**XtraBackup**

[XtraBackup](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231212)

[2 安装XtraBackup](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231213)

[2.1 安装XtraBackup binary版本](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231214)

[2.1.1 yum的安装方法：](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231215)

[2.1.2 直接下载rpm包安装](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231216)

[3 XtraBackup使用手册](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231217)

[3.1 使用innobackupex脚本](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231218)

[3.1.1 备份预备工作](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231219)

[3.1.2 全备和全备还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231220)

[3.1.2.1 使用innobackupex创建全备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231221)

[3.1.2.2 使用innobackupex预备全备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231222)

[3.1.2.3 使用innobackupex还原备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231223)

[3.1.3 增量备份和还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231224)

[3.1.3.1 创建增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231225)

[3.1.3.2 预备增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231226)

[3.1.3.3 还原增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231227)

[3.1.3.4 可以使用流做增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231228)

[3.1.4 部分备份和还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231229)

[3.1.4.1 创建部分备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231230)

[3.1.4.2 预备部分备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231231)

[3.1.4.3 还原部分备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231232)

[3.1.5 窄备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231233)

[3.1.5.1 创建窄备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231234)

[3.1.5.2 预备窄备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231235)

[3.1.5.3 还原窄备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231236)

[3.1.6 备份加密](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231237)

[3.1.7 其他功能](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231238)

[3.1.7.1 备份压缩和流](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231239)

[3.1.7.2 在复制环境下备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231240)

[3.1.7.3 加速备份进程](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231241)

[3.1.7.4 节流(throttling)备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231242)

[3.1.7.5 还原独立表](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231243)

[3.1.7.6 时间点还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231244)

[3.1.7.7 提高对FLUSH TABLES WITH READ LOCK控制](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231245)

[3.1.8 innobackupex工作原理](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231246)

[3.1.9 Reference](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231247)

[3.2 使用Xtrabackup](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231248)

[3.2.1 选择bianry](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231249)

[3.2.2 配置Xtrabackup](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231250)

[3.2.3 创建全备和还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231251)

[3.2.3.1 创建全备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231252)

[3.2.3.2 预备全备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231253)

[3.2.3.3 还原全备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231254)

[3.2.4 增量备份和还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231255)

[3.2.4.1 增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231256)

[3.2.4.2 预备增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231257)

[3.2.5 使用归档日志做增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231258)

[3.2.5.1 原理](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231259)

[3.2.5.2 创建备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231260)

[3.2.5.3 使用归档日志来预备备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231261)

[3.2.6 部分备份和预备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231262)

[3.2.6.1 备份部分备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231263)

[3.2.6.2 预备备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231264)

[3.2.7 窄备份和预备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231265)

[3.2.7.1 创建窄备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231266)

[3.2.7.2 预备窄备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231267)

[3.2.7.3 备份还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231268)

[3.2.8 其他功能](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231269)

[3.2.8.1 节流备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231270)

[3.2.8.2 使用脚本调用xtrabackup来执行备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231271)

[3.2.8.3 分析表统计信息](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231272)

[3.2.8.4 使用binary log](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231273)

[3.2.8.5 还原单个表](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231274)

[3.2.8.6 LRU dump备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231275)

[3.2.8.7 xtrabackup的限制](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231276)

[3.2.8.8 References](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231277)

[3.2.9 xbstream](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231278)

[3.2.10 xbcrypt](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231279)

[3.2.11 Xtrabackup原理](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231280)

[4 如何使用和案例(How-tos and Recipes)](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231281)

[4.1 innobackupex案例](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231282)

[4.1.1 本地全备(备份，预备，还原)](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231283)

[4.1.2 使用Stream备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231284)

[4.1.3 创建增量备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231285)

[4.1.4 创建压缩备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231286)

[4.1.5 备份还原独立分区](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231287)

[4.1.5.1 创建备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231288)

[4.1.5.2 预备备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231289)

[4.1.5.3 从备份还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231290)

[4.2 xtrabackup案例](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231291)

[4.3 其他案例](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231292)

[4.3.1 六个步骤安装一个slave](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231293)

[4.3.2 再增加一个slave](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231294)

[4.3.3 使用复制和pt-checksum验证备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231295)

[4.3.4 如何创建基于GTID的Slave](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231296)

[4.4辅助工具手册](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231297)

[参考](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_Toc396231298)

**2 安装XtraBackup**

这里只介绍yum和rpm安装方法

其他安装方法查看：

  [Installing Percona XtraBackup from Binaries](http://www.percona.com/doc/percona-xtrabackup/2.2/installation.html)

  [Compiling and Installing from Source Code](http://www.percona.com/doc/percona-xtrabackup/2.2/installation/compiling_xtrabackup.html)

**2.1 安装XtraBackup binary版本**

**2.1.1 yum的安装方法：**

**自动**

$ rpm -Uhv http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86\_64.rpm

然后会看到：

Retrieving http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86\_64.rpm

Preparing...                ########################################### [100%]

   1:percona-release        ########################################### [100%]

**手动**

[percona]

name = CentOS $releasever - Percona

baseurl=http://repo.percona.com/centos/$releasever/os/$basearch/

enabled = 1

gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-percona

gpgcheck = 1

**测试安装库**

使用yum list | grep percona来确保安装

yum list | grep percona

会出现以下信息：

percona-release.x86\_64                     0.0-1                       installed

...

Percona-Server-client-51.x86\_64            5.1.47-rel11.1.51.rhel5     percona

Percona-Server-devel-51.x86\_64             5.1.47-rel11.1.51.rhel5     percona

Percona-Server-server-51.x86\_64            5.1.47-rel11.1.51.rhel5     percona

Percona-Server-shared-51.x86\_64            5.1.47-rel11.1.51.rhel5     percona

Percona-Server-test-51.x86\_64              5.1.47-rel11.1.51.rhel5     percona

...

xtrabackup.x86\_64                          1.2-22.rhel5                percona

**2.1.2 直接下载rpm包安装**

使用wget下载rpm包，然后通过rpm包安装

参考：<http://www.cnblogs.com/cosiray/archive/2012/03/02/2376595.html>

参考：<http://blog.163.com/ji_1006/blog/static/10612341201382355716623/>

**3 XtraBackup使用手册**

**3.1 使用innobackupex脚本**

innobackupex是perl脚本对xtrabackup的封装，和功能扩展。

**3.1.1 备份预备工作**

**权限和连接**

xtrabackup需要连接到数据库和datadir操作权限。

xtrabackup或者innobackupex在使用过程中设计到2类用户权限：

1.系统用户，用来调用innobackupex或者xtrabackup

2.数据库用户，数据库内使用的用户

**连接到服务**：innobackupex或者xtrabackup通过—user和—password连接到数据库服务

$ innobackupex --user=DBUSER --password=SECRET /path/to/backup/dir/

$ innobackupex --user=LUKE  --password=US3TH3F0RC3 --stream=tar ./ | bzip2 -

$ xtrabackup --user=DVADER --password=14MY0URF4TH3R --backup --target-dir=/data/bkps/

**其他连接选项**：

| Option | Description |
| --- | --- |
| –port | The port to use when connecting to the database server with TCP/IP. |
| –socket | The socket to use when connecting to the local database. |
| –host | The host to use when connecting to the database server with TCP/IP. |

**需要的权限**：连接到服务是为了执行备份，需要在datadir上有read，write和execute权限。在数据库中需要以下权限：

   RELOAD和LOCK TABLES权限为了执行FLUSH TABLES WITH READ LOCK   。

   REPLICATION CLIENT为了获取binary log 位置

   CREATE TABLESPACE权限为了导入表，用户表级别的恢复

   SUPER权限在slave环境下备份用来启动和关闭slave线程

mysql>**CREATEUSER**'bkpuser'@'localhost' IDENTIFIED **BY**'s3cret';

mysql>**GRANT** RELOAD, **LOCK** TABLES, REPLICATION CLIENT **ON**\*.\***TO**'bkpuser'@'localhost';

mysql> FLUSH **PRIVILEGES**;

**3.1.2 全备和全备还原**

**3.1.2.1 使用innobackupex创建全备**

创建全备

$ innobackupex --user=DBUSER --password=DBUSERPASS /path/to/BACKUP-DIR/

会输出以下信息

innobackupex: Backup created in directory '/path/to/BACKUP-DIR/2013-03-25\_00-00-09'

innobackupex: MySQL binlog position: filename 'mysql-bin.000003', position 1946

111225 00:00:53  innobackupex: completed OK!

从信息中会发现备份被创建在/path/to/BACKUP-DIR/2013-03-25\_00-00-09

**内部机制**：在备份的时候innobackupex会调用xtrabackup来备份innodb表，并复制所有的表定义，其他引擎的表(MyISAM,MERGE,CSV,ARCHIVE)。

**其他选项**:

--no-timestamp，指定了这个选项备份会直接备份在BACKUP-DIR，不再创建时间戳文件夹。

--default-file，指定配置文件，用来配置innobackupex的选线。

**3.1.2.2 使用innobackupex预备全备**

创建完备份之后数据被没有马上可以被还原，需要回滚未提交事务，前滚提交事务，让数据库文件保持一致性。

innobackupex使用—apply-log来做预备备份

$ innobackupex --apply-log /path/to/BACKUP-DIR

成功则会输出：

111225  1:01:57  InnoDB: Shutdown completed; log sequence number 1609228

111225 01:01:57  innobackupex: completed OK!

成功后，备份可以被用来还原数据库了。

**内部机制**：读取备份文件夹中的配置文件，然后innobackupex重做已提交事务，回滚未提交事务，之后数据就被写到了备份的数据文件(innodb文件)中，并重建日志文件。这一步隐式调用了2次xtrabackup –prepare。跟多关于xtrabackup可以看之后的章节。

其他选项：

--user-memory：指定预备阶段可使用的内存，内存多则速度快，默认为10MB

$ innobackupex --apply-log --use-memory=4G /path/to/BACKUP-DIR

**3.1.2.3 使用innobackupex还原备份**

使用innobackupex --copy-back来还原备份

$ innobackupex --copy-back /path/to/BACKUP-DIR

会根据my.cnf复制所有备份到datadir下：

innobackupex: Finished copying back files.

111225 01:08:13  innobackupex: completed OK!

注：datadir必须是为空的，innobackupex –copy-back不会覆盖已存在的文件，还要注意，还原时需要先关闭服务，如果服务是启动的，那么就不能还原到datadir。

之后需要修改文件的所有者和权限：

$ chown -R mysql:mysql /var/lib/mysql

**3.1.3 增量备份和还原**

增量备份呢是为了减少空间使用和备份的时间。

增量备份的实现，依赖于innodb页上面的LSN（log sequence number），每次对数据库的修改都会导致LSN自增。

增量备份会复制指定LSN之后的所有数据页。

**3.1.3.1 创建增量备份**

**创建全备**

在创建增量备份之前需要一个全备，不然增量备份是没有意义的。

$ innobackupex /data/backups

这样就会在/data/backups下创建一个时间戳文件夹，如 /data/backups/2013-03-31\_23-01-18，然后文件夹内是备份文件。

检查备份文件夹下的xtrabackup-checkpoints，会有一下信息：

backup\_type = full-backuped

from\_lsn =0

to\_lsn =1291135

**创建第一个增量备份**

然后使用—incremental创建增量备份

$ innobackupex --incremental /data/backups --incremental-basedir=BASEDIR

BASEDIR指向之前的全备， /data/backups/2013-03-31\_23-01-18，成功后备份会生成在/data/backups下的时间戳目录中，如：/data/backups/2013-04-01\_23-01-18 ，把这个目录叫为记为 INCREMENTAL-DIR-1方面之后使用。

然后查看xtrabackup-checkpoints:

backup\_type = incremental

from\_lsn =1291135

to\_lsn =1352113

可以发现和全备不同的是，backup\_type为incremental，from\_lsn不为0。

**然后再创建一个增量备份**

在INCREMENTAL-DIR-1的基础上再创建一个增量备份，记为INCREMENTAL-DIR-2。

$ innobackupex --incremental /data/backups --incremental-basedir=INCREMENTAL-DIR-1

**增量备份替代方法**

可以使用指定—incremental-lsn来代替—incremental-basedir的方法创建增量备份。

innobackupex --incremental /data/backups --incremental-lsn=1291135

innobackupex --incremental /data/backups --incremental-lsn=1358967

注意：xtrabackup只会影响xtradb或者innodb的表，其他引擎的表在增量备份的时候只会复制整个文件，不会差异。

**3.1.3.2 预备增量备份**

预备增量备份需要2个步骤：

1.需要先预备全备，但是只重做已提交事务，不回滚未提交事务，然后应用到全备，也是只重做已提交事务，不回滚未提交事务

2.回滚未提交事务

如果已经回滚了未提交事务，那么就无法再应用增量备份。

注：在mariadb 10.0 上测试发现不加—redo-only预备全备，然后使用 –redo-only应用增量备份，mysql服务能够正常启动并且数据被成功还原

**在全备上，使用—redo-only只做已提交事务，不回滚未提交事务**

innobackupex --apply-log --redo-only BASE-DIR

会出现以下结果：

120103 22:00:12 InnoDB: Shutdown completed; log sequence number 1291135

120103 22:00:12 innobackupex: completed OK!

**应用第一个增量备份**

innobackupex --apply-log --redo-only BASE-DIR --incremental-dir=INCREMENTAL-DIR-1

输出结果，注意LSN的变化：

120103 22:08:43 InnoDB: Shutdown completed; log sequence number 1358967

120103 22:08:43 innobackupex: completed OK!

如果没有指定—incremental-dir，那么innobackupex会使用最近的一个在basedir中被创建的子目录。

**应用另外一个备份**

innobackupex --apply-log BASE-DIR --incremental-dir=INCREMENTAL-DIR-2

因为是最后一个增量备份所以没有必要再加—redo-only，这样最后一个增量也被应用到全备上了。

注：--redo-only除了最后一个不用加之外，其他的增量应用都要加，最后一个应用的时候可以直接进入回滚未提交事务阶段。如果加了也没事儿，服务启动的时候会进入recovery过程，来回滚

需要注意的是，应用增量备份的时候只能按照备份的顺序来应用。如果应用顺序错误，那么备份就不可用。如果无法确定顺序，可以使用xtrabackup-checkpoints来确定顺序。

**回滚未提交事务**

当应用完所有增量备份的时候，就需要回滚所有为完成事务（如果最后一步加了 –redo-only就需要回滚未提交，不执行的话在服务启动阶段服务会处理未提交事务）。

innobackupex --apply-log BASE-DIR

Note that the iblog\* files will not be created by **innobackupex**, if you want them to be created, use **xtrabackup –prepare**on the directory. Otherwise, the files will be created by the server once started.

注：

文中提到innodb事务日志(iblog\*)不会被创建，但是测试下使用了最后一步**回滚未提交事务**发现有iblog\*文件，而且上文提到 innobackupex会隐式执行两次 xtrabackup –prepare，在下文介绍xtrabackup时会提到，执行2次xtrabackup –preare会创建iblog\*文件，与文中提到不符。

**3.1.3.3 还原增量备份**

还原增量备份其实和还原全备一样

innobackupex --copy-back BASE-DIR

注意事项可以看：[使用innobackupex还原备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_3.1.2.3_使用innobackupex还原备份)

**3.1.3.4 可以使用流做增量备份**

先进行一个全备：

innobackupex /data/backups

使用本地：

innobackupex --incremental --incremental-lsn=LSN-number --stream=xbstream ./ > incremental.xbstream

解包方法

xbstream -x < incremental.xbstream

使用本地备份流到远程并解包

innobackupex  --incremental --incremental-lsn=LSN-number --stream=xbstream ./ | /

ssh user@hostname " cat - | xbstream -x -C > /backup-dir/"

**3.1.4 部分备份和还原**

xtrabackup可以使用部分备份，但是只能在一个表一个文件的状况下才能使用，设置mysql选项：innodb\_file\_per\_table。

还原部分备份使用表导入的方式，而不是—copy-back选项。

尽管很多场景下可以通过直接复制文件的方式，但是会产生一致性问题不建议使用。

**3.1.4.1 创建部分备份**

部分备份有3个选项可以使用：

--include:设置正则表达式的格式，匹配的就备份

--table-file:在文件中指定要备份的表，然后通过这个选项传入文件

--database:指定数据库列表

**使用include方式**

include 方式数据库名也可以匹配：

$ innobackupex --include='^mydatabase[.]mytable' /path/to/backup

这个选项是传给xtrabackup –tables，所有的数据库目录都会被创建，但是里面可能是空的。

**使用tables-file方式**

如:

$ echo "mydatabase.mytable" > /tmp/tables.txt

$ innobackupex --tables-file=/tmp/tables.txt /path/to/backup

这个选项是应用xtrabackup –tablefile，只有匹配到表的数据库目录才会被创建

**使用database方式**

innobackupex可以传递用空格隔开的数组，格式为：databasename[.tablename]

$ innobackupex --databases="mydatabase.mytable mysql" /path/to/backup

注意：--databasees选项只会对非innodb引擎表和frm文件产生影响，对于innodb数据文件总是备份的

**3.1.4.2 预备部分备份**

部分备份的预备需要使用—export：

$ innobackupex --apply-log --export /path/to/partial/backup

会出现以下，是因为innodb表保存了数据文件但是没有保存frm文件。

111225  0:54:06  InnoDB: Error: table 'mydatabase/mytablenotincludedinpartialb'

InnoDB: in InnoDB data dictionary has tablespace id 6,

InnoDB: but tablespace with that id or name does not exist. It will be removed from data dictionary.

之后会发现生成了.exp和.cfg文件。exp文件适用于percona server，cfg适用于mariadb和mysql。mariadb 10.0可以直接通过ibd和frm文件import。mysql 5.6之后可以不使用cfg来进行import，cfg如果存在会被用来做表结构的验证。

在已经预备好的备份上，可以使用—export和—apply-log创建.exp文件。

**3.1.4.3 还原部分备份**

先创建一个表，表结构需要和被还原的一样。

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

然后discard表空间

OTHERSERVER|mysql> ALTER TABLE mydatabase.mytable DISCARD TABLESPACE;

之后把文件复制到相应的目录下（注意文件的所有者和文件权限），需要文件.ibd,.exp或者.cfg文件(.cfg文件用户mysql5.6)。

然后import表空间

OTHERSERVER|mysql> ALTER TABLE mydatabase.mytable IMPORT TABLESPACE;

**3.1.5 窄备份**

窄备份指不备份secondary索引数据。这样可以减少备份的大小。缺点就是需要重建索引，会很慢。

**3.1.5.1 创建窄备份**

$ innobackupex --compact /data/backups

创建了之后查看xtrabackup\_checkpoint

backup\_type = full-backuped

from\_lsn =0

to\_lsn =2888984349

last\_lsn =2888984349

compact =1

compact=1说明该备份是窄备份。

**3.1.5.2 预备窄备份**

在预备窄备份的时候需要使用—rebuild-indexes来重新创建索引

$ innobackupex --apply-log --rebuild-indexes /data/backups/2013-02-01\_10-29-48

从输出上可以看到索引被重建

130201 10:40:20  InnoDB: Waiting for the background threads to start

Rebuilding indexes for table sbtest/sbtest1 (space id: 10)

  Found index k\_1

  Dropping 1 index(es).

  Rebuilding 1 index(es).

Rebuilding indexes for table sbtest/sbtest2 (space id: 11)

  Found index k\_1

  Found index c

  Found index k

  Found index c\_2

  Dropping 4 index(es).

  Rebuilding 4 index(es).

对于增量备份的应用可以先不重建索引，在应用最后一个差异备份的时候使用—rebuild-index来创建索引，每次都应用都重建索引太花时间。

注意：为了重建速度，可以使用并发创建索引，使用参数—rebuild-threads指定并发数。

**3.1.5.3 还原窄备份**

窄备份还原和全备还原一样直接使用—copy-back选项。

具体看：[使用innobackupex还原备份](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_3.1.2.3_使用innobackupex还原备份)

**3.1.6 备份加密**

具体看：[Encrypted Backups](http://www.percona.com/doc/percona-xtrabackup/2.2/innobackupex/encrypted_backups_innobackupex.html)

**3.1.7 其他功能**

**3.1.7.1 备份压缩和流**

Stream模式下，Xtrabackup的STDOUT可以指定tar或者xbstream格式输出。

流允许，其他程序过滤备份输出，提供更大的灵活存储backup。

使用流特性，需要指定—stream选项

$ innobackupex --stream=tar /tmp

innobackupex会用子程序启动xtrabackup –log-stream 定向到临时文件，然后使用xbstream把所有数据文件steam到STDOUT。

当压缩启动，xtrabackup压缩所有输出数据，但是元数据和非innodb文件不能被压缩。现在唯一支持的压缩算法是quicklz。会生产qpress归档格式的文件。

使用xbstream可以平法复制压缩可以提高备份速度。

使用xbstream流备份：

$ innobackupex --stream=xbstream /root/backup/ > /root/backup/backup.xbstream

使用流压缩:

$ innobackupex --stream=xbstream --compress /root/backup/ > /root/backup/backup.xbstream

解包：

$ xbstream -x <  backup.xbstream -C /root/backup/

流压缩并备份到另外一台机器：

$ innobackupex --compress --stream=xbstream /root/backup/ | ssh user@otherhost "xbstream -x -C /root/backup/"

使用tar备份：

$ innobackupex --stream=tar /root/backup/ > /root/backup/out.tar

使用tar流并备份到其他服务器

$ innobackupex --stream=tar ./ | ssh user@destination \ "cat - > /data/backups/backup.tar"

提取tar流，需要加i参数

$ tar -xizf backup.tar.gz

也可以压缩流

$ innobackupex --stream=tar ./ | gzip - > backup.tar.gz

$ innobackupex --stream=tar ./ | bzip2 - > backup.tar.bz2

**3.1.7.2 在复制环境下备份**

有2个选项用于从复制环境下备份

**slave-info**

--slave-info，会打印binary log的位置和master server名，并且以change master的方式写到xtrabackup\_slave\_info中。

**safe-slave-backup**

--safe-slave-backup,为了保证复制状态的一致性，这个选项会关闭slave sql线程，等待直到SHOW STATUS 中的Slave\_open\_temp\_tabls为了才启动备份。如果等待时间超过—safe-slave-backup-timeout就会报错默认300秒。备份成功后 slave sql thread会自动启动。

**3.1.7.3 加速备份进程**

**使用parallel和compress-threads加速**

当有多个文件时，可以使用使用—parallel加速备份，这个选项会指定xtrabackup备份文件的线程数。

$ innobackupex --parallel=4 /path/to/backup

如果使用xbstream可以考虑通过—compress-threads加速压缩进程，默认为1.

$ innobackupex --stream=xbstream --compress --compress-threads=4 ./ > backup.xbstream

**使用rsync加速**

为了加速复制过程，最小化FLUSH TABLES WITH READ LOCK堵塞时间，使用innobackupex –rsync。使用了这个选项所有文件都会在一个cp命令里面，而不是每个文件一个cp。并且innobackupex会调用2次 rsync，一次在执行FLUSH TABLES WITH READ LOCL之前，一次在之后。第二次执行的时候会把第一次之后的修改过的数据。

**3.1.7.4 节流(throttling)备份**

尽管innobackupex不会堵塞数据库操作，但是备份终会消耗系统资源。为了减少资源消耗，可以使用—throttle来限制每秒钟读写对次数。

**3.1.7.5 还原独立表**

使用xtrabackup来导出指定表，然后导入到XtraDB或者Mysql 5.6（测试可以的导入mariadb 10.0）

mariadb 10.0可以直接复制 ibd然后通过import tablespace倒入。

**导出表**

导出表使用—export选项：

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

会在发现多了一个.exp文件和.cfg文件(用于不同的mysql版本)

$ find /data/backups/mysql/ -name export\_test.\*

/data/backups/mysql/test/export\_test.exp

/data/backups/mysql/test/export\_test.ibd

/data/backups/mysql/test/export\_test.cfg

**导入表**

先创建一个表，表结构需要和被还原的一样。

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

然后discard表空间

OTHERSERVER|mysql> ALTER TABLE mydatabase.mytable DISCARD TABLESPACE;

之后把文件复制到相应的目录下（注意文件的所有者和文件权限），需要文件.ibd,.exp或者.cfg文件(.cfg文件用户mysql5.6)。

然后import表空间

OTHERSERVER|mysql> ALTER TABLE mydatabase.mytable IMPORT TABLESPACE;

**3.1.7.6 时间点还原**

和mysql手册中介绍的时间点还原一样，xtrabackup也是通过binary log进行时间点还原。

先进行备份

$ innobackupex /path/to/backup --no-timestamp

然后进行预备

$ innobackupex --apply-log /path/to/backup

在服务器中找出操作binary log和当前binary log状态

mysql> SHOW BINARY LOGS;

+------------------+-----------+

| Log\_name         | File\_size |

+------------------+-----------+

| mysql-bin.000001 |       126 |

| mysql-bin.000002 |      1306 |

| mysql-bin.000003 |       126 |

| mysql-bin.000004 |       497 |

+------------------+-----------+

mysql> SHOW MASTER STATUS;

+------------------+----------+--------------+------------------+

| File             | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |

+------------------+----------+--------------+------------------+

| mysql-bin.000004 |      497 |              |                  |

+------------------+----------+--------------+------------------+

然后查看 xtrabackup\_binlog\_info确定备份的binary log位置：

$ cat /path/to/backup/xtrabackup\_binlog\_info

mysql-bin.000003      57

还原数据库

$ innobackupex --copy-back /path/to/backup

然后使用mysqlbinlog取出binary log中的记录，然后输出到文件

$ mysqlbinlog /path/to/datadir/mysql-bin.000003 /path/to/datadir/mysql-bin.000004 \

    --start-position=57 > mybinlog.sql

然后检查输出的文件，确定要恢复到的位置，然后执行恢复

$ mysqlbinlog /path/to/datadir/mysql-bin.000003 /path/to/datadir/mysql-bin.000004 \

    --start-position=57 --stop-datetime="11-12-25 01:00:00" | mysql -u root -p

**3.1.7.7 提高对FLUSH TABLES WITH READ LOCK控制**

在备份的时候，为了保证数据一致性，在备份非innodb表的之前会先使用FLUSH TABLES WITH READ LOCK，这个语句在有长时间查询运行的情况下也能执行，但是其他的所有事都会被堵塞，Waiting for table flush或者Waiting for master to send event，这个时候不应该kill FLUSH TABLES WITH READ LOCK，而是应该kill掉那个大的查询。因为当有大的查询的时候，FLUSH TABLES WITH READ LOCK会被卡住。

为了能够避免这种事情发生需要实现2个事情：

1.innobackupex等一个好的时机运行

2.innobackupex可以kiil 所有查询或者只能存在SELECT查询。即kill所有阻止获取全局锁的查询。

**等待查询完成**

为了避免innobackupex等待FLUSH TABLES WITH READ LOCK执行太长时间，可以使用innobackupex –lock-wait-timeout，可以用来限制等待时间，一旦超时就会报错退出。

另外一个是设置等待查询的类型，innobackupex  --lock-wait-query-type 可取的值是all或者update，如果为all那么会等待所有长运行查询完成，如果是update，会等待除select之外的DML完成。

--lock-wait-threshold用来定义 --locl-wait-query-type中的长运行查询，如果超过--lock-wait-threshold都算长运行查询。

**Kill堵塞查询**

innobackupex可以kill所有阻止获取全局锁的查询。

可以通过指定--kill-long-queries-timeout用来指定执行FLUSH TABLES WITH READ LOCK后还可以执行的时间，0为不kill，--kill-long-query-type用来指定超时之后，kill的查询类型，可以是all或者select。

例如：

$ innobackupex --lock-wait-threshold=40 --lock-wait-query-type=all --lock-wait-timeout=180 --kill-long-queries-timeout=20 --kill-long-query-type=all /data/backups/

**3.1.8 innobackupex工作原理**

innobackupex是perl脚本，封装了xtrabackup和tar4ibd的功能。

**备份**

如果没有指定，innobackupex默认认为是备份模式。

默认innobackupex会以--suspend-at-end启动xtrabackup，来复制innodb文件。当xtrabackup复制完成，innobackupex发现创建的xtrabckup\_suspended\_2文件，就执行FLUASH TABLES WITH READ LOCL复制其他文件。

xtrabackup的确定，如果运行的时候没有带ibbackup参数，那么innobackupex会从xtrabackup\_binary上找，然后读取要启动xtrabackup的二进制文件。否则只能通过连接mysql服务确定使用的二进制文件。如果不能连接就错误。

决定好二进制文件之后，断开连接以二进制方式启动binary。

如果不是增量备份，连接到服务，如果启动了--safe-slave-backup等待slave，然后获取全局read lock，防止其他引擎的表被修改，读锁只锁定非innodb表，并且在innodb表复制完数据和日志后进行。

当所有文件备份完之后，恢复ibbackup并等待完成对事务复制，复制在备份期间生产的事务。然后释放表锁，slave启动并且连接关闭，然后删除xtrabckup\_suspended\_2文件。

**还原**

还原数据库使用 --copy-back选项

innobackupex通过读取my.cnf中的 [datadir](http://www.percona.com/doc/percona-xtrabackup/2.2/glossary.html#term-datadir), [innodb\_data\_home\_dir](http://www.percona.com/doc/percona-xtrabackup/2.2/glossary.html#term-innodb-data-home-dir), [innodb\_data\_file\_path](http://www.percona.com/doc/percona-xtrabackup/2.2/glossary.html#term-innodb-data-file-path),[innodb\_log\_group\_home\_dir](http://www.percona.com/doc/percona-xtrabackup/2.2/glossary.html#term-innodb-log-group-home-dir)并检查目录是否存在。

然后复制MyISAM表，索引，然后复制innodb表和索引，最后日志文件。复制的时候会保留文件属性，需要修改用户所有者。

恢复除了--cop-back也可以使用--move-back，不同的是一个是复制，一个是移动。

**3.1.9 Reference**

主要介绍一些选项：[The **innobackupex** Option Reference](http://www.percona.com/doc/percona-xtrabackup/2.2/innobackupex/innobackupex_option_reference.html)

**3.2 使用Xtrabackup**

**3.2.1 选择bianry**

Xtrabackup有4个binary：Xtrabackup，Xtrabackup\_51，Xtrabackup\_55，Xtrabackup\_56。

服务和binary对照：

| Server | **xtrabackup** binary |
| --- | --- |
| MySQL 5.1.\* | xtrabackup\_51 |
| MySQL 5.1.\* with InnoDB plugin | xtrabackup |
| MySQL 5.5.\* | xtrabackup\_55 |
| MySQL 5.6.\* | xtrabackup\_56 |
| MariaDB 5.1.\* | xtrabackup |
| MariaDB 5.2.\* | xtrabackup |
| MariaDB 5.3.\* | xtrabackup |
| MariaDB 5.5.\* | xtrabackup\_55 |
| MariaDB 10.0.\* | xtrabackup\_56 |
| Percona Server 5.0 | xtrabackup\_51 |
| Percona Server 5.1 | xtrabackup |
| Percona Server 5.5 | xtrabackup\_55 |
| Percona Server 5.6 | xtrabackup\_56 |

**3.2.2 配置Xtrabackup**

所有的Xtrabackup配置都是通过选项设置，可以是命令行参数，也可以通过配置文件my.cnf。

xtrabackup会读取[mysql],[xtrabackup]选项，若Xtrabackup读入，会覆盖[mysqld]的选项。

并不是所有的配置都要写入配置文件，写配置文件只是为了方便。

xtrabackup 并不接受和mysqld一样的语法，有些语法不支持如，--set-variable=<variable>=<value>要用 --variable=value语法。

xtrabackup并不需要特殊的存储，但是如果是NFS并不是以sync挂载，那么在执行fsync的时候可能并不是真正的同步数据。

**3.2.3 创建全备和还原**

**3.2.3.1 创建全备**

使用xtrabackup创建全备需要指定选项-backup，还需要指定--target\_dir，如果target不存在，会创建一个，如果存在那么如果是空的就会成功，如果不是空的，不会覆盖已存在文件，并且报错。

主要完成2个任务：

1.开启一个log copy线程，用来监控innodb日志文件(ib\_logfile\*),如果修改就复制到xtrabackup\_logfile，因为复制会持续很长时间，所以恢复进程需要所有从备份开始到结束的所有日志。

2.复制innodb数据文件到目标目录，这个并不是简单的复制，是和innodb引擎一样，从数据目录中一页一页的复制。

当数据文件完成复制，xtrabackup会停止对日志的复制，并在目标目录中创建xtrabackup\_checkpoint文件。

$ xtrabackup --backup --datadir=/var/lib/mysql/ --target-dir=/data/backups/mysql/

在备份输出的时候，可以看到日志的复制。

>> log scanned up to (3646475465483)

>> log scanned up to (3646475517369)

>> log scanned up to (3646475581716)

>> log scanned up to (3646475636841)

>> log scanned up to (3646475718082)

>> log scanned up to (3646475988095)

>> log scanned up to (3646476048286)

>> log scanned up to (3646476102877)

>> log scanned up to (3646476140854)

[01] Copying /usr/local/mysql/var/ibdata1

     to /usr/local/mysql/Backups/2011-04-18\_21-11-15/ibdata1

[01]        ...done

注意：日志复制线程是每秒检查一次，查看是否有新的日志被写入，因为日志文件是被回绕写的，所以日志复制线程要更上日志文件的修改，如果没有复制日志被覆盖了，那么就会报错。

备份的时间长短依赖于数据库的大小，然后时间都可以停止数据库，因为不会去修改数据库数据。

**3.2.3.2 预备全备**

备份完数据库之后，下一步是预备数据库，因为数据文件在某个时间点上，并不是一致的，所以需要预备让数据文件在某个时间点一致，--prepare就是来完成，让数据文件保持一致性。

注意：在innobackupex --apply-log的时候，innodb就是自动读取配置文件back-my.cnf就是使用--defaults-file=xxx来指定一个配置文件，传递给xtrabackup用于预备数据库备份。

可以在任何机器上进行预备，没必要在原服务器或者要还原的服务器上进行。

在恢复阶段，xtrabackup嵌入了修改过的innodb，禁止了innodb的标准安全监察，如日志文件大小是否准确。

prepare阶段就是使用这个嵌入的innodb来做通过日志文件对数据文件进行crash恢复。

xtrabackup --prepare --target-dir=/data/backups/mysql/

当预备完成就会有以下输出：

101107 16:40:15  InnoDB: Shutdown completed; log sequence number <LSN>

现在备份一致性已经没问题了，可以准备还原，但是为了能够更快还原，可以再执行一次预备，第一次执行的时候只让数据文件保持一致性，并没有创建日志文件。第二次执行的时候会创建日志文件。如果第一次预备后还原，启动服务，服务会自动创建日志文件，但是比较滑时间。当第二次运行预备的时候有一下输出:

**$** xtrabackup --prepare --target-dir=/data/backups/mysql/

xtrabackup: This target seems to be already prepared.

xtrabackup: notice: xtrabackup\_logfile was already used to '--prepare'.

101107 16:54:10  InnoDB: Log file ./ib\_logfile0 did not exist: new to be created

InnoDB: Setting log file ./ib\_logfile0 size to <SIZE> MB

InnoDB: Database physically writes the file full: wait...

101107 16:54:10  InnoDB: Log file ./ib\_logfile1 did not exist: new to be created

InnoDB: Setting log file ./ib\_logfile1 size to <SIZE> MB

InnoDB: Database physically writes the file full: wait...

101107 16:54:15  InnoDB: Shutdown completed; log sequence number 1284108

如果是第二次运行，运行时会有以下提示：

xtrabackup: This target seems to be already prepared.

xtrabackup: notice: xtrabackup\_logfile was already used to '--prepare'.

不推荐在执行预备的时候终端过程，这样可能会导致数据文件异常。

如果视图要加入增量备份，那么使用--apply-log-only，不然加不上增量备份。

**3.2.3.3 还原全备**

xtrabackup没有什么功能来还原备份，可以直接通过rsync，cp来还原数据库

注意：注意保持datadir必须是空的，并且mysql服务是停止的。不能还原到已经在运行的mysql服务中。

通过rsync还原：

$ rsync -avrP /data/backup/ /var/lib/mysql/

还原后注意修改所有者

$ chown -R mysql:mysql /var/lib/mysql

注意：xtrabackup只备份innodb数据文件，不会备份其他引擎的表，和frm文件。如果要对整个库备份还原可以使用innodbbackupex

**3.2.4 增量备份和还原**

**3.2.4.1 增量备份**

**增量备份原理**

xtrabackup和innobackupex都可以实现增量备份，也就是说只备份修改过的数据。

增量备份实现，依赖于innodb页中的LSN(log sequence number)增量备份会复制比之前的增量或者全备新的lsn页。

有2个算法找这样的页：

1.直接扫描数据页，并复制大于上次全备或者增量的lsn的页

2.类似percona server，启动 [changed page tracking](http://www.percona.com/doc/percona-server/5.5/management/changed_page_tracking.html)  会记录页的修改。这样的记录会被保存在一个map文件中。xtrabackup会使用这个文件读取需要备份的页数据。当然也可以使用--incremental-force-scan强制扫描所有数据页。

增量备份并不是比较全备的数据，如果没有上次的备份，可以使用指定--incremental-lsn来进行增量备份。增量备份只会备份比指定lsn大的数据页。当然需要全备来还原增量备份，不然增量备份是没有意义的。

**创建增量备份**

首先创建全备

xtrabackup --backup --target-dir=/data/backups/base --datadir=/var/lib/mysql/

查看xtrabackup\_checkpoint信息：

backup\_type = full-backuped

from\_lsn =0

to\_lsn =1291135

然后进行增量备份：

xtrabackup --backup --target-dir=/data/backups/inc1 \

--incremental-basedir=/data/backups/base --datadir=/var/lib/mysql/

在/data/backups/inc1下包含了delta文件，如果ibdata1.delta和test/table1.ibd.delta，检查增量备份的xtrabackup\_checkpoint：

backup\_type = incremental

from\_lsn =1291135

to\_lsn =1291340

在做一个增量备份：

xtrabackup --backup --target-dir=/data/backups/inc2 \

--incremental-basedir=/data/backups/inc1 --datadir=/var/lib/mysql/

**3.2.4.2 预备增量备份**

预备增量备份和在innobackupex一样，先不回滚方式应用全备，然后应用增量备份。

在innobackupex上使用 --apply-log-only来重做，但不回滚。

现在已有备份：

/data/backups/base

/data/backups/inc1

/data/backups/inc2

预备全备，不回滚未提交事务：

xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base

成功后输出：

101107 20:49:43  InnoDB: Shutdown completed; log sequence number 1291135

应用第一个增量备份：

xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base \

--incremental-dir=/data/backups/inc1

增量备份被应用到/data/backups/base，应用后输出：

incremental backup from 1291135 is enabled.

xtrabackup: cd to /data/backups/base/

xtrabackup: This target seems to be already prepared.

xtrabackup: xtrabackup\_logfile detected: size=2097152, start\_lsn=(1291340)

Applying /data/backups/inc1/ibdata1.delta ...

Applying /data/backups/inc1/test/table1.ibd.delta ...

.... snip

101107 20:56:30  InnoDB: Shutdown completed; log sequence number 1291340

应用最后一个增量备份：

xtrabackup --prepare --target-dir=/data/backups/base \

--incremental-dir=/data/backups/inc2

注意：除了最后一个外其他的都要使用--apply-log-only，如果最后一个也用了--apply-log-only，就还原了文件还是一直的，但是没有回滚未提交事务，当服务启动的时候会自动回滚未提交事务。

**3.2.5 使用归档日志做增量备份**

**3.2.5.1 原理**

在percona server 5.6.11-60.3加入了新功能，为xtradb日志归档，这个功能是，在老的日志文件被重写之前会被复制，因此保存了所有的redo日志。

归档日志的文件格式，ib\_log\_archive\_<LSN>，LSN表示这个归档文件开始的lsn。

ib\_log\_archive\_00000000010145937920

ib\_log\_archive\_00000000010196267520

这个功能由innodb\_log\_archive启动，保存的位置为innodb\_log\_arch\_dir(默认为数据文件夹)。

**3.2.5.2 创建备份**

创建一个全备

xtrabackup\_56 --backup --target-dir=/data/backup/ --datadir=/var/lib/mysql/

xtrabackup\_checkpoint如下：

backup\_type = full-backuped

from\_lsn =0

to\_lsn =1546908388

last\_lsn =1574827339

compact =0

**3.2.5.3 使用归档日志来预备备份**

xtrabackup\_56 --prepare --target-dir=/data/backup/ --innodb-log-arch-dir=/data/archived-logs/

执行后查看xtrabackup\_checkpoint，backup-type被修改：

backup\_type = full-prepared

from\_lsn =0

to\_lsn =1546908388

last\_lsn =1574827339

compact =0

另外也可以指定时间点预备：

xtrabackup\_56 --prepare --target-dir=/data/backup/ --innodb-log-arch-dir=/data/archived-logs/ --to-archived-lsn=5536301566

**3.2.6 部分备份和预备**

当服务使用innodb\_file\_per\_table的时候，xtrabackup支持部分备份。

**3.2.6.1 备份部分备份**

**使用--tables进行部分备份**

使用--tables备份，该选项的值是一个正则表达式，匹配的表名都会被备份。

备份test下的所有表：

$ xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/data/backups/ \

--tables="^test[.].\*"

备份test.t1表

$ xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/data/backups/ \

--tables="^test[.]t1"

**使用--tables-file进行备份**

--tables-file指向一个选项包了表名，如：

$ echo “mydatabase.mytable” > /tmp/tables.txt $ xtrabackup –backup –tables-file=/tmp/tables.txt

**3.2.6.2 预备备份**

在prepare的时候会出现很多warnings，是因为表存在在innodb，但是对于的ibd不存在，这些表在还原备份启动innodb的时候会被删除。

InnoDB: Reading tablespace information from the .ibd files...

101107 22:31:30  InnoDB: Error: table 'test1/t'

InnoDB: in InnoDB data dictionary has tablespace id 6,

InnoDB: but tablespace with that id or name does not exist. It will be removed from data dictionary.

**3.2.7 窄备份和预备**

窄备份是在备份的是否不备份secondary index让备份文件边小。窄备份可以通过--compact启动。

**3.2.7.1 创建窄备份**

$ xtrabackup --backup --compact --target-dir=/data/backups

查看备份后的xtrabackup\_checkpoint，compact为1说明是窄备份

backup\_type = full-backuped

from\_lsn =0

to\_lsn =2888984349

last\_lsn =2888984349

compact =1

**3.2.7.2 预备窄备份**

在预备的时候为了重建索引，需要使用选项--rebuild-indexes

$ xtrabackup --prepare --rebuild-indexes /data/backups/

输出：

[01] Checking if there are indexes to rebuild in table sakila/city (space id: 9)

[01]   Found index idx\_fk\_country\_id

[01]   Rebuilding 1 index(es).

[01] Checking if there are indexes to rebuild in table sakila/country (space id: 10)

[01] Checking if there are indexes to rebuild in table sakila/customer (space id: 11)

[01]   Found index idx\_fk\_store\_id

[01]   Found index idx\_fk\_address\_id

[01]   Found index idx\_last\_name

[01]   Rebuilding 3 index(es).

使用--rebuild-threads指定重建的线程数，加快重建速度：

$ xtrabackup --prepare --rebuild-indexes --rebuild-threads=16 /data/backups/

输出：

Starting 16 threads to rebuild indexes.

[09] Checking if there are indexes to rebuild in table sakila/city (space id: 9)

[09]   Found index idx\_fk\_country\_id

[10] Checking if there are indexes to rebuild in table sakila/country (space id: 10)

[11] Checking if there are indexes to rebuild in table sakila/customer (space id: 11)

[11]   Found index idx\_fk\_store\_id

[11]   Found index idx\_fk\_address\_id

[11]   Found index idx\_last\_name

[11]   Rebuilding 3 index(es).

对于增量备份的应用可以先不重建索引，在应用最后一个差异备份的时候使用—rebuild-index来创建索引，每次都应用都重建索引太花时间。

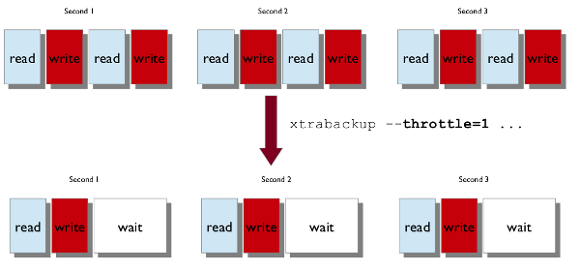
**3.2.7.3 备份还原**

使用命令rsync或者cp来还原备份，和全备的还原一样，请看：[3.2.3.3还原全备](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_3.2.3.3_还原全备)

**3.2.8 其他功能**

**3.2.8.1 节流备份**

--throttle用来控制每秒io次数，一次io，1MB。



如果在backup模式下，这个选项用来控制读写对的每秒次数。

默认不会节流，xtrabackup会读写是尽量快的方式。

**3.2.8.2 使用脚本调用xtrabackup来执行备份**

最典型的例子innobackupex，innobackupex是perl脚本调用xtrabackup来执行备份。

具体看：[Scripting Backups With xtrabackup](http://www.percona.com/doc/percona-xtrabackup/2.2/xtrabackup_bin/scripting_backups_xbk.html)

**3.2.8.3 分析表统计信息**

具体查看：[Analyzing Table Statistics](http://www.percona.com/doc/percona-xtrabackup/2.2/xtrabackup_bin/analyzing_table_statistics.html)

**3.2.8.4 使用binary log**

xtrabackup提取了inoodb的事务日志中提交事务，对于到binary log的位置。使用这个位置可以启动一个新的复制slave或者恢复一个时间点备份。

如果备份是一个来至于binary log启动的日志，xtrabackup会创建一个文件xtrabackup\_binlog\_info里面包含了，binary log文件名和位置。信息也会写在xtrabackup\_binlog\_pos\_innodb，这个文件只会在只有xtradb或者innodb情况下才会准确。其他情况下应该使用xtrabackup\_binlog\_info。

时间点恢复，和innobackupex的一样可以查看 [3.1.7.6时间点还原](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_3.1.7.6_时间点还原)

关于如何还原一个slave可以查看下面的内容：[4.3.1 六个步骤安装一个Slave](http://www.cnblogs.com/Amaranthus/archive/2014/08/19/3922570.html#_4.3.1_六个步骤安装一个slave)

**3.2.8.5 还原单个表**

在mariadb 10.0中可以直接通过ibd文件导入表。

**导出表**

先查找是否有这个文件存在

$ find /data/backups/mysql/ -name export\_test.\*

/data/backups/mysql/test/export\_test.ibd

然后导出表

$ xtrabackup --prepare --export --target-dir=/data/backups/mysql/

会产生exp文件

$ find /data/backups/mysql/ -name export\_test.\*

/data/backups/mysql/test/export\_test.exp

/data/backups/mysql/test/export\_test.ibd

/data/backups/mysql/test/export\_test.cfg

注意：mysql使用cfg文件，这个文件包含了innodb字典dump。这个格式和exp文件的不同，exp文件用于xtradb。严格来说cfg在mysql 5.6和percona 5.6之后是可以不用的，如果存在cfg文件，那么innodb会通过cfg文件做schema验证 。

**导入表**

导入表，在percona server使用xtradb，需要设置innodb\_import\_table\_from\_xtrabackup设置为可用，mysql5.6只要表结构一样都可以导入。

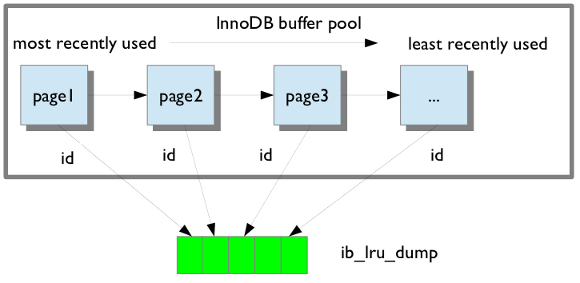
1.执行alter table discard tablespace

2.复制上一步生成的文件到对于的数据库目录

3.执行alter table import tablespace导入

**3.2.8.6 LRU dump备份**

这个功能减少了服务warm up的时间，在重启的时候直接导入ib\_lru\_dump文件中的数据，在备份的时候会自动备份。



如果my.cnf配置了，percona server启动了innodb\_buffer\_pool\_resotre\_at\_startup=1那么这个功能会自动启动。

这个功能在mariadb中有类似的功能：[**XtraDB/InnoDB Server System Variables**](https://mariadb.com/kb/en/xtradbinnodb-server-system-variables/)

**3.2.8.7 xtrabackup的限制**

1.在32位的系统下如果xtrabackup\_logfile大于4gb那么--prepare会报错

2.在第一次执行--prepare的时候不会生成ib\_logfile\*

3.xtrabackup只复制数据文件和日志，不会复制表定义，frm文件。

4.xtrabackup不支持--set-variable这种格式设置my.cnf

**3.2.8.8 References**

具体看：[The **xtrabackup** Option Reference](http://www.percona.com/doc/percona-xtrabackup/2.2/xtrabackup_bin/xbk_option_reference.html)

**3.2.9 xbstream**

具体看: [**xbstream**](http://www.percona.com/doc/percona-xtrabackup/2.2/xbstream/xbstream.html)

**3.2.10 xbcrypt**

具体看：[**xbcrypt**](http://www.percona.com/doc/percona-xtrabackup/2.2/xbcrypt/xbcrypt.html)

**3.2.11 Xtrabackup原理**

xtrabackup是基于innodb的crash恢复功能。复制innodb数据文件，但是数据是不一致的，然后使用crash恢复让数据文件一直。

当innodb启动会去检查数据文件和日志文件，然后重做已提交事务，执行未提交事务。

xtrabackup记下LSN，然后启动，复制数据文件。同时xtrabackup启动一个后台进程用来监控日志文件，然后复制修改，这个进程在备份期间一直是运行的，因为日志文件时回绕的，避免数据被覆盖无法恢复。直到备份完成。

第二阶段就是预备阶段，xtrabackup通过执行crash恢复，应用日志文件到数据文件上。这个过程在xtrabackup中实现。innobackupex增加了功能，会对myisam和.frm进行复制。innobackupex启动xtrabackup，等待复制innodb结束，然后执行FLUSH TABLES WITH READ LOCK，停止对mysql数据的修改。复制非innodb引擎表，知道复制完成，然后释放锁。

这样在prepare阶段后，innodb和非innodb相互保持了一致性。innodb会一直redo，直到备份完成。这个时间刚刚好和FLUSH TABLES WITH READ LOCK时间一直，所以innodb和非innodb是保持同步的。

**4 如何使用和案例(How-tos and Recipes)**

**4.1 innobackupex案例**

**4.1.1 本地全备(备份，预备，还原)**

**创建备份**

$ innobackupex /data/backups

备份完成

100313 02:43:07  innobackupex: completed OK!

**预备备份**

使用--apply-log来预备，使用--user-memory来加快预备速度

$ innobackupex --use-memory=4G --apply-log /data/backups/2010-03-13\_02-42-44/

完成

100313 02:51:02  innobackupex: completed OK!

**还原备份**

使用--copy-back来还原备份

innobackupex --copy-back /data/backups/2010-03-13\_02-42-44/

## Use chmod to correct the permissions, if necessary!

还原的位子由my.cnf中的datadir决定。

注意修改文件所有者

$ chown -R mysql:mysql /var/lib/mysql

**4.1.2 使用Stream备份**

**tar使用例子**

是用留备份归档到文件 ‘backup.tar’

$ innobackupex --stream=tar ./ > backup.tar

压缩归档文件

$ innobackupex --stream=tar ./ | gzip - > backup.tar.gz

加密备份

$ innobackupex --stream=tar . | gzip - | openssl des3 -salt -k "password" > backup.tar.gz.des3

把备份复制到远程

$ innobackupex --stream=tar ./ | ssh user@desthost "cat - > /data/backups/backup.tar"

使用netcat复制到远程

## On the destination host:

$ nc -l 9999 | cat - > /data/backups/backup.tar

## On the source host:

$ innobackupex --stream=tar ./ | nc desthost 9999

和上面一样只是一样完成

$ ssh user@desthost "( nc -l 9999 > /data/backups/backup.tar & )" \

&& innobackupex --stream=tar ./  |  nc desthost 9999

限制传输速度为10MB/s 需要pv工具。可以通过apt-get 安装

$ innobackupex --stream=tar ./ | pv -q -L10m \

| ssh user@desthost "cat - > /data/backups/backup.tar"

在备份的时候计算checksum

## On the destination host:

$ nc -l 9999 | tee >(sha1sum > destination\_checksum) > /data/backups/backup.tar

## On the source host:

$ innobackupex --stream=tar ./ | tee >(sha1sum > source\_checksum) | nc desthost 9999

## compare the checksums

## On the source host:

$ cat source\_checksum

65e4f916a49c1f216e0887ce54cf59bf3934dbad  -

## On the destination host:

$ destination\_checksum

65e4f916a49c1f216e0887ce54cf59bf3934dbad  -

[**xbstream**](http://www.percona.com/doc/percona-xtrabackup/2.2/glossary.html#term-xbstream)**使用例子**

备份并归档为‘backup.xbstream

innobackupex --stream=xbstream ./ > backup.xbstream

使用压缩归档

innobackupex --stream=xbstream --compress ./ > backup.xbstream

解包

xbstream -x <  backup.xbstream

把备份发送到其他目录

innobackupex --compress --stream=xbstream ./ | ssh user@otherhost "xbstream -x"

并发压缩归档

innobackupex --compress --compress-threads=8 --stream=xbstream --parallel=4 ./ > backup.xbstream

**4.1.3 创建增量备份**

**创建备份**

先创建一个全备：

innobackupex --user=USER --password=PASSWORD /path/to/backup/dir/

全备会生成一个时间戳的子目录，备份在子目录里，如/path/to/backup/dir/2011-12-24\_23-01-00/，并记为$FULLBACK

创建增量备份

innobackupex --incremental /path/to/inc/dir \

  --incremental-basedir=$FULLBACKUP --user=USER --password=PASSWORD

生成的目录为：/path/to/inc/dir/2011-12-25\_00-01-00/并记为$INCREMENTALBACKUP

**预备备份**

innobackupex --apply-log --redo-only $FULLBACKUP \

 --use-memory=1G --user=USER --password=PASSWORD

--user-memory可以加速预备速度。

输出：

111225 01:10:12 InnoDB: Shutdown completed; log sequence number 91514213

然后应用增量

innobackupex --apply-log --redo-only $FULLBACKUP

 --incremental-dir=$INCREMENTALBACKUP

 --use-memory=1G --user=DVADER --password=D4RKS1D3

因为是应用到 $FULLBACK下的，所以不再增量备份文件夹下。

如果还原多个增量备份，但是忘记了备份顺序可以查看xtrabackup\_checkpoint文件

如：

backup\_type = full-backuped

from\_lsn =0

to\_lsn =1291135

backup\_type = incremental

from\_lsn =1291135

to\_lsn =1291340

一旦都预备好之后，就可以回滚未提交事务，然后还原备份使用了

innobackupex-1.5.1 --apply-log $FULLBACKUP --use-memory=1G \

  --user=$USERNAME --password=$PASSWORD

**4.1.4 创建压缩备份**

**备份**

带--compress创建压缩备份

$ innobackupex --compress /data/backup

如果想要加快速度，可以使用--compress-threads加速

$ innobackupex --compress --compress-threads=4 /data/backup

输出：

...

[01] Compressing ./imdb/comp\_cast\_type.ibd to /data/backup/2013-08-01\_11-24-04/./imdb/comp\_cast\_type.ibd.qp

[01]        ...done

[01] Compressing ./imdb/aka\_name.ibd to /data/backup/2013-08-01\_11-24-04/./imdb/aka\_name.ibd.qp

[01]        ...done

...

130801 11:50:24  innobackupex: completed OK

**预备**

在预备之前先要使用qpress解压

$ for bf in `find . -iname "\*\.qp"`; do qpress -d $bf $(dirname $bf) && rm $bf; done

在xtrabackup2.1.4之后也可以使用--decompress解压

$ innobackupex --decompress /data/backup/2013-08-01\_11-24-04/

这个选项加压文件原来的文件会被替换为解压后的文件。

注意：使用--decompress需要安装qpress，并且--parallel可以和--decompress一起使用，加速解压缩。

然后使用--apply-log预备

$ innobackupex --apply-log /data/backup/2013-08-01\_11-24-04/

**还原**

使用--copy-back还原数据库

$ innobackupex --copy-back /data/backups/2013-08-01\_11-24-04/

--copy-back读取复制到my.cnf中的datadir的值

之后修改文件的所有者

$ chown -R mysql:mysql /var/lib/mysql

之后再启动服务

**4.1.5 备份还原独立分区**

xtrabackup可以部分备份，这个也让独立区的备份还原变成可能，只要启动innodb\_file\_per\_table。

先创建一个分区表

create table t.tb\_part (id int ,v int) partition by hash(id) partitions 4;

然后插入数据

insert t.tb\_part values(1,1),(2,1),(3,1),(4,1);

这样保证每个分区都有1条记录。

**4.1.5.1 创建备份**

使用innobackupex --include进行备份，还有很多其他方法进行部分备份：

innobackupex --user=root  --include='^t[.]tb\_part' /home/tiansign/mysql\_backup

**4.1.5.2 预备备份**

和独立表还原类似，使用--export进行预备

innobackupex --apply-log --export /home/tiansign/mysql\_backup/2014-08-19\_23-25-55/

ls

tb\_part.frm       tb\_part#P#p0.ibd  tb\_part#P#p2.cfg  tb\_part#P#p3.exp

tb\_part.par       tb\_part#P#p1.cfg  tb\_part#P#p2.exp  tb\_part#P#p3.ibd

tb\_part#P#p0.cfg  tb\_part#P#p1.exp  tb\_part#P#p2.ibd

tb\_part#P#p0.exp  tb\_part#P#p1.ibd  tb\_part#P#p3.cfg

有类似如下输出

xtrabackup: export option is specified.

xtrabackup: export metadata of table 't/tb\_part#P#p0' to file `./t/tb\_part#P#p0.exp` (1 indexes)

xtrabackup:     name=GEN\_CLUST\_INDEX, id.low=58, page=3

xtrabackup: export metadata of table 't/tb\_part#P#p2' to file `./t/tb\_part#P#p2.exp` (1 indexes)

xtrabackup:     name=GEN\_CLUST\_INDEX, id.low=60, page=3

xtrabackup: export metadata of table 't/tb\_part#P#p3' to file `./t/tb\_part#P#p3.exp` (1 indexes)

xtrabackup:     name=GEN\_CLUST\_INDEX, id.low=61, page=3

xtrabackup: export metadata of table 't/tb\_part#P#p1' to file `./t/tb\_part#P#p1.exp` (1 indexes)

xtrabackup:     name=GEN\_CLUST\_INDEX, id.low=59, page=3

**4.1.5.3 从备份还原**

这里主要介绍maridb 10.0的方法，也适用于mysql5.6

首先创建表结构：

create table test.tb\_part (id int ,v int) partition by hash(id) partitions 4;

create table test.p3 (id int ,v int) ;

然后discard

alter table test.p3 discard tablespace;

复制cfg和ibd到数据库目录下（在mariadb 10.0之后可以不用cfg文件，直接使用ibd文件）

cp tb\_part#P#p3.ibd /home/db/test/p3.ibd

然后修改所有者

chown mysql:mysql /home/db/test/p3.ibd

最后 import 表

alter table test.p3 import tablespace;

最后通过exchange partition方式把表数据交换到分区表中

alter table test.tb\_part exchange partition p3 with table test.p3;

验证

select \*from test.tb\_part;

**4.2 xtrabackup案例**

具体看：[Recipes for xtrabackup](http://www.percona.com/doc/percona-xtrabackup/2.2/how-tos.html#recipes-for-xtrabackup)

**4.3 其他案例**

**4.3.1 六个步骤安装一个slave**

**需要的东西**

1.TheMaster服务器

       a.需要安装mysql并且能公国tcp/ip访问

       b.配置了SSH服务

       c.有个系统账号，并有一些权限

       d.有个数据库账号，也有相应的权限

       e.服务的binlog启动，并且server-id为1

2.TheSlave,另一个系统安装了mysql，其他都和TheMaster一样，但是server-id要为2

3.2个系统都要安装xtrabackup

**步骤1：创建一个备份并准备**

TheMaster$ innobackupex --user=yourDBuser --password=MaGiCdB1 /path/to/backupdir

完成之后预备

TheMaster$ innobackupex --user=yourDBuser --password=MaGiCdB1 /

           --apply-log /path/to/backupdir/$TIMESTAMP/

如果你有特定的配置文件，那么通过--defaults-file=XXXX/my.cnf指定。默认使用备份目录下的backup-my.cnf。

**步骤2：把备份复制到TheSlave**

使用rsync来同步备份文件

TheMaster$ rsync -avprP -e ssh /path/to/backupdir/$TIMESTAMP TheSlave:/path/to/mysql/

关闭，mysql服务，并保存原先的数据库

TheSlave$ mv /path/to/mysql/datadir /path/to/mysql/datadir\_bak

然后把备份复制到mysql配置的datadir下

TheSlave$ mv /path/to/mysql/$TIMESTAMP /path/to/mysql/datadir

修改文件所有者

TheSlave$ chown mysql:mysql /path/to/mysql/datadir

**步骤3：配置Master**

配置slave，master连接账号

TheMaster|mysql>**GRANT** REPLICATION SLAVE **ON**\*.\*  **TO**'repl'@'$slaveip'

 IDENTIFIED **BY**'$slavepass';

保证TheSlave可以通过这个账号连接到TheMaster

TheSlave$ mysql --host=TheMaster --user=repl --password=$slavepass

验证权限

mysql> SHOW GRANTS;

**步骤4：配置Slave**

先复制master 上的配置文件

TheSlave$ scp user@TheMaster:/etc/mysql/my.cnf /etc/mysql/my.cnf

然后修改server-id=2

server-id=2

启动slave上的服务

**步骤5：启动复制**

先查看xtrabackup\_binlog\_info来确定binary log的位置

TheSlave$ cat /var/lib/mysql/xtrabackup\_binlog\_info

TheMaster-bin.000001     481

使用CHANGE MASTER，账号密码使用刚才在master中申请的账号。

TheSlave|mysql>**CHANGE** MASTER **TO**

                MASTER\_HOST='$masterip',

                MASTER\_USER='repl',

                MASTER\_PASSWORD='$slavepass',

                MASTER\_LOG\_FILE='TheMaster-bin.000001',

                MASTER\_LOG\_POS=481;

然后启动slave

TheSlave|mysql> START SLAVE;

**步骤6：检查**

TheSlave|mysql> SHOW SLAVE STATUS \G

         ...

         Slave\_IO\_Running: Yes

         Slave\_SQL\_Running: Yes

         ...

         Seconds\_Behind\_Master: 13

         ...

io和SQL都要运行，Seconds\_behind\_master是现在slave执行的语句在master上是13秒之前的数据。这个是master和slave之间的延迟。

**4.3.2 再增加一个slave**

过程基本上和上面类似

1.备份slave，要带上--slave-info，带上这个选项会产生一个xtrabackup\_slave\_info其中的master的binary log和位置都记录在这个文件上

TheSlave$ innobackupex --user=yourDBuser --password=MaGiCiGaM /

          --slave-info /path/to/backupdir

2.预备，增加了use-memory来加快预备速度

TheSlave$ innobackupex --apply-log --use-memory=2G /path/to/backupdir/$TIMESTAMP/

3.复制到是slave

rsync -avprP -e ssh /path/to/backupdir/$TIMESTAMP TheNewSlave:/path/to/mysql/datadir

4.在master上，再创建一个账号，当然也可以使用同一个账号，只要能连接上就可以

heMaster|mysql>**GRANT** REPLICATION SLAVE **ON**\*.\*  **TO**'repl'@'$newslaveip'

             IDENTIFIED **BY**'$slavepass';

5.复制配置文件，在启动的时候不启动复制，并设置server-id为3

TheNEWSlave$ scp user@TheSlave:/etc/mysql/my.cnf /etc/mysql/my.cnf

修改配置文件

Skip-slave-start

server-id=3

启动slave的服务

6.通过xtrabackup\_slave\_info获取master的日志名和位置

TheNEWSlave|mysql>**CHANGE** MASTER **TO**

                   MASTER\_HOST='$masterip',

                   MASTER\_USER='repl',

                   MASTER\_PASSWORD='$slavepass',

                   MASTER\_LOG\_FILE='TheMaster-bin.000001',

                   MASTER\_LOG\_POS=481;

启动slave

TheSlave|mysql> START SLAVE;

**4.3.3 使用复制和pt-checksum验证备份**

具体看：[Verifying Backups with replication and pt-checksum](http://www.percona.com/doc/percona-xtrabackup/2.2/howtos/backup_verification.html)

**4.3.4 如何创建基于GTID的Slave**

具体看：[How to create a new (or repair a broken) GTID based slave](http://www.percona.com/doc/percona-xtrabackup/2.2/howtos/recipes_ibkx_gtid.html)

**4.4辅助工具手册**

具体看：[Auxiliary Guides](http://www.percona.com/doc/percona-xtrabackup/2.2/how-tos.html#aux-guides)

**参考**

[Xtrabackup安装及使用](http://www.cnblogs.com/cosiray/archive/2012/03/02/2376595.html)

[Percona Xtrabackup - Documentation](http://www.percona.com/doc/percona-xtrabackup/2.2/index.html)

[xtrabackup原理及实施](http://www.baidu-ops.com/2013/05/26/xtrabackup)