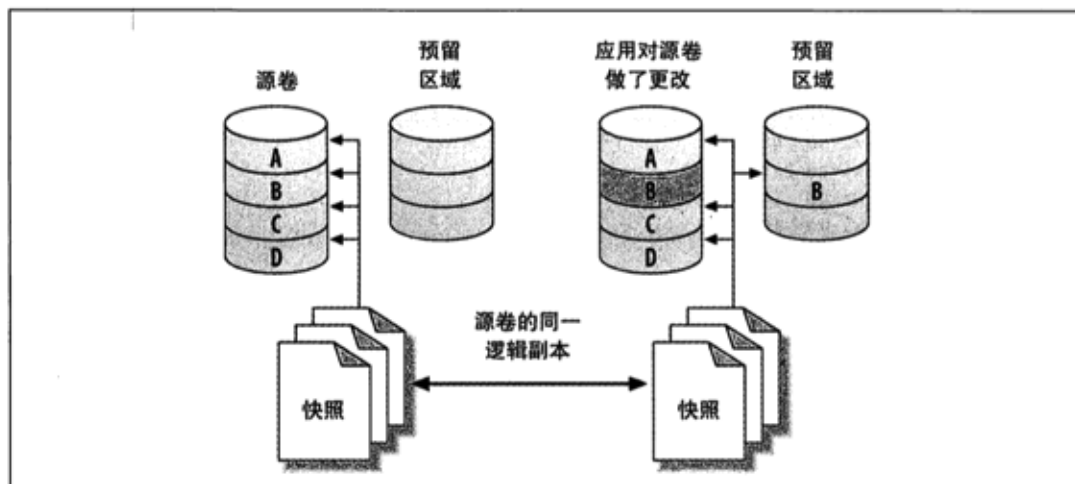
冷备份及停止业务进行备份。

1.2.2 快照备份

首先要介绍的热备份是快照备份，快照备份是指通过文件系统支持的快照功能对数据库进行备份。备份的原理是将所有的数据库文件放在同一分区中，然后对该分区执行快照工作，对于Linux而言，需要通过LVM(Logical Volumn Manager)来实现。LVM使用写时复制(copy-on-write)技术来创建快照，例如，对整个卷的某个瞬间的逻辑副本，类似于数据库中的innodb存储引擎的MVCC，只不过LVM的快照在文件系统层面，而MVCC在数据库层面，而且仅支持innodb存储引擎。

LVM有一个快照预留区域，如果原始卷数据有变化时，LVM保证在任何变更写入之前，会复制受影响块到快照预留区域。简单来说，快照区域内保留了快照点开始时的一致性的所有old数据。对于更新很少的数据库，快照也会非常小。

对于MySQL而言，为了使用快照备份，需要将数据文件，日志文件都放在一个逻辑卷中，然后对该卷快照备份即可。由于快照备份，只能本地，因此，如果本地的磁盘损坏，则快照也就损坏了。快照备份更偏向于对误操作防范，可以将数据库迅速恢复到快照产生的时间点，然后结合二进制日志可以恢复到指定的时间点。基本原理如下图所示：



1.2.3 逻辑备份（文本表示：SQL 语句）

冷备份和快照备份由于其弊端在生产环境中很少使用，使用更多是MySQL自带的逻辑备份和物理备份工具，本节主要讲逻辑备份，MySQL官方提供了Mysqldump逻辑备份工具，虽然已经足够好，但存在单线程备份慢的问题。在社区提供了更优秀的逻辑备份工具mydumper，它的优势主要体现在多线程备份，备份速度更快。

1.2.4 其他常用的备份方式

物理备份（数据文件的二进制副本）

全量备份概念

全量数据就是数据库中所有的数据（或某一个库的全部数据）；

全量备份就是把数据库中所有的数据进行备份。

mysqldump会取得一个时刻的一致性数据。

增量备份（刷新二进制日志）

增量数据就是指上一次全量备份数据之后到下一次全备之前数据库所更新的数据

对于mysqldump,binlog就是增量数据。

1.2.5 备份工具的介绍

- 1、mysqldump: mysql原生自带很好用的逻辑备份工具
- 2、mysqlbinlog: 实现binlog备份的原生态命令
- 3、xtrabackup: precona公司开发的性能很高的物理备份工具

1.3 mysqldump备份介绍

备份的基本流程如下：

- 1.调用FTWRL(flush tables with read lock)，全局禁止读写
- 2.开启快照读，获取此时的快照(仅对innodb表起作用)
- 3.备份非innodb表数据(*.frm,*.myi,*.myd等)
- 4.非innodb表备份完毕后，释放FTWRL锁
- 5.逐一备份innodb表数据
- 6.备份完成。

整个过程，可以参考我同事的一张图，但他的这张图只考虑innodb表的备份情况，实际上在unlock tables执行完毕之前，非innodb表已经备份完毕，后面的t1,t2和t3实质都是innodb表，而且5.6的mysqldump利用保存点机制，每备份完一个表就将一个表上的MDL锁释放，避免对一张表锁更长的时间。

1.3.1 mysqldump备份流程



1.3.2 常用的备份参数

参数	参数说明
-A	备份全库
-B	备某一个数据库下的所有表
-R, --routines	备份存储过程和函数数据
--triggers	备份触发器数据
	告诉你备份后时刻的binlog位置

<code>--master-data={1 2}</code>	如果等于1，则将其打印为CHANGE MASTER命令; 如果等于2，那么该命令将以注释符号为前缀。
<code>--single-transaction</code>	对innodb引擎进行热备
<code>-F, --flush-logs</code>	刷新binlog日志
<code>-x, --lock-all-tables</code>	锁定所有数据库的所有表。这是通过在整个转储期间采用全局读锁来实现的。
<code>-l, --lock-tables</code>	锁定所有表以供读取
<code>-d</code>	仅表结构
<code>-t</code>	仅数据
<code>--compact</code>	减少无用数据输出(调试)

一个完整的备份语句：
innodb引擎的备份命令如下：

```
mysqldump -A -R --triggers --master-data=2 --single-transaction |gzip >/opt/all_$(date +%F).sql.gz
```

适合多引擎混合（例如：myisam与innodb混合）的备份命令如下：

```
mysqldump -A -R --triggers --master-data=2 |gzip >/opt/all_$(date +%F).sql.gz
```

1.3.3 -A 参数

备份全库，备份语句

```
mysqldump -uroot -p123 -A > /backup/full.sql
```

1.3.4 -B 参数

备某一个数据库下的所有表

增加建库（create）及“use库”的语句，可以直接接多个库名，同时备份多个库* -B 库1 库2

```
mysqldump -uroot -p123 -B world > /backup/worldb.sql
```

备份语句：

```
create database if not 存在
use db1
drop table
create table
insert into
```

不加-B备份数据库时，只是备份数据库下的所有表，不会创建数据库

只能备份单独的数据库（一般用于备份单表时使用）

```
mysqldump -uroot -p123 world > /backup/world.sql
```

备份单表

```
mysqldump -uroot -p123 world city > /backup/world_city.sql
```

对于单表备份的粒度，再恢复数据库数据时速度最快。

备份多个表

```
mysqldump 库1 表1 表2 表3 >库1.sql
mysqldump 库2 表1 表2 表3 >库2.sql
```

分库备份:for循环

```
mysqldump -uroot -p'mysql123' -B mysql ...
mysqldump -uroot -p'mysql123' -B mysql_utf8 ...
mysqldump -uroot -p'mysql123' -B mysql ...
.....
```

分库备份

```
for name in `mysql -e "show databases;"|sed 1d`
do
  mysqldump -uroot -p'mysql123' -B $name
done
```

1.3.5 --master-data={1|2}参数

告诉你备份后时刻的binlog位置

2为注释 1为非注释，要执行的(主从复制)

```
[root@db02 logs]# sed -n '22p' /opt/t.sql
CHANGE MASTER TO MASTER_LOG_FILE='clsn-bin.000005', MASTER_LOG_POS=344;
[root@db02 logs]# mysqldump -B --master-data=2 clsn >/opt/t.sql
```

1.3.6 --single-transaction 参数

对innodb引擎进行热备

只支持innodb引擎

使用该参数会单独开启一个事务进行备份，利用事务的快照技术实现。

基于事务引擎:不用锁表就可以获得一致性的备份。

体现了ACID四大特性中的隔离性，生产中99% 使用innodb事务引擎。

虽然支持热备，并不意味着你可以再任意时间点进行备份，特别是业务繁忙期，不要做备份策略，一般夜里进行备份。

innodb引擎的备份命令如下：

```
mysqldump -A -B -R --triggers --master-data=2 --single-transaction |gzip >/opt/all.sql.gz
```

1.3.7 --flush-logs参数/-F

刷新binlog日志

每天晚上0点备份数据库

```
mysqldump -A -B -F >/opt/$(date +%F).sql
```

```
[root@db02 ~]# ll /application/mysql/logs/
-rw-rw---- 1 mysql mysql 168 Jun 21 12:06 clsn-bin.000001
-rw-rw---- 1 mysql mysql 168 Jun 21 12:06 clsn-bin.000002
-rw-rw---- 1 mysql mysql 210 Jun 21 12:07 clsn-bin.index
```

提示:每个库都会刷新一次.

1.3.8 压缩备份

压缩备份命令:

```
mysqldump -B --master-data=2 clsn|gzip >/opt/t.sql.gz
```

解压:

```
zcat t.sql.gz >t1.sql
gzip -d t.sql.gz #删压缩包
gunzip all_2017-12-22.sql.gz
```

一个完整的备份语句

innodb引擎的备份命令如下:

```
mysqldump -A -R --triggers --master-data=2 --single-transaction |gzip >/opt/all.sql.gz
```

适合多引擎混合(例如: myisam与innodb混合) 的备份命令如下:

```
mysqldump -A -R --triggers --master-data=2 |gzip >/opt/all_$(date +%F).sql.gz
```

1.3.9 使用Mysqldump备份进行恢复实践

备份innodb引擎数据库clsn并压缩:

```
mysqldump -B -R --triggers --master-data=2 clsn|gzip >/opt/all_$(date +%F).sql.gz
```

人为删除clsn数据库:

```
[root@db02 opt]# mysql -e "drop database clsn;"
[root@db02 opt]# mysql -e "show databases;"
```

恢复数据库:

```
使用gzip解压 gzip -d xxx.gz
shell> mysql opt/all_2017-1222.sql
或
mysql> set sql_log_bin=0;
Query OK, 0 rows affected (0.00 sec)
mysql> source /backup/all_2017-12-22.sql
```

验证数据:

```
[root@db02 opt]# mysql -e "use clsn;select * from test;"
```

1.4 【模拟】增量恢复企业案例

1.4.1 前提条件:

- 1.具备全量备份 (mysqldump) 。
- 2.除全量备份以外, 还有全量备份之后产生的所有binlog增量日志。

1.4.2 环境准备

(1)准备环境:

```
drop database clsn;
CREATE DATABASE clsn;
USE `clsn`;
CREATE TABLE `test` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
INSERT INTO `test` VALUES (1,'clsn'),(2,'znix'),(3,'inca'),(4,'zuma'),(5,'kaka');
```

查看创建好的数据

```
mysql> select * from test;
+----+-----+
| id | name |
+----+-----+
| 1  | clsn |
| 2  | znix |
| 3  | inca |
| 4  | zuma |
| 5  | kaka |
+----+-----+
5 rows in set (0.00 sec)
```

(2)模拟环境:

```
mkdir /data/backup -p
date -s "2017/12/22"
```

全备份:

```
mysqldump -B --master-data=2 --single-transaction clsn|gzip>/data/backup/clsn_$(date +%F).sql.gz
```

模拟增量:

```
mysql -e "use clsn;insert into test values(6,'haha');"
mysql -e "use clsn;insert into test values(7,'hehe');"
mysql -e "select * from clsn.test;"
```

(3)模拟误删数据:


```
date -s "2017/12/22 11:40"
mysql -e "drop database clsn;show databases;"
```

出现问题10分钟后,发现问题,删除了数据库了.

1.4.3 恢复数据准备

(1)采用iptables防火墙屏蔽所有应用程序的写入。

```
[root@clsn ~]# iptables -I INPUT -p tcp --dport 3306 ! -s 172.16.1.51 -j DROP
#<==非172.16.1.51禁止访问数据库3306端口。
```

或采用mysql 配置参数,但是需要重启数据库

```
--skip-networking
```

复制二进制日志文件

```
cp -a /application/mysql/logs/clsn-bin.* /data/backup/
```

截取日志

```
zcat clsn_2017-12-22.sql.gz >clsn_2017-12-22.sql
sed -n '22p' clsn_2017-12-22.sql
mysqlbinlog -d clsn --start-position=339 clsn-bin.000008 -r bin.sql
```

需要恢复的日志:

```
1.clsn_2017-12-22.sql
2.bin.sql
grep -i drop bin.sql
sed -i '/^drop.*/d' bin.sql
```

1.4.4 进行数据恢复

恢复数据

```
[root@db02 backup]# mysql <clsn_2017-12-22.sql
[root@db02 backup]# mysql -e "show databases;"
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| clsn |
| znix |
| performance_schema |
+-----+
```

查看数据库

```
mysql> select * from test;
+----+-----+
| id | name |
+----+-----+
```

```
+----+----+
| 1 | clsn |
| 2 | znix |
| 3 | inca |
| 4 | zuma |
| 5 | kaka |
+----+----+
5 rows in set (0.00 sec)
```

恢复增量数据:

```
[root@db02 backup]# mysql clsn <bin.sql
mysql> select * from test;
+----+----+
| id | name |
+----+----+
| 1 | clsn |
| 2 | znix |
| 3 | inca |
| 4 | zuma |
| 5 | kaka |
| 6 | haha |
| 7 | hehe |
+----+----+
7 rows in set (0.00 sec)
```

恢复完毕。

调整iptables允许用户访问。

1.4.5 多个binlog问题

```
mysqlbinlog -d clsn --start-position=339 clsn-bin.000009 clsn-bin.000010 -r bin1.sql

mysql clsn <bin1.sql
```

1.5 mysql数据库实际生产惨案

1.5.1 发生背景

- 1、mysql服务器会在每天夜里0点全量备份
- 2、某个开发人员某个阳光明媚的上午，喝着茶，优雅的误删除了clsn_oss（核心）数据库。
- 3、导致公司业务异常停止，无法正常提供服务。

1.5.2 怎么解决的

- 1、当前系统进行评估。

什么损坏了，有没有备份，

恢复数据时间（误操作的数据有关，备份、恢复策略），

恢复业务时间

- 2、恢复方案

- (1) 恢复0点的全备，到测试库
- (2) 恢复0点开始到故障时间点的binlog，到测试库
- (3) 将误操作的数据导出，恢复到生产库。
- (4) 检验数据是不是完整的（开发测试环境测试恢复成功数据库）
- (5) 检验完成之后，重新开启生产业务

1.5.3 项目总结

- 1、经过我的恢复处理，30分钟整体业务重新提供服务（速度慢。。。）
- 2、在以后的工作中制定严格的开发规范，开发，开发。
- 3、将来制定更好的架构方案。

1.6 备份工具的选择

数据量范围：30G -> TB级别

1.6.1 数据量大，变换量小

- (1) 全备分花费的成本较高，mysqldump+binlog实现全备 + 增量备份，缺点是恢复成本比备份时间成本还高
- (2) xtrabackup：可以较长时间做一次全备，其余时间都是增量，全量备份空间成本很高如果数据量在30G->TB级别的话，更推荐使用xtrabackup工具。

1.6.2 数据量小，变化量大

只需要考虑时间成本。

只用全备备份即可，两种工具选择都可以。恢复成本上xtrabackup小一些

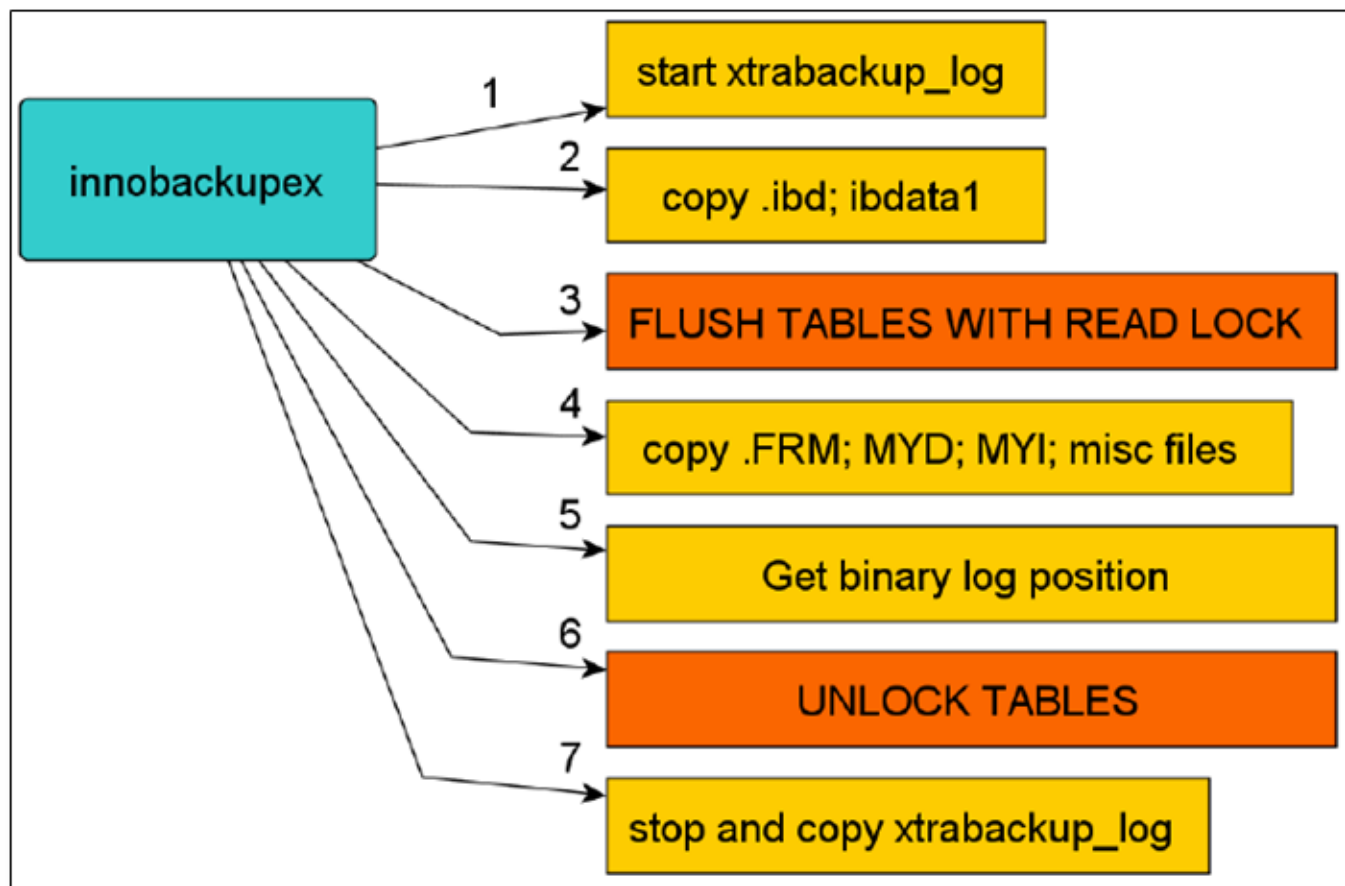
1.6.3 数据量、变化量都大

时间成本和空间成本都要考虑了。

数据量达到PB或更高时（facebook），mysqldump可能成为首选，占用空间小，但技术成本高。需要对mysqldump进行二次开发（大数据量公司首选）。

1.7 xtrabackup备份软件

percona公司官网 <https://www.percona.com/>



1.7.1 Xtrabackup介绍

Xtrabackup是由percona开源的免费数据库热备份软件，它能对InnoDB数据库和XtraDB存储引擎的数据库非阻塞地备份（对于MyISAM的备份同样需要加表锁）；mysqldump备份方式是采用的逻辑备份，其最大的缺陷是备份和恢复速度较慢，如果数据库大于50G，mysqldump备份就不太适合。

Xtrabackup安装完成后有4个可执行文件，其中2个比较重要的备份工具是**innobackupex**、**xtrabackup**

- 1) **xtrabackup** 是专门用来备份InnoDB表的，和mysql server没有交互；
- 2) **innobackupex** 是一个封装xtrabackup的Perl脚本，支持同时备份innodb和myisam，但在对myisam备份时需要加
- 3) **xbcrypt** 加密解密备份工具
- 4) **xbstream** 流传打包传输工具，类似tar
- 5) 物理备份工具，在同级数据量基础上，都要比逻辑备份性能好的多，特别是在数据量较大的时候，体现的更加明

1.7.1 Xtrabackup优点

- 1) 备份速度快，物理备份可靠
- 2) 备份过程不会打断正在执行的事务（无需锁表）
- 3) 能够基于压缩等功能节约磁盘空间和流量
- 4) 自动备份校验
- 5) 还原速度快
- 6) 可以流传将备份传输到另外一台机器上
- 7) 在不增加服务器负载的情况备份数据

8) 物理备份工具, 在同级数据量基础上, 都要比逻辑备份性能要好的多。几十G到不超过TB级别的条件下。但在同数据量级别, 物理备份恢复数据上有一定优势。

1.7.2 备份原理

拷贝数据文件、拷贝数据页

对于innodb表可以实现热备。

- (1)在数据库还有修改操作的时刻, 直接将数据文件备走, 此时, 备份走的数据对于当前mysql来讲是不一致的
- (2)将备份过程中的redo和undo一并备走。
- (3)为了恢复的时候, 只要保证备份出来的数据页lsn能和redo lsn匹配, 将来恢复的就是一致的数据。redo应

对于myisam表实现自动锁表拷贝文件。

备份开始时首先会开启一个后台检测进程, 实时检测mysql redo的变化, 一旦发现有新的日志写入, 立刻将日志记入后台日志文件xtrabackup_log中, 之后复制innodb的数据文件—系统表空间文件ibdatax, 复制结束后, 将执行flush tables with readlock,然后复制.frm MYI MYD等文件, 最后执行unlock tables,最终停止xtrabackup_log

1.7.3 xtrabackup的安装

安装依赖关系

```
wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-6.repo
yum -y install perl perl-devel libaio libaio-devel perl-Time-HiRes perl-DBD-MySQL
```

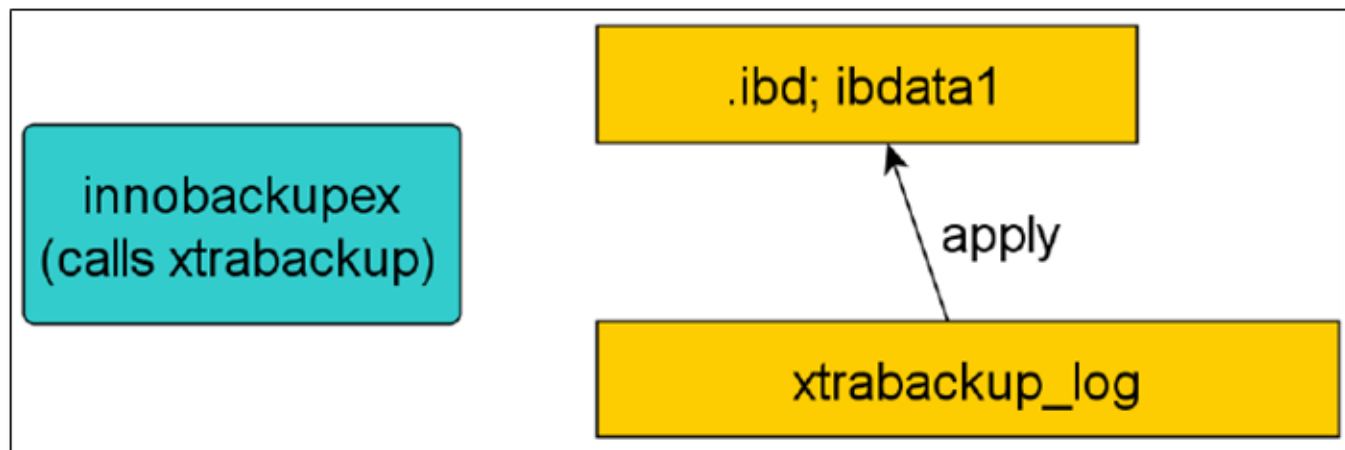
下载软件包, 并安装软件

```
wget https://www.percona.com/downloads/XtraBackup/Percona-XtraBackup-2.4.4/binary/redhat/6/x86_64/percona-xtrabackup-24-2.4.4-1.el6.x86_64.rpm
yum -y install percona-xtrabackup-24-2.4.4-1.el6.x86_64.rpm
```

1.8 xtrabackup实践操作

1.8.1 全量备份与恢复

这一阶段会启动xtrabackup内嵌的innodb实例, 回放xtrabackup日志xtrabackup_log, 将提交的事务信息变更应用到innodb数据/表空间, 同时回滚未提交的事务(这一过程类似innodb的实例恢复)。恢复过程如下图:



备份

创建备份目录

```
mkdir /backup -p
```

进行第一次全量备份

```
innobackupex --defaults-file=/etc/my.cnf --user=root --password=123 --socket=/application/mysql/tmp/
```

恢复前准备

恢复数据前的准备(合并xtrabackup_log_file和备份的物理文件)

```
innobackupex --apply-log --use-memory=32M /backup/xfull/
```

查看合并后的 checkpoints 其中的类型变为 full-prepared 即为可恢复。

```
[root@db02 full1]# cat xtrabackup_checkpoints
backup_type = full-prepared
from_lsn = 0
to_lsn = 4114824
last_lsn = 4114824
compact = 0
recover_binlog_info = 0
```

破坏数据库数据文件

```
[root@db02 full1]# cd /application/mysql/data/
[root@db02 data]# ls
auto.cnf  db02.pid  ibdata2    mysql      mysql-bin.index  world
clsn      haha      ib_logfile0 mysql-bin.000001  oldboy
db02.err  ibdata1   ib_logfile1 mysql-bin.000002  performance_schema
[root@db02 data]# \rm -rf .//*
[root@db02 data]# ls
[root@db02 data]# killall mysql
```

恢复方法

方法一:直接将备份文件复制回来

```
cp -a /backup/full/ /application/mysql/data
chown -R mysql:mysql /application/mysql/data
```

方法二:使用innobackupex命令进行恢复(推荐)

```
[root@db02 mysql]# innobackupex --copy-back /backup/xfull
[root@db02 mysql]# chown -R mysql:mysql /application/mysql/
```

说明：无论使用那种恢复方法都要恢复后需改属组属主，保持与程序一致。

```
[root@db02 data]# cd /application/mysql/data/
[root@db02 data]# ls
clsn      ibdata2      ibtmp1      performance_schema      xtrabackup_info
haha      ib_logfile0  mysql      world
ibdata1   ib_logfile1  oldboy     xtrabackup_binlog_pos_innodb
```

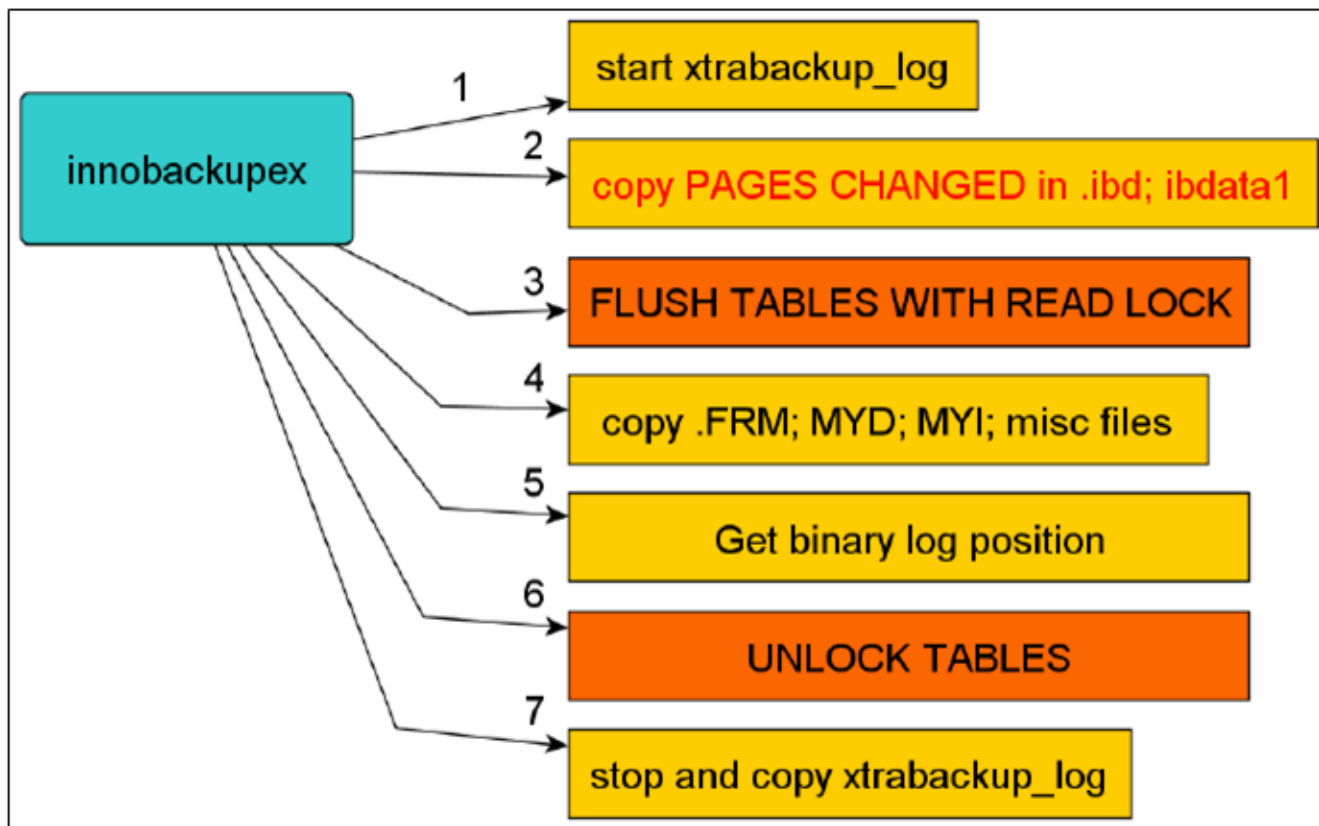
启动是数据库

```
[root@db02 data]# /etc/init.d/mysqld start
```

1.8.2 增量备份与恢复

innobackupex增量备份过程中的“增量”处理，其实主要是相对innodb而言，对myisam和其他存储引擎而言，它仍然是全拷贝(全备份)

“增量”备份的过程主要是通过拷贝innodb中有变更的“页”（这些变更的数据页指的是“页”的LSN大于xtrabackup_checkpoints中给定的LSN）。增量备份是基于全备的，第一次增备的数据必须要基于上一次的全备，之后的每次增备都是基于上一次的增备，最终达到一致性的增备。增量备份的过程如下，和全备的过程很类似，区别仅在第2步。



增量备份从哪增量？

基于上一次的备份进行增量。

redo默认情况下是一组两个文件，并且有固定大小。其使用的文件是一种轮询使用方式，他不是永久的，文件随时可能被覆盖。

注意：千万不要在业务繁忙时做备份。

备份什么内容

- 1、可以使用binlog作为增量
- 2、自带的增量备份，基于上次备份后的变化的数据页，还要备份在备份过程中的undo、redo变化

怎么备份

- 1、先进行第一次全备

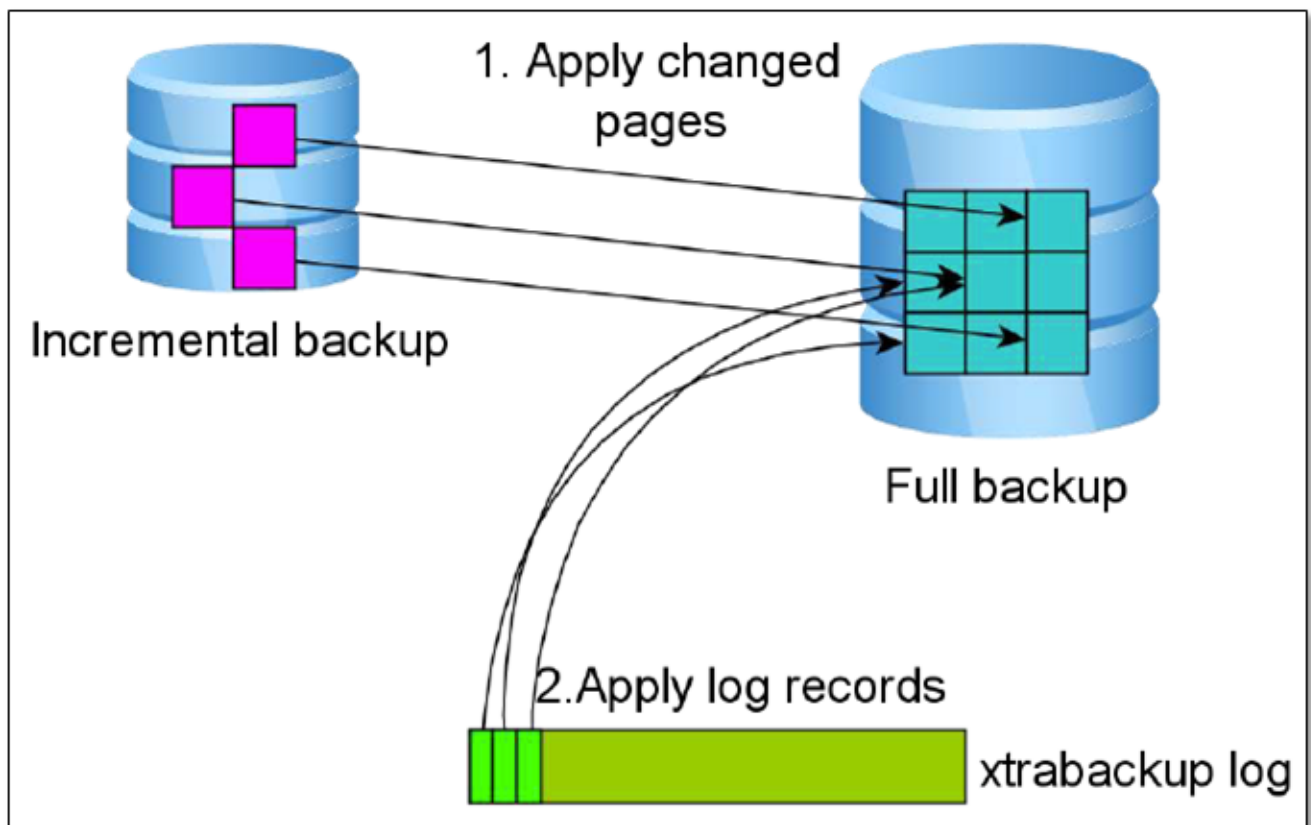
```
innobackupex --user=root --password=123 --no-timestamp /backup/xfull
```

对原库做了修改，修改了小红那行然后commit。

- 2、再进行增量备份

```
innobackupex --user=root --password=123 --incremental --no-timestamp --incremental-basedir=/backup/
```

怎么恢复



- 1、先应用全备日志（--apply-log，暂时不需要做回滚操作--redo-only）

```
innobackupex --apply-log --redo-only /backup/xfull/
```

- 2、合并增量到全备中（一致性的合并）

```
innobackupex --apply-log --incremental-dir=/backup/xinc1 /backup/xfull/  
innobackupex --apply-log /backup/xfull
```


3、合并完成进行恢复

方法一:直接将备份文件复制回来

```
cp -a /backup/full/ /application/mysql/data
chown -R mysql:mysql /application/mysql/data
```

方法二:使用innobackupex命令进行恢复(推荐)

```
[root@db02 mysql]# innobackupex --copy-back /backup/xfull
[root@db02 mysql]# chown -R mysql:mysql /application/mysql/
```

说明: 无论使用那种恢复方法都要恢复后需改属组属主, 保持与程序一致。

1.8.3 数据库备份策略

每周的周日进行一次全备; 周一到周六每天做上一天增量, 每周轮询一次。

xfull	--apply-log --redo-only 保证last-lsn=周一增量开始lsn
xinc1	合并周一的增量到全备, 并apply-log --redo-only 保证last-lsn=周二增量开始lsn
xinc2	合并周二的增量到全备, 并apply-log --redo-only 保证last-lsn=周三增量开始lsn
xinc3	合并周三的增量到全备, 并apply-log --redo-only 保证last-lsn=周四增量开始lsn
xinc4	合并周四的增量到全备, 并apply-log --redo-only 保证last-lsn=周五增量开始lsn
xinc5	合并周五的增量到全备, 并apply-log --redo-only 保证last-lsn=周六增量开始lsn
xinc6	合并周六的增量到全备, --apply-log 准备恢复即可

1.8.4 真实生产实战案例分析

背景: 某物流公司网站核心系统, 数据量是220G, 每日更新量100M-200M

备份方案: xtrabackup全备+增量

备份策略 (crontab) :

1、周六 晚上0点全备

0 0 * * 6 zjs_full.sh —这行可以没有

2、周一至周五、周日 是增量, 基于上一天增量

0 1 * * 0-5 zjs_inc.sh —这行可以没有

故障场景:

周三的时候, 下午两点, 开发人员误删除了一张表zjs_base, 大约10G。

项目职责:

- 1) 指定恢复方案、利用现有备份;
- 2) 恢复误删除数据;

3) 制定运维、开发流程规范。

恢复流程：

- a) 准备上周六全备。
- b) 合并周日、周一、周二增量。
- c) 在测试库恢复以上数据，数据的目前状态应该周三凌晨**1:00**
- d) 需要恢复的数据状态是，下午**2**点钟左右
- e) 从**1**点开始的binlog恢复到删除之前转台
- f) 导出删除的表zjs_base，恢复到生产库，验证数据可用性、完整性。
- g) 启动应用连接数据库。

总结：经过30分钟将误删表恢复了。服务总共停止40分钟。

1.8.5 故障恢复小结

恢复思路：

- 1、首先确保断开所有应用，保证数据的安全。
- 2、检查用于恢复的备份存在吗。
- 3、设计快速、安全恢复简单方案，制定突发问题解决办法。

具体恢复流程：

- 1、准备上周六全备，并--apply-log --redo-only
- 2、合并增量，周日、周一、周二 --apply-log --redo-only 周三 --apply-log
- 3、在测试库恢复以上数据，数据的目前状态应该周三凌晨**1:00**
- 4、需要恢复的数据状态是，下午**2**点钟左右，删除zjs_base之前的数据状态
从**1**点开始的binlog恢复到删除之前的那个events的position。
- 5、导出删除的表zjs_base，恢复到生产库，验证数据可用性、完整性。
- 6、启动应用连接数据库。

确定恢复所需时间

恢复窗口要多长？----> 预计**3**小时

和你恢复+验证+意外情况有关。

业务停多长时间？----> **6**小时？或者更多？更少？

1.8.6 【模拟】生产事故恢复

数据创建阶段

- 1、创建备份需要的目录

```
mkdir full inc1 inc2
```

- 2、周日全备

```
innobackupex --user=root --password=123 --no-timestamp /backup/xbackup/full/
```

3、模拟数据变化

```
use oldboy
create table test(id int,name char(20),age int);
insert into test values(8,'outman',99);
insert into test values(9,'outgirl',100);
commit;
```

4、周一增量备份

```
innobackupex --user=root --password=123 --incremental --no-timestamp --incremental-basedir=/backup/x
```

5、模拟数据变化

```
use oldboy
insert into test values(8,'outman1',119);
insert into test values(9,'outgirl1',120);
commit;
```

6、周二的增量备份

```
innobackupex --user=root --password=123 --incremental --no-timestamp --incremental-basedir=/backup/x
```

7. 再插入新的行操作

```
use oldboy
insert into test values(10,'outman2',19);
insert into test values(11,'outgirl2',10);
commit;
```

模拟误操作事故

模拟场景，周二下午2点误删除test表

```
use oldboy;
drop table test;
```

准备恢复数据

1.准备xtrabackup备份，合并备份

```
innobackupex --apply-log --redo-only /backup/xbackup/full
innobackupex --apply-log --redo-only --incremental-dir=/backup/xbackup/inc1 /backup/xbackup/full
innobackupex --apply-log --incremental-dir=/backup/xbackup/inc2 /backup/xbackup/full
innobackupex --apply-log /backup/xbackup/full
```

2. 确认binlog起点, 准备截取binlog。

```
cd /backup/xbackup/inc2/
cat xtrabackup_binlog_info
mysql-bin.000001 1121
```

3. 截取到drop操作之前的binlog

```
mysqlbinlog --start-position=1121 /tmp/mysql-bin.000003
找到drop之前的event和position号做日志截取, 假如 1437
mysqlbinlog --start-position=1121 --stop-position=1437 /tmp/mysql-bin.000003 >/tmp/incbinlog
```

4. 关闭数据库、备份二进制日志

```
/etc/init.d/mysqld stop
cd /application/mysql/data/
cp mysql-bin.000001 /tmp
```

5. 删除MySQL所有数据

```
cd /application/mysql/data/
rm -rf *
```

恢复数据

1. 将全量备份的数据恢复到数据目录下

```
innobackupex --copy-back /backup/xbackup/full/
chown -R mysql:mysql /application/mysql/data/
/etc/init.d/mysqld start
```

2. 恢复binlog记录

```
set sql_log_bin=0
source /tmp/incbinlog.sql
```

1.8.7 xtarbackup导出

(1)“导出”表 导出表是在备份的prepare阶段进行的, 因此, 一旦完全备份完成, 就可以在prepare过程中通过export选项将某表导出了:

```
innobackupex --apply-log --export /path/to/backup
```

此命令会为每个innodb表的表空间创建一个以.exp结尾的文件，这些以.exp结尾的文件则可以用于导入至其它服务器。

(2)“导入”表 要在mysql服务器上导入来自于其它服务器的某innodb表，需要先在当前服务器上创建一个跟原表结构一致的表，而后才能实现将表导入：

```
mysql> CREATE TABLE mytable (...) ENGINE=InnoDB;
```

然后将此表的表空间删除：

```
mysql> ALTER TABLE mydatabase.mytable DISCARD TABLESPACE;
```

接下来，将来自于“导出”表的服务器的mytable表的mytable.ibd和mytable.exp文件复制到当前服务器的数据目录，然后使用如下命令将其“导入”：(记得改权限)

```
mysql> ALTER TABLE mydatabase.mytable IMPORT TABLESPACE;
```

示例:

```
innobackupex --user=root --password=123 --no-timestamp /backup/xbackup/full/
```

进入到全备的数据库目录下

```
[root@db02 haha]# ls
db.opt  PENALTIES.frm  PENALTIES.ibd  PLAYERS.frm  PLAYERS.ibd
[root@db02 haha]# pwd
/backup/xbackup/full/haha
```

导出表

```
[root@db02 haha]# innobackupex --apply-log --export /backup/xbackup/full/
[root@db02 haha]# ls
db.opt      PENALTIES.exp  PENALTIES.ibd  PLAYERS.exp  PLAYERS.ibd
PENALTIES.cfg  PENALTIES.frm  PLAYERS.cfg    PLAYERS.frm
```

创建出同结构表

```
CREATE TABLE `PLAYERS` (
  `PLAYERNO` int(11) NOT NULL,
  `NAME` char(15) NOT NULL,
  `INITIALS` char(3) NOT NULL,
  `BIRTH_DATE` date DEFAULT NULL,
  `SEX` char(1) NOT NULL,
  `JOINED` smallint(6) NOT NULL,
```

```
`STREET` varchar(30) NOT NULL,  
`HOUSENO` char(4) DEFAULT NULL,  
`POSTCODE` char(6) DEFAULT NULL,  
`TOWN` varchar(30) NOT NULL,  
`PHONENO` char(13) DEFAULT NULL,  
`LEAGUENO` char(4) DEFAULT NULL,  
PRIMARY KEY (`PLAYERNO`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

复制恢复数据到库下

```
[root@db02 haha]# cp PLAYERS.ibd PLAYERS.exp /application/mysql/data/backup/  
cp: overwrite `/application/mysql/data/backup/PLAYERS.ibd'? y
```

恢复数据

```
mysql> ALTER TABLE backup.PLAYERS DISCARD TABLESPACE;  
Query OK, 0 rows affected (0.00 sec)
```

1.8.8 innobackupex参数说明

参数	参数说明
-compress	该选项表示压缩innodb数据文件的备份。
-compress-threads	该选项表示并行压缩worker线程的数量。
-compress-chunk-size	该选项表示每个压缩线程worker buffer的大小，单位是字节，默认是64K。
-encrypt	该选项表示通过ENCRYPTION_ALGORITHM的算法加密innodb数据文件的备份，目前支持的算法有ASE128,AES192,AES256。
-encrypt-threads	该选项表示并行加密的worker线程数量。
-encrypt-chunk-size	该选项表示每个加密线程worker buffer的大小，单位是字节，默认是64K。
-encrypt-key	该选项使用合适长度加密key，因为会记录到命令行，所以不推荐使用。
-encryption-key-file	该选项表示文件必须是一个简单二进制或者文本文件，加密key可通过以下命令行命令生成：opensslrand -base64 24。
-include	该选项表示使用正则表达式匹配表的名字[db.tb]，要求为其指定匹配要备份的表的完整名称，即databasename.tablename。
-user	该选项表示备份账号。
	该选项表示备份的密码。

password	
-port	该选项表示备份数据库的端口。
-host	该选项表示备份数据库的地址。
-databases	该选项接受的参数为数据名，如果要指定多个数据库，彼此间需要以空格隔开；如：“xtra_test dba_test”，同时，在指定某数据库时，也可以只指定其中的某张表。如：“mydatabase.mytable”。该选项对innodb引擎表无效，还是会备份所有innodb表。此外，此选项也可以接受一个文件为参数，文件中每一行为一个要备份的对象。
-tables-file	该选项表示指定含有表列表的文件，格式为database.table，该选项直接传给tables-file。
-socket	该选项表示mysql.sock所在位置，以便备份进程登录mysql。
-no-timestamp	该选项可以表示不要创建一个时间戳目录来存储备份，指定到自己想要的备份文件夹。
-ibbackup	该选项指定了使用哪个xtrabackup二进制程序。IBBACKUP-BINARY是运行percona xtrabackup的命令。这个选项适用于xtrabackup二进制不在你搜索和工作目录，如果指定了该选项，innobackupex自动决定用的二进制程序。
-slave-info	该选项表示对slave进行备份的时候使用，打印出master的名字和binlog pos，同样将这些信息以change master的命令写入xtrabackup_slave_info文件。可以通过基于这份备份启动一个从库。
-safe-slave-backup	该选项表示为保证一致性复制状态，这个选项停止SQL线程并且等到show status中的slave_open_temp_tables为0的时候开始备份，如果没有打开临时表，backup会立刻开始，否则SQL线程启动或者关闭知道没有打开的临时表。如果slave_open_temp_tables在-safe-slave-backup-timeout（默认300秒）秒之后不为0，从库sql线程会在备份完成的时候重启。
-kill-long-queries-timeout	该选项表示从开始执行FLUSH TABLES WITH READ LOCK到kill掉阻塞它的这些查询之间等待的秒数。默认值为0，不会kill任何查询，使用这个选项xtrabackup需要有Process和super权限。
-kill-long-query-type	该选项表示kill的类型，默认是all，可选select。
-ftwrl-wait-threshold	该选项表示检测到长查询，单位是秒，表示长查询的阈值。
-ftwrl-wait-query-type	该选项表示获得全局锁之前允许那种查询完成，默认是ALL，可选update。
	该选项表示生成了包含创建备份时候本地节点状态的文件xtrabackup_galera_info文件，该选项只

-galera-info	适用于备份PXC。
-stream	该选项表示流式备份的格式，backup完成之后以指定格式到STDOUT，目前只支持tar和xbstream。
-defaults-file	该选项指定了从哪个文件读取MySQL配置，必须放在命令行第一个选项的位置。
-defaults-extra-file	该选项指定了在标准defaults-file之前从哪个额外的文件读取MySQL配置，必须在命令行的第一个选项的位置。一般用于存备份用户的用户名和密码的配置文件。
---defaults-group	该选项表示从配置文件读取的组，innobackupex多个实例部署时使用。
-no-lock	该选项表示关闭FTWRL的表锁，只有在所有表都是InnoDB表并且不关心backup的binlog pos点，如果有任何DDL语句正在执行或者非InnoDB正在更新时（包括mysql库下的表），都不应该使用这个选项，后果是导致备份数据不一致，如果考虑备份因为获得锁失败，可以考虑safe-slave-backup立刻停止复制线程。
-tmpdir	该选项表示指定-stream的时候，指定临时文件存在哪里，在streaming和拷贝到远程server之前，事务日志首先存在临时文件里。在使用参数stream=tar备份的时候，你的xtrabackup_logfile可能会临时放在/tmp目录下，如果你备份的时候并发写入较大的话 xtrabackup_logfile可能会很大(5G+)，很可能会撑满你的/tmp目录，可以通过参数-tmpdir指定目录来解决这个问题。
-history	该选项表示percona server 的备份历史记录在percona_schema.xtrabackup_history表。
-incremental	该选项表示创建一个增量备份，需要指定-incremental-basedir。
-incremental-basedir	该选项表示接受了一个字符串参数指定含有full backup 的目录为增量备份的base目录，与-incremental同时使用。
-incremental-dir	该选项表示增量备份的目录。
-incremental-force-scan	该选项表示创建一份增量备份时，强制扫描所有增量备份中的数据页。
-incremental-lsn	该选项表示指定增量备份的LSN，与-incremental选项一起使用。
-incremental-	该选项表示存储在PERCONA_SCHEMA.xtrabackup_history基于增量备份的历史记录的名字。Percona Xtrabackup搜索历史表查找最近（innodb_to_lsn）成功备份并且

history-name	将 to_lsn 值 作为 增量 备份 启动 出事 lsn. 与 innobackupex-incremental-history-uuid 互斥。如果没有检测到有效的lsn，xtrabackup会返回error。
- incremental-history-uuid	该 选 项 表 示 存 储 在percona_schema.xtrabackup_history基于增量备份的特定历史记录的UUID。
-close-files	该选项表示关闭不再访问的文件句柄，当xtrabackup打开表空间通常并不关闭文件句柄目的是正确的处理DDL操作。如果表空间数量巨大，这是一种可以关闭不再访问的文件句柄的方法。使用该选项有风险，会有产生不一致备份的可能。
- compact	该选项表示创建一份没有辅助索引的紧凑的备份。
- throttle	该选项表示每秒IO操作的次数，只作用于backup阶段有效。apply-log和-copy-back不生效不要一起用。

1.9 参考文献


```
https://www.cnblogs.com/cchust/p/5452557.html
http://www.cnblogs.com/gomysql/p/3650645.html  xtrabackup 详解
https://www.percona.com/software/mysql-database/percona-xtrabackup
https://learn.percona.com/hubfs/Manuals/Percona_Xtra_Backup/Percona_XtraBackup_2.4/Percona-XtraBacku
```

赞0

如无特殊说明，文章均为本站原创，转载请注明出处

- 转载请注明来源：Mysql 备份恢复与xtrabackup备份
- 本文永久链接地址：https://www.nmtui.com/clsn/lx346.html

该文章由 惨绿少年 发布

惨绿少年Linux www.nmtui.com