# 背景

备份的几个重要理由：

1. 灾难恢复
2. 审计
3. 测试
4. 误操作恢复

建立数据库后，下一个重要的事情就是设置备份。做备份的方法主要有两种，一种是逻辑备份，它将所有数据库、表结构、数据和存储例程导出到一组可以再次执行的SQL语句中，以重新创建数据库的状态；另一种是物理备份，它包含了系统上的所有文件，这里的系统是指数据库用于存储所有数据库实体的系统。

● 逻辑备份工具：mysqldump、mysqlpump和mydumper（不随MySQL提供）。

● 物理备份工具：XtraBackup（不随MySQL提供）和普通文件备份。

对于时间点恢复，备份应该能够提供开始做备份之前的二进制日志的位置。这被称为连续的备份。

强烈建议从一个从（slave）服务器备份到mount于其上的文件中。

# 概述

## frm

与表相关的元数据信息都存放在.frm文件中，主要是表结构的定义信息，不论什么存储引擎，每一个表都会有一个以表名命名的.frm文件。

## .MYD和.MYI

.MYD：MY Data，是MyISAM存储引擎专用的用于存放MyISAM表的数据；

.MYI：MY Index，也是专属于MyISAM存储引擎的主要存放MyISAM表的索引相关信息。

## .ibd和.ibdata

两者都是专属于InnoDB存储引擎的数据库文件。

当采用共享表空间时所有InnoDB表的数据均存放在.ibdata中，所以当表越来越多时，这个文件会变得很大；相对应的.ibd就是采用独享表空间时InnoDB表的数据文件。

之所以有两种文件来存放Innodb的数据（包括索引），是因为Innodb的数据存储方式能够通过配置来决定是使用共享表空间存放存储数据，还是独享表空间存放存储数据。独享表空间存储方式使用“.ibd”文件来存放数据，且每个表一个“.ibd”文件，文件存放在和MyISAM数据相同的位置。如果选用共享存储表空间来存放数据，则会使用ibdata文件来存放，所有表共同使用一个（或者多个，可自行配置）ibdata文件。

## .opt

此文件在每一个自建的库里都会有，记录这个库的默认使用的字符集和校验规则。

## rbd

# 备份/恢复策略

对于一个 DBA来说，定制合理的备份策略无疑是很重要的，以下是我们在进行备份或恢复操作时需要考虑的一些因素。

确定要备份的表的存储引擎是事务型还是非事务型，两种不同的存储引擎备份方式在处理数据一致性方面是不太一样的。

确定使用全备份还是增量备份。全备份的优点是备份保持最新备份，恢复的时候可以花费更少的时间；缺点是如果数据量大，将会花费很多的时间，并对系统造成较长时间的压力。增量备份则恰恰相反，只需要备份每天的增量日志，备份时间少，对负载压力也小；缺点就是恢复的时候需要全备份加上次备份到故障前的所有日志，恢复时间会长一些。

可以考虑采取复制的方法来做异地备份，但是记住，复制不能代替备份，它对数据库的误操作也无能为力。

要定期做备份，备份的周期要充分考虑系统可以承受的恢复时间。备份要在系统负载较小的时候进行。

确保MySQL打开log-bin选项，有了BINLOG，MySQL才可以在必要的时候做完整恢复，或基于时间点的恢复，或基于位置的恢复。

要经常做备份恢复测试，确保备份是有效的，并且是可以恢复的。

参考：《MySQL DBA修炼之道》

《跟老男孩学Linux运维：MySQL数据库入门与提高实践》

# 数据库备份

## 背景

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：

1. 硬件故障
2. 软件故障
3. 自然灾害
4. 黑客攻击
5. 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：

1. 能够容忍丢失多少数据
2. 恢复数据需要多长时间
3. 需要恢复哪一些数据

## 概述

一般情况下, 我们需要备份的数据分为以下几种：

1、数据

2、二进制日志, InnoDB事务日志

3、代码(存储过程、存储函数、触发器、事件调度器)

4、服务器配置文件

**备份和冗余的区别：**

备份：能够防止由于机械故障以及人为误操作带来的数据丢失，例如将数据库文件保存在了其他地方。

冗余：数据有多份冗余，但不等备份，只能防止机械故障带来的数据丢失，例如主备模式、数据库集群。

## 分类

### 完全备份/部分备份

根据备份数据集可以分为：

1. 完全备份
2. 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)

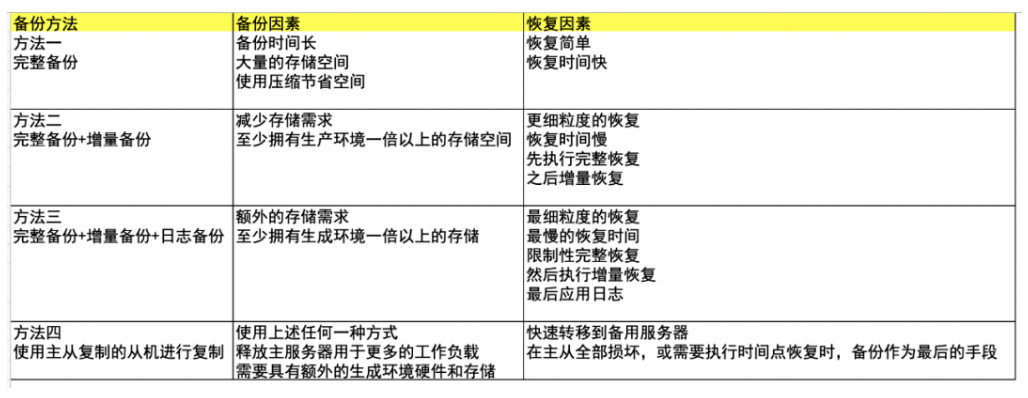
#### 完整备份

完整备份：完全备份会备份数据完整的副本。如果不需要恢复到特定的时间点，那么只使用完整备份的策略即可满足需要。例如，如果数据不经常更改或不重要，并且可以容忍损失几天的数据，那么用户可以每天甚至每周运行完整的备份。

完整备份+增量备份：用增量备份补充完整备份，使用户能够更频繁地运行备份，并将恢复时间减少到几个小时内。增量备份不需要太多的磁盘空间，因此可以减少存储需求和成本。

完整备份+增量备份+日志：如果需要在几分钟内恢复，或者恢复到特定时间点，那么将完整备份和增量备份与事务日志备份结合起来是非常合适的。可以使用事务日志的备份应用完整备份、增量备份和前滚。

除了上述方法，备份还需要考虑备份时长、恢复时长、恢复级别、存储要求及存储的限制等因素。综合考虑后，制定适合自己的备份策略。



#### 增量备份/差异备份

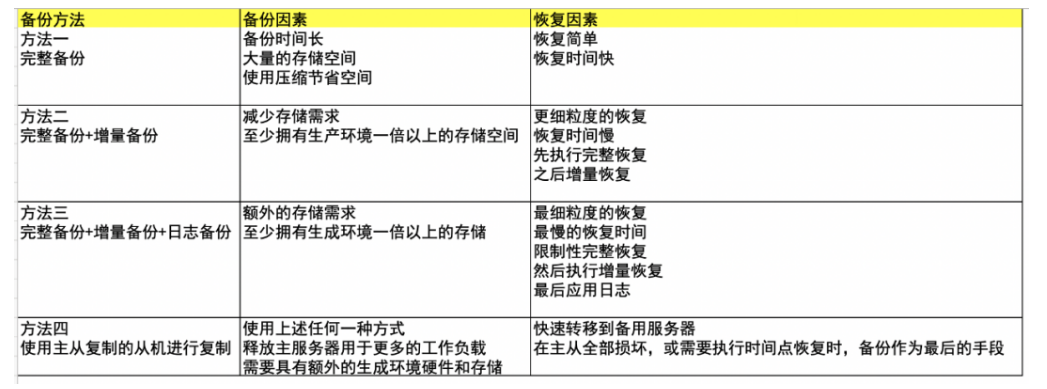
而部分备份又分为以下两种：

1. 增量备份
2. 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。

除了上述方法，备份还需要考虑备份时长、恢复时长、恢复级别、存储要求及存储的限制等因素。综合考虑后，制定适合自己的备份策略。



### 冷备/热备/温备

**在MySQl中我们备份数据一般有几种方式：**

热备份：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，比如mysqlhotcopy（MyISAM），ibbackup（InnoDB），xtrabackup。

温备份：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：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，比如cp。

**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 √

### 物理备份/逻辑备份

根据MySQL数据库中数据的备份方式：

**物理备份：**

直接复制数据库文件。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。

适用于大型数据库环境，不受存储引擎的限制，但不能恢复不同的MySQL版本。一般是在数据库彻底关闭或者不能完成正常提供服务的前提下进行的备份，如tar、cp、xtrabackup（数据库可以正常提供服务）、lvm snapshot、rsync等。

**逻辑备份：**

备份的是建表、建库、插入等操作所执行的SQL语句（DDL、DML、DCL）。

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)。

适用于中小型数据库，效率相对较低。一般在数据库正常提供服务的前提下进行，如mysqldump、mydumper、select into outfile（表的导入导出）等。

## 备份策略

备份过程必须要考虑的因素：

1. 必须制定详细的备份计划（备份频率、时间点、周期）
2. 备份数据应该放在非数据库本地，并建议有多份副本
3. 必须做好数据恢复的演练（每隔一段时间，对备份的数据在测试环境中进行模拟恢复，保证当出现数据灾难的时候能够及时恢复数据）
4. 根据数据应用的场合、特点选择正确的备份工具
5. 数据的一致性
6. 服务的可用性

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：

1、直接cp/tar复制数据库文件

2、mysqldump+复制BIN LOGS

3、lvm2快照+复制BIN LOGS

4、xtrabackup

**以上的几种解决方案分别针对于不同的场景：**

1. 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
2. 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
3. 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
4. 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

## 逻辑备份

逻辑备份工具主要有：mysqldump、mysqlpump、mydumper，物理备份工具主要有：xtrabackup。

**使用mysqldump命令备份**

　　mysqldump命令将数据库中的数据备份成一个文本文件。表的结构和表中的数据将存储在生成的文本文件中。

　　mysqldump先查出需要备份的表的结构，再在文本文件中生成一个CREATE语句。然后，将表中的所有记录转换成一条INSERT语句。然后通过这些语句，就能够创建表并插入数据。

### mysqldump

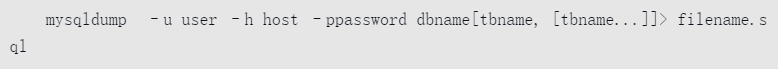
#### 概述

**参考：**

[https://mp.weixin.qq.com/s?\_\_biz=MzI3NDA4OTk1OQ==&mid=2649900721&idx=1&sn=6cea6e9d1be8fd543d8c5e824b3b2ee8&mpshare=1&scene=24&srcid=#rd](https://mp.weixin.qq.com/s?__biz=MzI3NDA4OTk1OQ==&mid=2649900721&idx=1&sn=6cea6e9d1be8fd543d8c5e824b3b2ee8&mpshare=1&scene=24&srcid=" \l "rd)

mysqldump是一个广泛使用的逻辑备份工具。它提供了多种选项来包含或排除数据库、选择要备份的特定数据、仅备份不包含数据的schema，或者只备份存储的例程而不包括其他任何东西，等等。

MySQLdump是MySQL提供的一个非常有用的数据库备份工具。MySQLdump命令执行时，可以将数据库备份成一个文本文件，该文件中实际包含了多个CREATE和INSERT语句，使用这些语句可以重新创建表和插入数据。MySQLdump备份数据库语句的基本语法格式如下：



user表示用户名称；host表示登录用户的主机名称；password为登录密码；dbname为需要备份的数据库名称；tbname为dbname数据库中需要备份的数据表，可以指定多个需要备份的表；右箭头符号“>”告诉MySQLdump将备份数据表的定义和数据写入备份文件；filename.sql为备份文件的名称。

**本质：**导出的是SQL语句文件

**优点：**无论是什么存储引擎，都可以用mysqldump备份成SQL语句

**缺点：**速度较慢，导入时可能会出现格式不兼容的突发情况，无法直接做增量备份

#### 参数

#### 操作

##### 备份一个数据库

　　mysqldump基本语法：

　　mysqldump -u username -p dbname table1 table2 ...-> BackupName.sql

　　其中：

dbname参数表示数据库的名称；

table1和table2参数表示需要备份的表的名称，为空则整个数据库备份；

BackupName.sql参数表设计备份文件的名称，文件名前面可以加上一个绝对路径。通常将数据库被分成一个后缀名为sql的文件；

　　使用root用户备份test数据库下的person表

mysqldump -u root -p test person > D:\backup.sql

备份文件包含了一些信息，文件开头首先表明了备份文件使用的MySQLdump工具的版本号；然后是备份账户的名称和主机信息，以及备份的数据库的名称，最后是MySQL服务器的版本号，在这里为8.0.13。

备份文件接下来的部分是一些SET语句，这些语句将一些系统变量值赋给用户定义变量，以确保被恢复的数据库的系统变量和原来备份时的变量相同，例如：



该SET语句将当前系统变量character\_set\_client的值赋给用户定义变量@old\_character\_set\_client。其他变量与此类似。

备份文件的最后几行MySQL使用SET语句恢复服务器系统变量原来的值，例如：



该语句将用户定义的变量@old\_character\_set\_client中保存的值赋给实际的系统变量character\_set\_client。

备份文件中以“--”字符开头的行为注释语句；以“/\*!”开头、“\*/”结尾的语句为可执行的MySQL注释，这些语句可以被MySQL执行，但在其他数据库管理系统中将被作为注释忽略，以提高数据库的可移植性。

另外，备份文件开始的一些语句以数字开头，代表的是MySQL版本号，这些语句只有在指定的MySQL版本或者比该版本高的情况下才能执行。例如，40101，表明这些语句只有在MySQL版本号为4.01.01或者更高的条件下才可以被执行。

##### 备份数据库的某个表

MySQLdump还可以备份数据中的某个表，其语法格式为：



tbname表示数据库中的表名，多个表名之间用空格隔开。备份表和备份数据库中所有表的语句中不同的地方在于，要在数据库名称dbname之后指定需要备份的表名称。

##### 备份多个数据库

语法：

mysqldump -u username -p --databases dbname2 dbname2 > Backup.sql

加上了--databases选项，然后后面跟多个数据库

mysqldump -u root -p --databases test mysql > D:\backup.sql

另外，使用--all-databases参数可以备份系统中所有的数据库，语句如下：



使用参数--all-databases时，不需要指定数据库名称。

**提示：**

如果在服务器上进行备份，并且表均为MyISAM表，就应该考虑使用MySQLhotcopy，因为可以更快地进行备份和恢复。

##### 备份所有数据库

mysqldump命令备份所有数据库的语法如下：

mysqldump -u username -p -all-databases > BackupName.sql

#### 总结

### mysqlpump

mysqlpump是一个非常类似于mysqldump的程序，但它带有一些额外的功能。

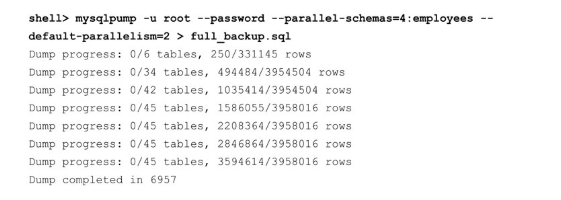
#### 并行处理

可以通过指定线程数量（根据CPU数量）加速备份过程。

例如，使用8个线程进行完整备份：



甚至可以指定每个数据库的线程数。在我们的案例中，employees数据库比company数据库大很多。所以可以为employees数据库指定4个线程，为company数据库指定2个线程：



再来看一个分配线程的示例，其中 3个线程用于 db1和 db2数据库，2 个线程用于db3和db4数据库，还有4个线程用于其他数据库：



你会注意到有一个进度条可以帮助估计时间。

#### 使用正则表达式排除/包含数据库对象

对以prod结尾的所有数据库进行备份：



假设某些数据库中有一些测试表，我们希望将它们从备份中排除。可以使用--exclude-tables选项来指定，该选项将排除所有数据库中名称为test的表：



每个包含和排除选项的值都是适当对象类型以逗号分隔的名称列表。允许在对象名称中使用通配符：

●%匹配零个或多个字符的任何序列。

●\_匹配任何单个字符。

除了数据库和表，还可以包含或排除触发器、例程、事件和用户，例如--include-routines，-include-events和-exclude-triggers。要详细了解包含和排除选项，请参阅https：//dev.mysql.com/doc/refman/8.0/en/mysqlpump.html＃mysqlpump-filtering。

#### 备份用户

在mysqldump中，你不会在CREATE USER或GRANT语句中获得用户的备份；相反，你必须备份mysql.user 表。使用 mysqlpump，可以将用户账户备份为账户管理语句（CREATE USER和GRANT），而不是将用户账户插入mysql系统数据库中：



还可以通过指定--exclude-users选项来排除某些用户：



#### 压缩备份

可以通过压缩备份来减少磁盘空间和网络带宽的占用。可以使用--compress-output=lz4或--compress-output=zlib。

请注意，你需要有相应的解压缩工具：



执行下面的语句进行解压缩：



使用zlib执行此下面的语句：



执行下面的语句进行解压缩：



加速重新加载你会注意到，在输出中，辅助索引从 CREATE TABLE 语句中省略了。这将加速恢复过程。我们将使用ALTER TABLE语句在INSERT结尾处添加这些索引。

#### 加速重新加载

你会注意到，在输出中，辅助索引从 CREATE TABLE 语句中省略了。这将加速恢复过程。我们将使用ALTER TABLE语句在INSERT结尾处添加这些索引。

### mydumper

mydumper是一个类似mysqlpump的逻辑备份工具。

与mysqldump相比，mydumper在以下方面具有优势。

● 并行（因此速度更快）和性能（避免使用复杂的字符集转换例程，因而代码总体上很高效）。

● 一致性。mydumper维护所有线程的快照，提供准确的主库和从库日志位置等。mysqlpump不保证一致性。

● 更易于管理输出（将表和元数据文件分离，并且方便查看/解析数据）。mysqlpump将所有内容写入一个文件，这限制了加载部分数据库对象的选项。

● 使用正则表达式包含和排除数据库对象。

● 有用于终止阻塞备份和所有后续查询的长事务的选项。

mydumper是一款开源的备份工具，需要单独安装。

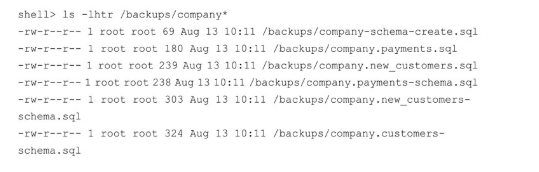
#### 完全备份

以下命令会将所有数据库备份到/backups文件夹中：

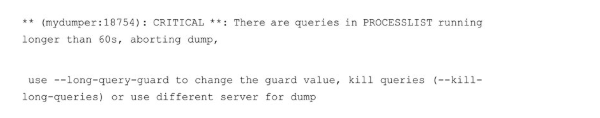


多个文件将在/backups文件夹中被创建。每个数据库的CREATE DATABASE语句均为＜database\_name＞-schema-create.sql，每个表都有自己的schema和数据文件。schema 文件存储为＜database\_name＞.＜table＞-schema.sql，数据文件存储为＜database\_name＞.＜table＞.sql。

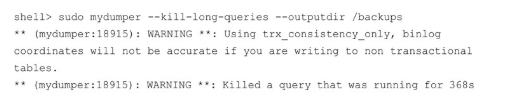
视图存储为＜database\_name＞.＜table＞-schema-view.sql。存储的例程、触发器和事件存储为＜database\_name＞-schema-post.sql（如果目录未创建，则使用sudo mkdir-pv/backups）：



如果有任何查询超过60秒，mydumper将失败并出现以下错误提示：

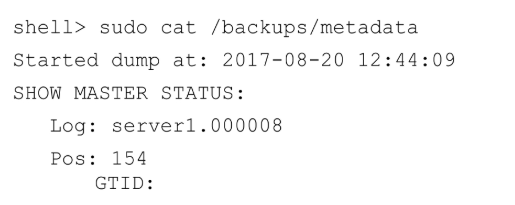


为了避免这种情况，可以使用--kill-long-queries选项，或将--long-query-guard设置为更大的值。--kill-long-queries选项会结束所有长于60秒（或--long-query-guard设置的值）的查询。请注意--kill-long-queries 还会因为一个 bug 而杀死复制线程（https：//bugs.launchpad.net/mydumper/+bug/1713201）：

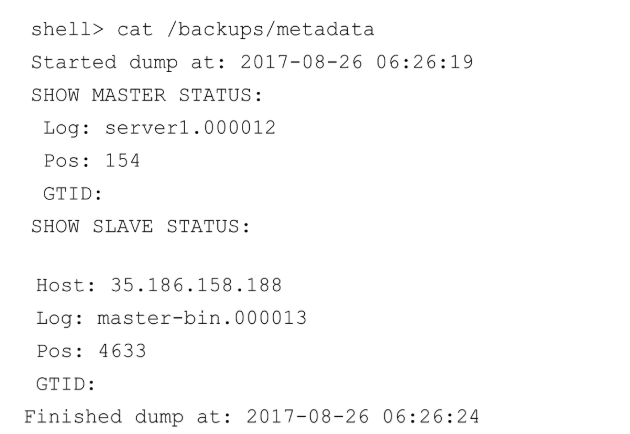


#### 一致的备份

备份目录中的元数据文件包含用于一致备份的二进制日志坐标。在主服务器上，备份目录中的元数据文件会捕获二进制日志位置：

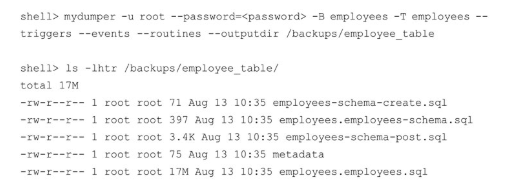


在从服务器上，备份目录中的元数据文件会捕获主服务器和从服务器的二进制日志位置：



#### 备份单独表

以下命令会将employees数据库的employees表备份到/backups目录中：



这些文件的命名规则如下：

● employees-schema-create.sql包含CREATE DATABASE语句。

● employees.employees-schema.sql包含CREATE TABLE语句。

● employees-schema-post.sql包含ROUTINES、TRIGGERS和EVENTS。

● employees.employees.sql包含INSERT语句形式表示的实际数据。

#### 使用正则表达式来备份特定的数据库

可以使用regex选项包含/排除特定数据库。以下命令将从备份中排除mysql和test数据库：



#### 采用mydumper备份大表

为了加速大表的转储和恢复，可以将它分成小块。块的大小可以通过它包含的行数来指定，每个块将被写入一个单独的文件中。

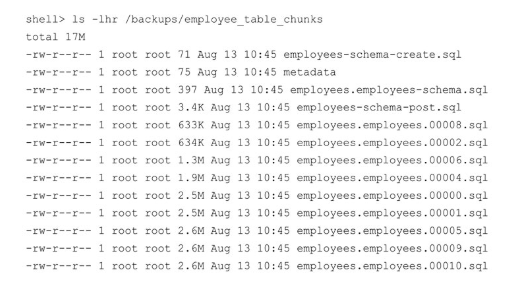


●-t：指定线程的数量。

●--trx-consistency-only：如果只使用事务表，例如InnoDB，那么使用此选项将使锁定最小化。

●--rows：将表分成这些行的块。

对于每一个块，都会创建一个名为database\_name＞.＜table\_name＞.＜number＞.sql的文件，其中数字用零填充，补足5位：



#### 无阻塞备份

为了提供一致的备份，mydumper 通过执行 FLUSH TABLES WITH READ LOCK 来获取GLOBALLOCK。

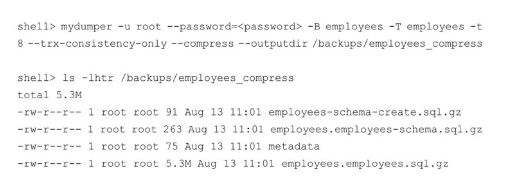
如果有任何长时间运行的事务，使用 FLUSH TABLES WITH READLOCK是多么危险。为了避免这种情况，可以传递--kill-long-queries选项来终止阻塞查询，而不是中止mydumper。

●--trx-consistency-only：相当于mysqldump的--single-transaction，但具有binlog位置。显然，这个位置只适用于事务表。使用这个选项的好处之一是全局读锁只保持在线程的协调过程中，所以一旦事务开始，该选项就会被释放。

●--use-savepoints：减少元数据锁定问题（需要SUPER权限）。

#### 压缩备份

可以指定--compress选项来进行压缩备份：



#### 仅备份数据

可以使用--no-schemas选项来跳过schema并且仅备份数据：



### SELECT INTO OUTFILE

MySQL中，可以使用SELECT...INTO OUTFILE语句将表的内容导出为一个文本文件。其基本的语法格式如下：

SELECT [列名] FROM table [WHERE 语句]

INTO OUTFILE '目标文件' [OPTION];

该语句分为两个部分。前半部分是一个普通的SELECT语句，通过这个SELECT语句来查询所需要的数据；后半部分是导出数据的。其中，“目标文件”参数指出将查询的记录导出到哪个文件中；“OPTION”参数为可选参数选项，其可能的取值有：

FIELDS TERMINATED BY '字符串'：设置字符串为字段之间的分隔符，可以为单个或多个字符。默认值是“\t”。

FIELDS ENCLOSED BY '字符'：设置字符来括住字段的值，只能为单个字符。默认情况下不使用任何符号。

FIELDS OPTIONALLY ENCLOSED BY '字符'：设置字符来括住CHAR、VARCHAR和TEXT等字符型字段。默认情况下不使用任何符号。

FIELDS ESCAPED BY '字符'：设置转义字符，只能为单个字符。默认值为“\”。

LINES STARTING BY '字符串'：设置每行数据开头的字符，可以为单个或多个字符。默认情况下不使用任何字符。

LINES TERMINATED BY '字符串'：设置每行数据结尾的字符，可以为单个或多个字符。默认值是“\n”。

FIELDS和LINES两个子句都是自选的，但是如果两个子句都被指定了，FIELDS必须位于LINES的前面。

### 文件系统快照

## 物理备份

### cp/tar直接复制整个数据库目录

　　MySQL有一种非常简单的备份方法，就是将MySQL中的数据库文件直接复制出来。这是最简单，速度最快的方法。

不过在此之前，要先将服务器停止（冷备），这样才可以保证在复制期间数据库的数据不会发生变化。如果在复制数据库的过程中还有数据写入，就会造成数据不一致。这种情况在开发环境可以，但是在生产环境中很难允许备份服务器。

注意：这种方法不适用于InnoDB存储引擎的表，而对于MyISAM存储引擎的表很方便。同时，还原时MySQL的版本最好相同。

这是一种简单、快速、有效的备份方式。要想保持备份的一致性，备份前需要对相关表执行LOCKTABLES操作，然后对表执行FLUSH TABLES。这样当复制数据库目录中的文件时，允许其他客户继续查询表。需要FLUSH TABLES语句来确保开始备份前将所有激活的索引页写入硬盘。当然，也可以停止MySQL服务再进行备份操作。

这种方法虽然简单，但并不是最好的方法。因为这种方法对InnoDB存储引擎的表不适用。使用这种方法备份的数据最好恢复到相同版本的服务器中，不同的版本可能不兼容。

冷备的优点：

1. 备份简单，只要复制相关文件即可；
2. 备份文件易于在不同操作系统，不同MySQL版本上进行恢复；
3. 恢复相当简单，只需要把文件恢复到指定位置即可；
4. 恢复速度非常快，不需要执行任何SQL语句，也不需要重建索引。

冷备的缺点：

1. InnoDB存储引擎冷备的文件通常比逻辑文件大很多，因为表空间存放着需要其他的数据，如undo段，插入缓冲等信息；
2. 冷备也不总是可以轻易地跨平台，操作系统、MySQL版本、文件大小敏感和浮点数格式都会成为问题。

### rsync

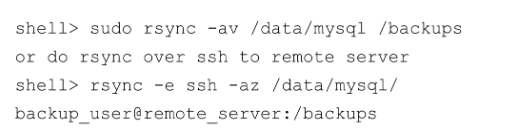
这是一种物理备份方法，可以通过直接复制数据目录中的文件来进行备份。由于在复制文件时写入了新数据，因此备份将不一致并且无法使用。为了避免这种情况，必须先关闭MySQL，复制文件，然后启动MySQL。此方法不适用于每日备份，但非常适合在维护时段进行升级或降级时使用，或者在进行主机交换时使用。

操作步骤：

1. 关闭MySQL服务器：



1. 将文件复制到数据目录中（你的目录可能不同）：



1. 启动MySQL服务器：



### Mysqlhotcopy

MySQLhotcopy是一个Perl脚本，最初由Tim Bunce编写并提供。它使用LOCK TABLES、FLUSHTABLES和cp或scp来快速备份数据库。它是备份数据库或单个表最快的途径，但它只能运行在数据库目录所在的机器上，并且只能备份MyISAM类型的表。MySQLhotcopy在UNIX系统中运行。

　　一看名字就知道是**热备份**。因此，mysqlhotcopy支持不停止MySQL服务器备份。而且，mysqlhotcopy的备份方式比mysqldump快。mysqlhotcopy是一个perl脚本，主要在Linux系统下使用。其**使用LOCK TABLES、FLUSH TABLES和cp来进行快速备份**。

　　原理：先将需要备份的数据库加上一个读锁，然后用FLUSH TABLES将内存中的数据写回到硬盘上的数据库，最后，把需要备份的数据库文件复制到目标目录。

命令格式如下：

[root@localhost ~]# mysqlhotcopy [option] dbname1 dbname2 backupDir/

dbname：数据库名称；

backupDir：备份到哪个文件夹下；

**常用选项：**

--help：查看mysqlhotcopy帮助；

--allowold：如果备份目录下存在相同的备份文件，将旧的备份文件加上\_old；

--keepold：如果备份目录下存在相同的备份文件，不删除旧的备份文件，而是将旧的文件更名；

--flushlog：本次备份之后，将对数据库的更新记录到日志中；

--noindices：只备份数据文件，不备份索引文件；

--user=用户名：用来指定用户名，可以用-u代替；

--password=密码：用来指定密码，可以用-p代替。使用-p时，密码与-p之间没有空格；

--port=端口号：用来指定访问端口，可以用-P代替；

--socket=socket文件：用来指定socket文件，可以用-S代替；

mysqlhotcopy并非mysql自带，需要安装Perl的数据库接口包；下载地址为:http://dev.mysql.com/downloads/dbi.html

提示：

MySQLhotcopy只是将表所在的目录复制到另一个位置，只能用于备份MyISAM和ARCHIVE表。备份InnoDB类型的数据表时会出现错误信息。由于它复制本地格式的文件，因此也不能移植到其他硬件或操作系统下。

目前，该工具也仅仅能够备份MyISAM类型的表。

### ibbackup

ibbackup是innodb公司的一个**热备份**工具，专门对innodb存储引擎进行物理热备份，此工具是收费的，不能免费使用。

### Xtrabackup

#### 概述

xtrabackup是Percona公司CTO Vadim参与开发的一款基于**InnoDB**的**在线热备工具**，具有开源，免费，支持在线热备，备份恢复速度快，占用磁盘空间小等特点，并且支持不同情况下的多种备份形式。它能在不关闭服务器的情况下复制普通文件。但为了避免不一致，它会使用REDO日志文件。XtraBackup被许多公司广泛用作标准备份工具。与逻辑备份工具相比，其优势是备份速度非常快，恢复速度也非常快。

#### 原理

xtrabackup包含两个主要的工具，即xtrabackup和innobackupex，二者区别如下：

1、xtrabackup只能备份innodb和xtradb两种引擎的表，而不能备份myisam引擎的表；

2、innobackupex是一个封装了xtrabackup的Perl脚本，支持同时备份innodb和myisam，但在对myisam备份时需要加一个全局的读锁。还有就是myisam不支持增量备份。

参考：

<https://www.cnblogs.com/gomysql/p/3650645.html>

**XtraBackup工作原理：**

1. XtraBackup 复制 InnoDB 数据文件，这会导致内部不一致的数据，但是它会对文件执行崩溃恢复，以使其再次成为一个一致的可用数据库。
2. 这样做是可行的，因为 InnoDB 维护一个 REDO 日志，也称为事务日志。REDO日志包含了InnoDB数据每次更改的记录。当InnoDB启动时，REDO日志会检查数据文件和事务日志，并执行两个步骤。它将已提交的事务日志条目应用于数据文件，并对任何修改了数据但未提交的事务执行undo操作。
3. Percona XtraBackup会在启动时记住日志序列号（LSN），然后复制数据文件。这需要一些时间来完成，如果文件正在改变，那么它们会在不同的时间点反映数据库的状态。同时，PerconaXtraBackup运行一个后台进程，用于监视事务日志文件，并从中复制更改。Percona XtraBackup需要持续这样做，因为事务日志是以循环方式写入的，并且可以在一段时间后重新使用。PerconaXtraBackup开始执行后，需要复制每次数据文件更改对应的事务日志记录。

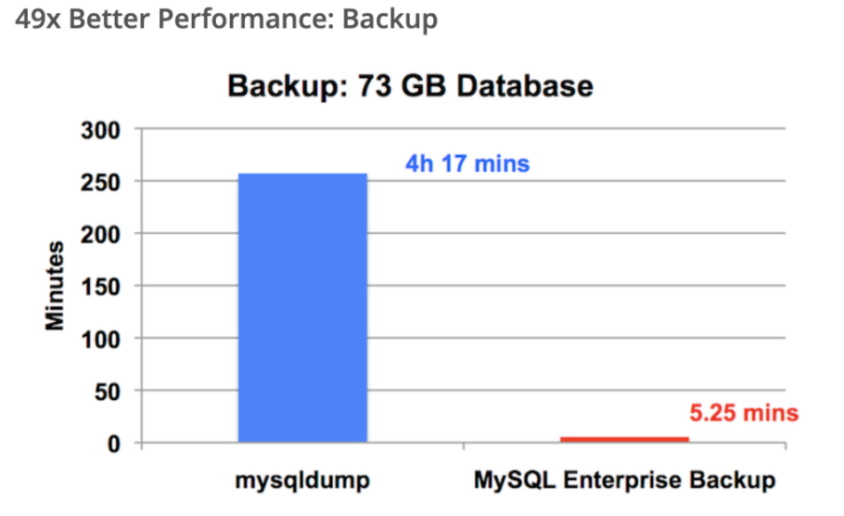
### MEB

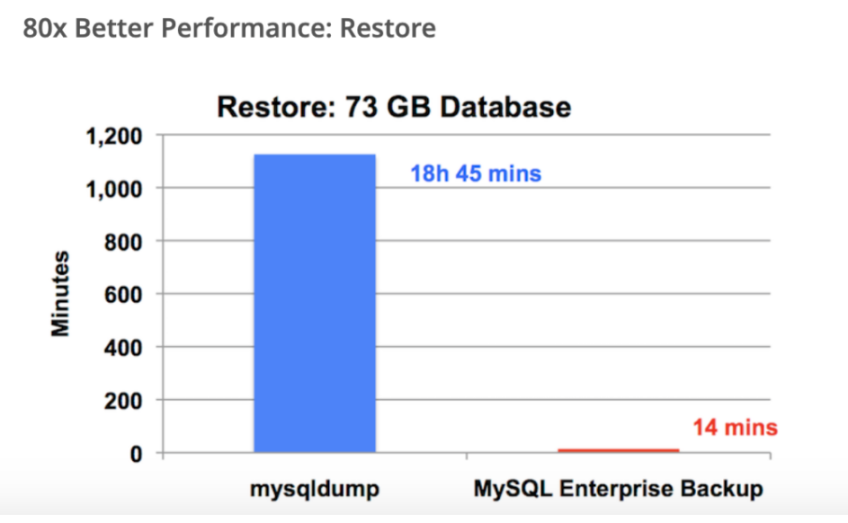
<https://mp.weixin.qq.com/s/3Sh0-fjX-EZ3O9I50TJa_A>

#### 概述

MySQL在其企业版里提供了一款备份工具——MySQL Enterprise Backup，简称MEB。MEB是MySQL商业版中提供的备份工具，属于**物理备份**。

MEB是一款跨平台的高效备份工具，它支持在线“热”备份，增量备份、差异备份、部分备份、压缩备份等一系列主流的备份功能。MySQL Enterprise Backup在优化了InnoDB表备份的同时，还能够备份和恢复MySQL支持的各种存储引擎创建的表。读写过程可以由多个线程独立并行进行，并且，不同的线程可以对单个文件的不同块进行读取、写入处理，使得备份和恢复过程快速执行,**相对于逻辑备份工具mysqldump有着显著的性能提升**。





#### 命令

MEB提供了一个命令行的客户端mysqlbackup，MEB的全部功能都通过它来执行。MEB的功能非常强大，通过它可以进行如下工作：

**备份数据库：**一个完整的备份周期包括，备份、验证和恢复。MEB支持如下类型的备份：

备份到单一文件，支持将其存放到其它服务器、磁带、云存储

完整备份

差异备份或增量备份

压缩备份

部分备份

乐观备份

备份In-memory数据

定期备份

备份时使用validate命令可以对备份结果的完整性进行验证。此外，还可以通过在另一台服务器上恢复备份数据并在新数据目录上运行mysqld来验证备份是否成功。然后可以执行SHOW语句来验证数据库和表结构，并执行查询来验证数据库的进一步细节。

注意：不要将备份目录当做mysql的数据目录直接启动，会引起数据损坏！

**恢复数据库：**MEB可以同如下方法恢复数据库：

恢复压缩备份

恢复加密备份

恢复增量备份

恢复表

恢复使用-tts进行的备份

恢复云存储的备份

恢复表空间到不同的位置

准备/恢复备份目录

**备份加密的InnoDB表空间：**MEB支持对表空间文件加密（TDE）的数据库进行备份和恢复。

**使用Redo日志归档备份：**MEB可以利用Redo日志归档进行备份，以防止Redo日志被覆盖引发的问题。

**主从复制使用MEB：**通过MEB备份主服务器，并在一个新的从服务器上恢复备份来搭建主从复制，而不需要停止主服务器。

**群组复制使用MEB：**与备份单机相同，MEB也可以用于组成员的快速备份和恢复。

**备份加密：**MEB支持使用AES区块加密（CBC模式）对备份进行加密，以增强安全性。

**媒体管理软件使用MEB：**MEB支持使用媒体管理软件将备份保存至大容量的存储，例如可以使用支持System Backup to Tape (SBT) API的软件将数据备份到磁带。

**容器使用MEB：**使用企业版MySQL容器，可以对在同一个主机上的其他容器中的mysql进行备份。

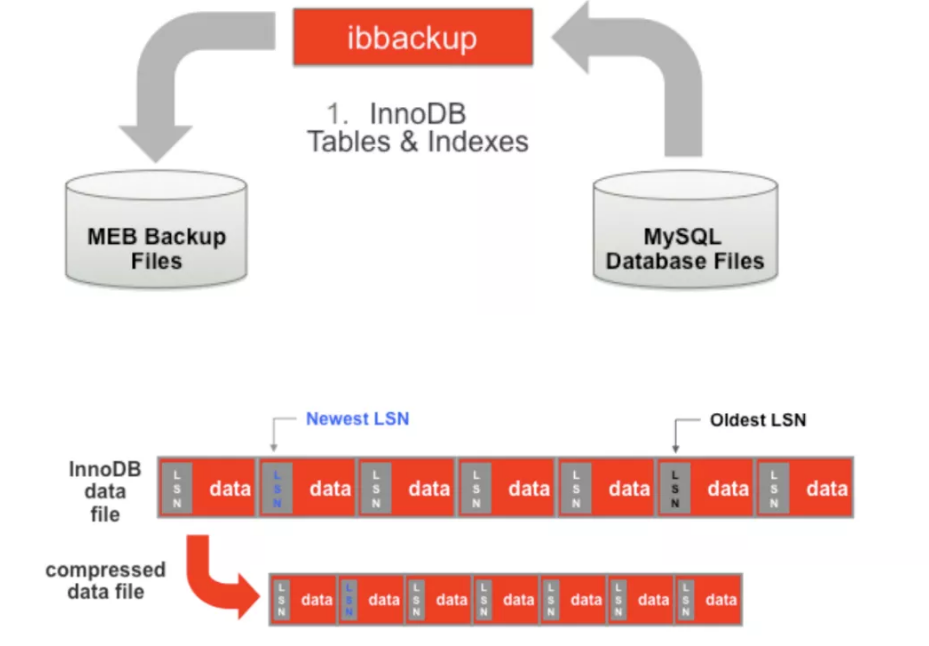
#### 原理

同XtraBackup一样，mysqlbackup的使用过程同样包含如下三个步骤：备份（--backup）->应用日志（--apply-log）->恢复（--copy-back）。

**工作过程：**

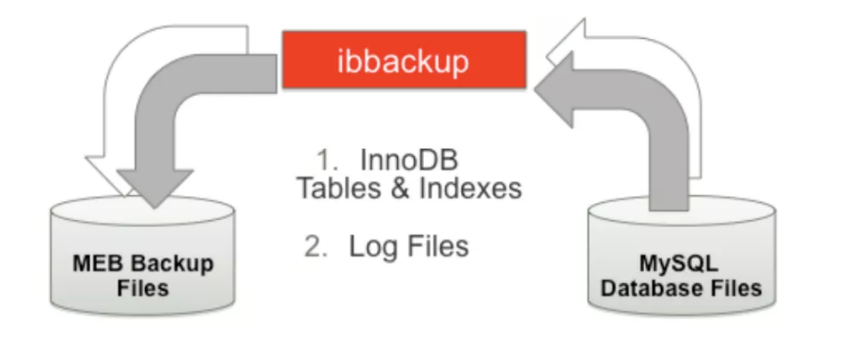
1、备份InnoDB表。

备份首先通过ibbackup将InnoDB的数据文件进行压缩和复制。文件通过压缩可以达到70%~90%，并且会标记最早和最新的LSN。

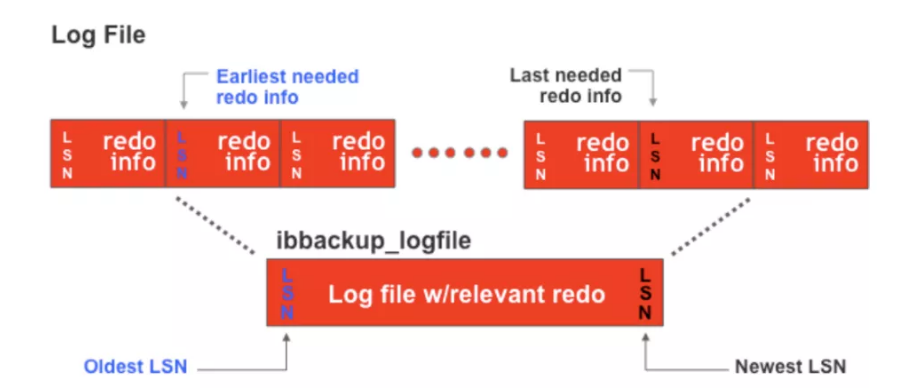


2、备份InnoDB日志。

在复制数据文件期间积累的InnoDB日志文件，使用LSN进行复制。

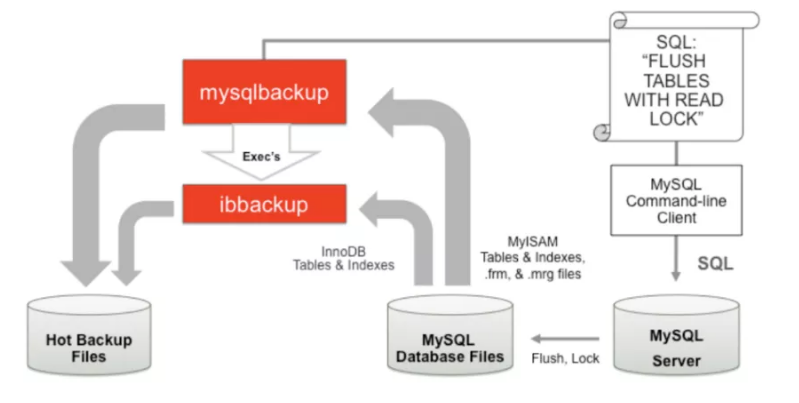


只复制日志文件中包含重做信息的部分，并覆盖从备份开始到备份结束的时间。通过这个操作实现一致性。



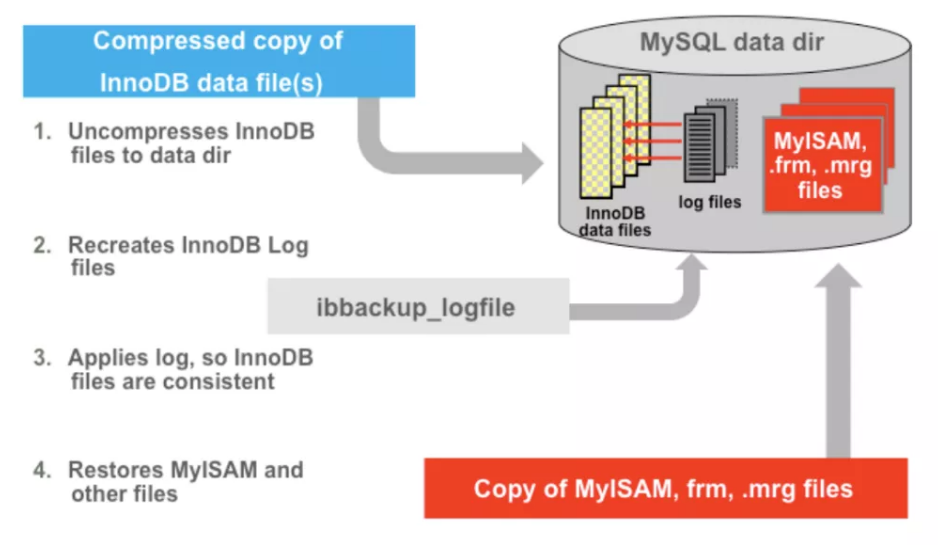
3、备份MyISAM表。

mysqlbackup相当于封装了ibbackup。更容易使用的接口去备份MyISAM数据和InnoDB数据，因此，它可以进行一致性的恢复。在备份期间完全可以访问InnoDB表，但是MyISAM表会有一个表锁，无法进行更新。



4、恢复数据库

恢复数据库首先将InnoDB文件解压缩到data目录。MySQL Enterprise Backup重新创建InnoDB日志文件，并应用这些日志，以使InnoDB文件恢复到一致的状态，然后恢复MyISAM数据。



介绍一下MEB相对于PXB的优势在哪里。

1、MEB可以显示备份的过程

2、MEB可以通过Oracle secure backup将数据备份到磁带

3、MEB支持离线备份

4、MEB支持备份日志文件用于时间点恢复

5、MEB可以在恢复时更改表名称

### 锁定实例

从MySQL 8开始，我们可以锁定实例进行备份了，这将允许在线备份期间的DML，并阻止可能导致快照不一致的所有操作。

**操作步骤：**

1. 在开始备份之前，请锁定需要备份的实例：



1. 执行备份，完成后解锁实例：

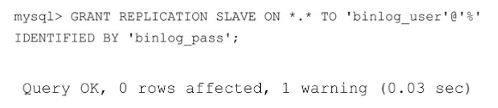


## 二进制日志备份

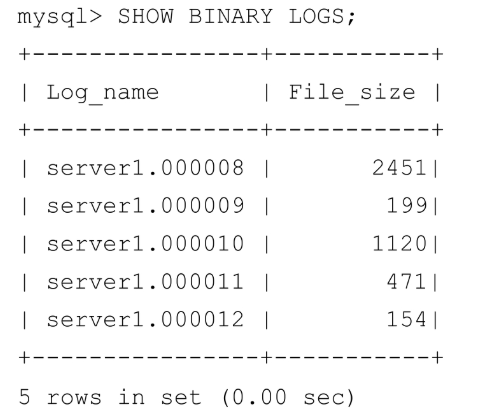
备份二进制日志：该进程将二进制日志从数据库服务器流式传输到远程备份服务器。既可以从从服务器也可以从主服务器进行二进制日志备份。如果你正在从主服务器进行二进制日志备份，并在从服务器进行实际备份，则应使用--dump-slave获取相应的主日志位置。如果你使用的是mydumper或XtraBackup，则主和从二进制日志位置会被同时提供。

**操作步骤：**

1. 在服务器上创建一个复制用户，并设置一个强密码：

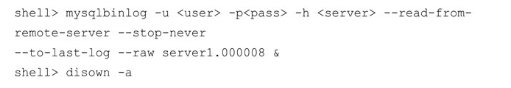


1. 检查服务器上的二进制日志：

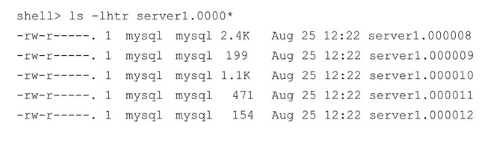


你可以在服务器上找到第一个可用的二进制日志，可以从这里开始备份。在这个例子中，它是server1.000008。

1. 登录到备份服务器并执行以下命令，会将二进制日志从MySQL服务器复制到备份服务器。你可以使用nohup或disown：



1. 验证是否正在备份二进制日志：



## 备份工具

常用的几种备份工具：

1. 社区版备份工具：mysqldumper、mysqlhotcopy
2. 企业版安装包中的备份工具：mysqlbackup
3. 第三方备份工具：xtrabackup和innnobackupex（物理备份）

### cp/tar

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。

### mysqlhotcopy

mysqlhotcopy：物理备份工具，名不副实的的一个工具, 几乎冷备, 仅支持MyISAM存储引擎。

1. 企业版和社区版都包含
2. Perl写的脚本，本质上是使用锁表语句后再拷贝数据
3. 只支持MyISAM数据引擎

### mysqlbackup

1. 在线备份
2. 增量备份
3. 部分备份
4. 在某个特定时间的一致性状态的备份

### mysqldump

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。

1. 企业版和社区办都包含
2. 本质上使用SQL语句描述数据库以及数据并导出
3. 在MyISAM引擎上锁表，InnoDB引擎上锁行
4. 数据量很大时不推荐使用

### lvm2 snapshot

lvm2 snapshot：几乎热备, 借助文件系统管理工具进行备份。Logical VolumeManager (LVM)提供了对任意一个LogicalVolume(LV)做“快照”(snapshot)的功能，以此来获得一个分区的状态一致性备份。

### xtrabackup

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。

1. xtrabackup是一个对InnoDB做数据备份的工具，支持在线热备份（备份时不影响数据读写），是商业备份工具InnoDB Hotbackup的一个很好的替代品
2. Xtrabackup有两个主要的工具：xtrabackup、innobackupex
3. xtrabackup只能备份InnoDB和XtraDB两种数据表，不能备份MyISAM类型的表
4. Innobackupex是将xtrabackup进行封装的perl脚本，所以能同时备份处理innodb和myisam存储引擎，但在处理myisam时需要加一个读锁。

# 数据库恢复

## 背景

根据我以往的一些经验来说，通常需要从备份恢复数据的场景有如下几种：

1. 被误删库了
2. 被误删表了，类型为TRUNCATE或者DROP
3. 被误删列了，类型为ALTER ... DROP COLUMN
4. 被误删数据了，类型为DELETE或者UPDATE或者REPLACE
5. 表空间损坏或出现坏块了

根据场景来说，我们可以大致分为两类：

第一类为不可逆恢复，也就是通常的DDL，比如上述的1、2、3、5等场景

第二类为可逆的恢复，通常可以利用binlog进行回滚（要求binlog格式为ROW，binlog\_image为FULL），也就是对应上述的场景4

对于第二类的恢复需求一般来说都比较容易处理，可以利用binlog回滚工具，例如业界比较著名的有binlog2sql以及MyFlash等，这里暂不赘述，我们重点来讨论第一类需求。

为了达到快速恢复的目的，业界DBA经常会采用的方式就是部署一个延迟从库来解决，我们公司目前 所有的核心DB都部署了延迟从库。但是即便有了延迟从库，假设我们错过了延迟的时间，或者在后续利用延迟从库恢复的时候指定错了位点，导致了误删DDL同样应用到了从库，这个时候我们就没有办法利用延迟从库这根救命稻草了。

## 分类

### 全备恢复（异机恢复）

此时，我们只能通过备份来进行数据恢复了。首先我们需要恢复全备，通常来说就是xtrabackup备份的物理备份了。假设你的备份在远程的机器上，那么你可能需要做如下几步动作来进行全备恢复：

1. 将备份scp或者rsync到目标实例机器上
2. 假设备份文件是压缩的情况下，需要解压
3. 解压完成后，需要apply redo log
4. 更改文件权限
5. 假设你直接将文件拷贝到的目标实例的datadir目录下，那么这一步你就可以直接启动mysqld，假设不是，那么你还需要将数据文件move-back或者copy-back到目标实例的datadir
6. 实例启动

### 增备恢复

到这里，全备已经恢复完成了，接下来需要做的就是增量恢复了。按照我们之前的备份方案，我们需要通过binlog来完成增量数据的恢复。对于binlog恢复，我们通常需要以下几个步骤

1. 确定全备对应的binlog位点，也就是需要恢复的起始点
2. 解析主库的binlog，确定误删数据的位点，作为我们恢复的终点
3. 利用mysqlbinlog —start-position —stop-position+管道的方式，将binlog恢复到目标实例上

binlog恢复的方式有很多种，你可以用的是原先master上的binlog，也可以用binlogserver上的binlog，需要做的就是找到binlog恢复的终点即可。

### 增备恢复优化

到这里，你可能会觉得，利用binlog恢复有点麻烦。确实是这样的，利用mysqlbinlog命令并没有办法指定恢复到哪个GTID，只能通过解析binlog，找到需要恢复到的GTID对应的pos位点才行，这对于自动化来说实现起来会比较麻烦。另外，如果利用mysqlbinlog命令恢复，属于单线程恢复，假设需要恢复的binlog量比较多的话，那么这个增量恢复的时间可想而知。

那么有什么办法能加速binlog应用呢？这里我们就想到了MySQL5.7的并行复制，如果我们能用到sql thread的并行复制，是不是这个问题就解决了呢？

### master上binlog恢复

我们回到全备恢复的位点，我们将新实例作为原先的master的slave，然后恢复到指定的GTID位置就可以了呢？没错，这是一种非常简便又轻松还不容易出错的方式，并且还可以利用并行复制的原理来加速binlog应用的目的。但是这种方式的一个要求就是原先的master最老的binlog包含了我们需要的起始恢复位点，这个很容易想到，所以，这将成为我们首选的恢复方式。

### binlogserver上binlog恢复

假设原先master上的binlog已经被purge了，那么我们那需要从binlog上去恢复。有人可能会想到将binlogserver上的binlog拷贝到原先的master上，然后通过修改binlog index来达到注册的目的，实际上这并不可取，具体原因可以见《手动注册binlog文件造成主从异常》。

我们可以采取的方式是什么呢？就是利用binlogserver做成伪装master，然后将从库change上去，其思想就是欺骗slave，让slave的io\_thread将缺失的binlog拉取过来，sql\_thread并行应用binlog event（我们将在下一节具体演示这种方式）。

**优化后的恢复流程**

经过优化以后，我们的增备恢复流程就变成了，首先通过master上的binlog进行恢复，如果发现master上的binlog已经被purge了，那么通过binlogserver上的binlog进行恢复，这样一来我认为是比较科学合理的恢复流程。

各种恢复方式时效性对比



### 业务恢复

到这里，我们已经完成了全量+增量的备份数据恢复，这个时候需要同研发确认数据，确认完成以后将对应的表恢复到原先的master，通常采用的方式有：

mysqldump导出+导入目标实例

表空间传输

## 思路

1. 利用全备的sql文件中记录的CHANGE MASTER语句，binlog文件及其位置点信息，找出binlog文件中增量的那部分。
2. 用mysqlbinlog命令将上述的binlog文件导出为sql文件，并剔除其中的drop语句。
3. 通过全备文件和增量binlog文件的导出sql文件，就可以恢复到完整的数据。

MySQL常见的有三种备份恢复方式：

1、利用Mysqldump+二进制日志实现备份

2、利用LVM快照+二进制日志实现备份

3、使用Xtrabackup备份

## 逻辑恢复

### mysqldump

还原mysqldump命令备份的数据库，语法如下：

　　mysql -u root -p [dbname] < backup.sq

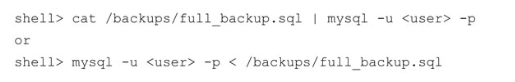
　　注：这种是采用加载SQL语句的方式实现恢复。

示例：

mysql -u root -p < C:\backup.sql

**操作步骤：**

登录备份所在的服务器：



要在远程服务器上恢复，可以使用-h ＜主机名＞选项：



当恢复一个备份时，该备份的语句将被记录到二进制日志中，这可能会拖慢恢复过程。如果不希望恢复过程被写入二进制日志，则可以使用SET SQL\_LOG\_BIN=0；选项在session（会话）级别关闭这个功能：



或者使用：



**建议：**

1. 由于备份恢复需要很长时间，因此建议在screen会话内启动恢复过程，即使断开与服务器的连接，备份的恢复也会继续。

注：可以使用nohup sh \*\*.sh在后台执行备份恢复任务。

1. 有时候，在恢复期间可能会出现故障。如果将--force选项传递给MySQL，恢复过程将继续：



### myloader

myloader是多线程恢复mydumper备份集的工具。myloader与mydumper是一起的，不需要单独安装。

myloader的常用选项有：要连接的MySQL服务器的主机名（默认值为localhost）、用户名、密码和端口。

#### 恢复完整的数据库



各选项的含义如下所述：

●--overwrite-tables：这个选项会删除已经存在的表。

●--compress-protocol：该选项在MySQL连接上使用压缩。

●--threads：该选项指定要使用的线程数量，默认值是4。

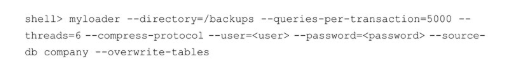
●--queries-per-transaction：指定每个事务的查询数量，默认值是1000。

●--directory：指定要导入的转储目录。

#### 恢复单个数据库

可以指定--source-db ＜db\_name＞，仅恢复单个数据库。

假设你想恢复company数据库：



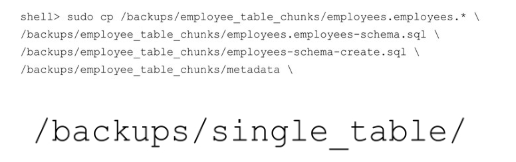
#### 恢复单个表

mydumper将每个表的备份写入单独的.sql文件。你可以选择这个.sql文件并恢复：



如果这个表被拆分为chunk（块），则可以将与这个表相关的所有chunk和信息复制到一个目录并指定其位置。

复制所需的文件：



使用myloader加载，它会自动检测chunk并加载它们：



### mysqladmin

mysqladmin -uroot -p create db\_name

mysql -uroot -p db\_name < /backup/mysqldump/db\_name.db

注：在导入备份数据库前，db\_name如果没有，是需要创建的； 而且与db\_name.db中数据库名是一样的才可以导入。

### source

soure方法：

mysql > use db\_name

mysql > source /backup/mysqldump/db\_name.db

### mysqlimport

### LOAD DATA INFILE

如果是通过SELECT INTO OUTFILE导出的符号分割文件，可以使用LOAD DATA INFILE通过相同的参数来加载。也可以使用mysqlimport，这是LOAD DATA INFILE的一个包装。

### alter table xxx import tablespace

将需要备份的数据库表文件.frm和.ibd文件拷贝到对应的目录下，在当前的数据库上执行建表语句，创建一个与待备份表一样的空表。

然后执行释放表空间语句：

alter table xxx drop tablespace; //此时删除data目录下该表对应的ibd文件

然后将对应的ibd文件拷贝到这个目录，接着执行：

alter table xxx import tablespace;

### 基于时间点恢复

一旦恢复完整的备份后，你需要恢复二进制日志以获得时间点（point-in-time）恢复。备份集提供截止到备份可用时的二进制日志坐标。

应该根据 mysqldump 中指定的--dump-slave 或--master-data选项从备份所在的服务器中选择二进制日志备份。

**操作步骤：**

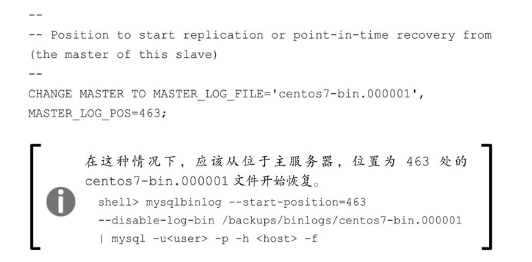
#### mysqldump或mysqlpump

根据你传给 mysqldump/mysqlpump 的选项，二进制日志信息被作为 CHANGE MASTER TO命令存储在SQL文件中。

1. 如果你用了--master-data，则应使用从服务器的二进制日志：

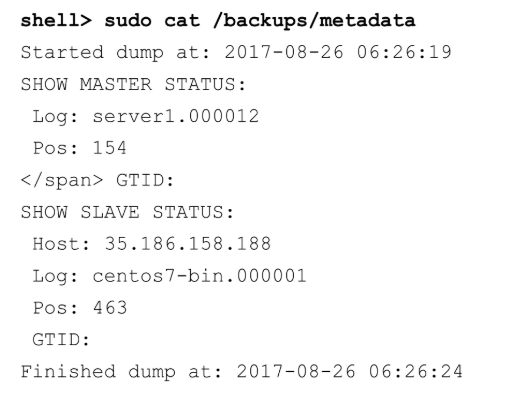


1. 如果你用了--dump-slave，则应该使用主服务器上的二进制日志：



#### mydumper

二进制日志信息可以从元数据中获取：



如果你已经从从服务器中获取二进制日志备份，则应从位置为 154（SHOW MASTER STATUS）的server1.000012文件开始恢复：



如果你有来自主服务器的二进制日志备份，则应从位置为463（SHOW SLAVE STATUS）的centos7-bin.000001文件开始恢复：



### 基于位置恢复

即binlog。

## 物理恢复

### 还原直接复制目录的备份

通过这种方式还原时，必须保证两个MySQL数据库的版本号是相同的。MyISAM类型的表有效，对于InnoDB类型的表不可用，InnoDB表的表空间不能直接复制。

### ibbackup

### Xtrabackup

### MEB

# 分布式数据库实践

## 数据备份

### 备份数据

分布式数据库业务数据包括四类备份：

1. data数据备份
2. binlog日志备份
3. 活跃GTID备份
4. 元数据备份（分布式特有）

在恢复操作时，需要同时用到这四类备份的结果文件。

#### data数据文件

data数据文件备份分为全量备份和增量备份。

##### 全量备份

备份恢复操作中必用到的文件。

操作步骤：

1. 开启xtrabackup备份任务，记录xtrabackup\_log
2. 拷贝所有表的\*.ibd数据文件和ibdata1文件到指定备份目录
3. 上述完成拷贝后，执行如下命令：

FLUSH TABLE WITH READ LOCK;

1. 拷贝.frm、.MYD、.MYI、misc files
2. 记录拷贝完表文件时的重做日志的检查点LSN
3. 执行命令，解除步骤3中的锁：

UNLOCK TABLES;

1. 结束xtrabackup备份任务，拷贝xtrabackup\_log，将所有备份文件压缩在back\_FULL\_$back\_start\_time.xbstream文件内。

##### 增量备份

恢复操作中指定恢复的时刻如果在增量备份时刻之后，建议选择全量备份文件+增量备份文件一起做恢复操作。

操作步骤：

1、开启xtrabackup备份任务，记录xtrabackup\_log

2、拷贝所有表的\*.ibd数据文件距离上一次全量备份产生变化的数据和ibdata1文件距离上一次全量备份产生变化的数据到指定备份目录，文件后缀名.delta和.meta

3、上述完成拷贝后，执行如下命令：

FLUSH TABLE WITH READ LOCK;

4、拷贝.frm、.MYD、.MYI、misc files

5、记录拷贝完表文件时的重做日志的检查点LSN

6、执行命令，解除步骤3中的锁：

UNLOCK TABLES;

7、结束xtrabackup备份任务，拷贝xtrabackup\_log，将所有备份文件压缩在back\_INCREMENTAL\_$back\_start\_time.xbstream文件内。

#### binlog日志

备份DB节点$HOME/data/binlog目录下的binlog二进制文件，恢复操作中用于数据一致性处理。

#### 活跃事务列表和binlog位置

备份集群活跃事务列表及对应的各个主DB当前binlog位置。

1. 活跃事务列表：可用于保证全局节点恢复数据一致性
2. binlog：在恢复任意时刻时，定位binlog备份文件位置。在之前恢复过程中用到（最新恢复已经不要这个信息，直接使用GTID）。

#### 集群元数据

备份集群相关的元数据，主要包括数据字典，用户密码，索引信息。

### 备份策略

#### data数据

**定时备份**

分布式数据库集群内部每个MySQL节点的默认备份策略为：

1. 周日备份全量数据，其他时间分别在周日全量备份的基础上做一次增量备份。如此，可通过全量数据和其他任一增量备份数据，快速恢复到指定一天的数据
2. 也可以根据业务要求，调整定时备份策略，每天的备份模式可以在“不备份”、“全量备份”、“增量备份”中三选一

**实时备份**

用户也可以在操作界面手动备份。手动备份可以满足某些特定场景下的实时备份需要。

#### binlog日志

每个DB节点定时备份binlog，默认备份间隔时间2小时。

注：在配置项中可以设置（单位：s）。

#### 活跃事务列表和binlog位置

若活跃事务列表和binlog位置备份开关打开，则：

1. 每隔30s向GTM查询当前活跃事务列表信息和向db代理进程查询当前binlog位置信息，并将查询结果信息保存在.current文件中
2. 每隔60s查看.current文件是否达到1M，如果达到1M，将.current文件归档到备份目录Active\_TX\_Archive下

注：在配置文件中可以设置事务开关（ON/OFF）。

#### 集群元数据

### 备份目录结构

### 备份工具

#### Xtrabackup

数据备份。

#### Mysqldump

元数据备份。

### 备份原理

#### 流程

##### data数据

###### 定时备份流程

1. 操作界面创建定时备份任务下发到MDS库，MDS将备份任务入库（RDB），对应的表为mds.timing\_backup\_task\_info
2. MDS将备份任务（全量、增量、是否强制备份主）下发到CM
3. CM收到实时备份任务执行

3.1 首先检查该cluster是否备份

3.2 在确认备份任务信息（如备份备机需要确认该group内是否存在OK的备机，如不存在则该group备份主机，备机的选择按照(机房,ip,port)的顺序选择）

3.3 CM设置备份类型（全量/增量，如果是增量备份，去备份结果audit文件里找之前是否有全量备份的结果文件(根据IP+port寻找)，没有的话，按照全量备份去做备份，有的话，找到LSN）

3.4 CM向GTM申请活跃事务列表，并设置超时计时器（获取该集群的GTID(最大GTID和活跃GTID以及产生的时间)请求）

1. GTM向CM发送当前活跃事务列表
2. CM接收到活跃事务列表后向主DB的DBAgent获取binlog position
3. DBAgent将binlog pos上传到CM中
4. CM将备份任务发给DBAgent（备份请求包含备份类型、备份开始时间、GTID(GTM)、GTID\_SET(主DB)），CM设置备份超时计时器（超时自动认为失败）
5. DBAgent收到CM发出的备份请求执行备份

8.1 收到CM的备份请求后，将受到的GTID(GTM)、主GTID\_SET(DB)信息写入xtrabackup\_active\_tx\_info文件

8.2 调用innobackupex\_ddb.sh脚本去执行备份

8.3 获取备份文件目录及临时目录，确定是否需要加密和压缩

8.4 查询、创建、清空备份临时目录，将DBAgent生成的xtrabackup\_active\_tx\_info文件放入daradir/performance\_schema/xtrabackup\_active\_tx\_info.frm

8.5 调用innobackupex工具备份

###### 实时备份流程

1. 操作界面将实时备份请求发给MDS
2. MDS将实时备份任务发送给CM
3. CM接收到实时备份请求后，向各个节点DBAgent发起备份请求
4. DBAgent接收到备份请求后，将备份文件保存在本地$HOME/backup\_root
5. DBAgent备份完成后，将备份完成结果情况反馈给CM
6. CM接收到各个节点DBAgent的备份结果，汇总后给MDS
7. MDS接收到备份结果后，入库，相应的表是：

gdb\_omm.gdb\_cluster\_backup\_history

gdb\_omm.gdb\_cluster\_backup\_history\_detail

###### 备份目录

备份文件存放路径可以在配置文件中设置，会自动生成子目录“DBCluster\_集群ID/Node\_groupid\_ip/DATA\_BACKUP”。

##### binlog日志

###### 原理

Binlog日志通过DBAgent定时任务完成

默认备份间隔时刻2小时

DBAgent将当前备份位置记录在etc/dbagent\_info/binlogbackup.info中，作为下一个备份开启位置。

###### 备份目录

可以在配置文件中设置备份目录，会自动生成子目录“DBCluster\_集群ID\_Node\_groupid\_ip/BINLOG\_BACKUP”，备份的binlog二进制文件就存放在这个子目录下。

##### 活跃事务列表和binlog位置

###### 原理

CM每隔30s向GTM和DBAgent获取最新的活跃事务列表信息和当前binlog位置信息，并将获取的信息保存在clustermanager.ini中的backup\_root\_directory配置路径下的.current文件中。

###### 备份目录

可以在配置文件中设置备份目录，有定时任务每隔60s检查一次配置路径下的.current文件是否达到1M，如果达到1M，归档到配置路径下的子目录“Active\_TX\_Archive”下。

##### 集群元数据

###### 原理

CM（Cluster Manager）向MDS（MetaDataServer）发送元数据备份时，会全量备份该集群下的数据字典、密码信息和索引信息，在每次该集群有DDL操作时，会增量备份数据字典和索引信息。

RDB备份时单机数据库备份，没有一致性问题，其命令为：

mysqldump -u${user} -p${passwd} -P${port} -h${ip} --database gdb\_omm in alarm measure sys

注：元数据信息并不大，所以可以采用mysqldump，对于数据量很大的库操作非常慢。更重要的是这个是单机数据库，不需要上述的xtrabackup（适用于集群）。

###### 备份目录

备份的目录可以在配置文件配置，会自动生成子目录“MetaData/集群ID”，备份的数据文件都存放在这个子目录下。

#### 系统影响

备份过程对系统影响如下：

1. 备份过程中消耗数据文件所在磁盘的IO，需要从本地拷贝数据文件
2. 备份文件需要从本地上传到NFS共享目录，影响DB服务器的网络IO，建议使用单独的网卡配置NFS
3. 备份过程中会有短暂的flush table write read lock锁（秒级别的锁）

综上所述，建议备份操作与批业务操作放在不同的时间段执行。

## 数据恢复

### 恢复分类

恢复可以指定恢复到任意时刻，根据恢复时使用到的文件情况分为：

#### 全量恢复

仅仅用到全量备份文件的恢复。

#### 增量恢复

既用到了全量备份文件，又用到了备份文件的恢复。

建议恢复操作时，指定恢复时刻为备份结束时刻。

对于集群，根据备份源可以分为：

本地系统内恢复：备份和恢复的集群在同一套系统内

非本系统恢复：备份和恢复的集群属于不同的系统

### 恢复原理

#### 单DB恢复

恢复流程如下：

1. 获取恢复所需要的文件
2. 通过全量备份文件（及增量备份文件）恢复DB
3. 应用binlog文件
4. 回滚恢复时刻的活跃事务

注：单DB没有元数据信息，所以不需要恢复元数据。

#### 集群恢复

恢复流程如下：

1. 对集群的元数据进行恢复
2. 恢复集群中每个Group的DB

2.1 获取恢复所需要的文件

2.2 通过全量备份文件（及增量备份文件）恢复主机DB

2.3 对主机DB应用binlog文件

2.4 对主机DB回滚恢复时刻的活跃事务

2.5 对主机DB打tar包

2.6 将主机生成的tar包解压到每个备机DB

1. 设定GTM的最大GTID值

#### 流程

##### 数据恢复

1. 操作界面下发恢复任务给MDS

MDS将恢复任务下发给CM的信息包含：

1. 备份结果文件
2. 备份结果文件里的DB
3. 集群ID
4. 恢复的时间
5. 恢复的DB
6. CM在接收到恢复任务后执行以下操作：

2.1 条件检查

CM接收到任务后执行条件检查及准备：

1. 从backupResultFileName文件中提取如下信息：

备份类型（全量/增量）

全量备份文件路径

增量备份文件路径

1. 检查backupdb/restoredb是否存在
2. 检查集群状态，以下状态下不能执行恢复：

luster不存在

roup不存在

group主备切换中

restoredb备份中

1. 如果restoredb是master且满足下列条件之一，不能执行恢复：

所属组有slave

所属组正在执行task

所属组正在执行load

1. 把xxx.current活跃事务列表改名放入CM\_xx文件TODO
2. 如果resotredb是master，则通知PM停服
3. 查看DB的备份binlog时间，是否大于恢复的时间：否则向DBAgent发送备份binlog的时间到恢复时间的请求，CM收到成功响应后悔进行下一步

2.2 执行恢复

上述检查通过后，CM通知DBAgent开始执行恢复操作。

1. DBAgent接收到恢复请求：

3.1

3.2

3.3

3.4

3.5 调用restore.py第五阶段，执行已提交事务回滚

根据活跃事务

##### 元数据恢复

###### 恢复主机

若恢复的是主机，则执行如下命令：

1. 首先到备机上，登录ommdb，删除主备关系：

su - omm

cd /home/omm/bin

mysql -uroot -p’’ -h备机IP -P备机Port --default-character-set=utf8

mysql>stop slave;

mysql>reset slave all;

1. 再连接主机，恢复主机数据：

su - omm

cd /home/omm/bin

mysql -uroot -p’’ -h主机IP -P主机Port --default-character-set=utf8

注：在backup\_db\_\*\*\*\*.sql一般第30行左右有DEFAULT CHARACTER SET utf8\_bin，即字符集为utf8，如果不是需要设置。

mysql>reset master;

mysql>source backup\_db\_\*\*\*\*.sql;

1. 完成恢复后利用show variable like ‘%semi%master%’;查看半同步状态是否正确
2. 此时主备切换，以前的主机变成备机，以前的备机变成主机，在原主机上连接mysql，建立主备关系：

mysql>stop slave;

mysql>change master to master\_host=’原备机IP’,master\_user=’repl’,master\_password=’’,master\_port=,master\_auto\_position=1;

mysql>start slave;

5、show varibales like ‘%semi%slave%’;查看半同步状态

###### 恢复备机

###### 同时恢复主备机