<https://www.cnblogs.com/mysql-dba/p/4915140.html>

<http://dev.yesky.com/281/35291281.shtml>

　每一种逻辑错误发生的概率都极低，但是当多种可能性叠加的时候，小概率事件就 放大成很大的安全隐患，这时候备份的必要性就凸显了。那么在众多的[MySQL](http://mydown.yesky.com/soft/32/32183.shtml)备份方式中，哪一种才是适合我们的呢？

　　UPDATE or DELETE whitout where…

　　table was DROPPed accidentally…

　　INNODB was corrupt…

　　entire datacenter loses power…

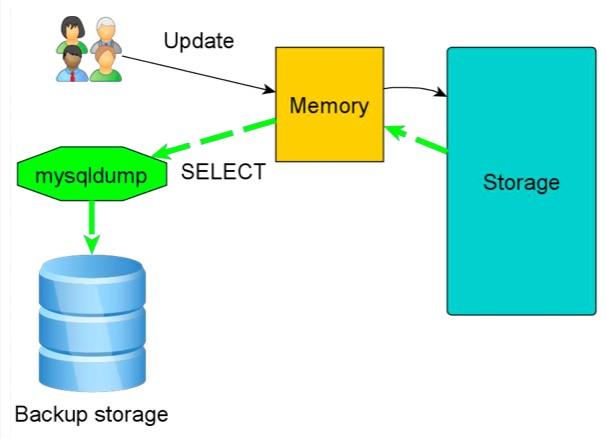
　　从数据安全的角度来说，[服务器](http://product.yesky.com/service/)磁盘都会做raid，MySQL本身也有主从、drbd等容灾机制，但它们都无法完全取代备份。容灾和高可用能帮我们 有效的应对物理的、硬件的、机械的故障，而对我们犯下的逻辑错误却无能为力。每一种逻辑错误发生的概率都极低，但是当多种可能性叠加的时候，小概率事件就 放大成很大的安全隐患，这时候备份的必要性就凸显了。那么在众多的MySQL备份方式中，哪一种才是适合我们的呢？

　　**常见的备份方式**

　　MySQL本身为我们提供了mysqldump、mysqlbinlog远程备份工具，percona也为我们提供了强大的Xtrabackup， 加上开源的mydumper，还有基于主从同步的延迟备份、从库冷备等方式，以及基于文件系统快照的备份，其实选择已经多到眼花缭乱。而备份本身是为了恢 复，所以能够让我们在出现故障后迅速、准确恢复的备份方式，就是最适合我们的，当然，同时能够省钱、省事，那就非常完美。下面就我理解的几种备份工具进行 一些比较，探讨下它们各自的适用场景。

　　**1. mysqldump & mydumper**

　　mysqldump是最简单的逻辑备份方式。在备份myisam表的时候，如果要得到一致的数据，就需要锁表，简单而粗暴。而在备份innodb表 的时候，加上–master-data=1 –single-transaction 选项，在事务开始时刻，记录下binlog pos点，然后利用mvcc来获取一致的数据，由于是一个长事务，在写入和更新量很大的数据库上，将产生非常多的undo，显著影响性能，所以要慎用。



　　优点：简单，可针对单表备份，在全量导出表结构的时候尤其有用。

　　缺点：简单粗暴，单线程，备份慢而且恢复慢，跨IDC有可能遇到时区问题。

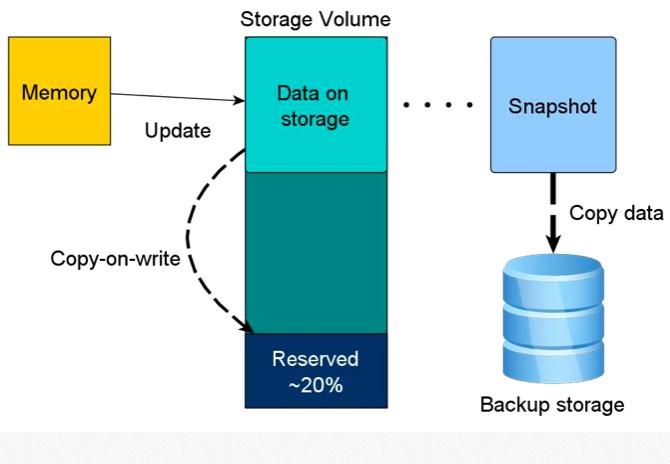
　　mydumper是mysqldump的加强版。相比mysqldump：

　　内置支持压缩，可以节省2-4倍的存储空间。

　　支持并行备份和恢复，因此速度比mysqldump快很多，但是由于是逻辑备份，仍不是很快。

　　**2. 基于文件系统的快照**

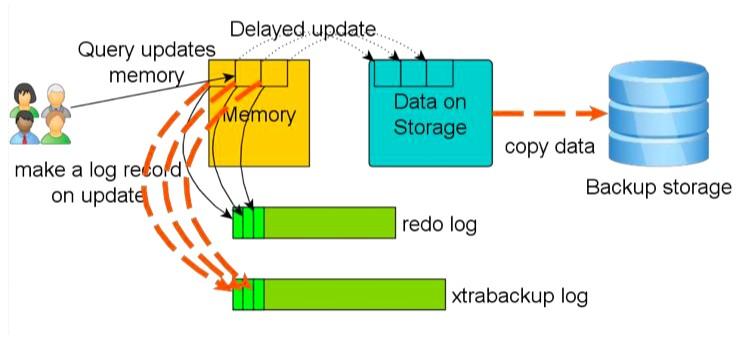
　　基于文件系统的快照，是物理备份的一种。在备份前需要进行一些复杂的设置，在备份开始时刻获得快照并记录下binlog pos点，然后采用类似copy-on-write的方式，把快照进行转储。转储快照本身会消耗一定的IO资源，而且在写入压力较大的实例上，保存被更改 数据块的前印象也会消耗IO，最终表现为整体性能的下降。而且服务器还要为copy-on-write快照预留较多的磁盘空间，这本身对资源也是一种浪 费。因此这种备份方式我们使用的不多。



　　**3. Xtrabackup**

　　这或许是最为广泛的备份方式。percona之所以家喻户晓，Xtrabackup应该功不可没。它实际上是物理备份+逻辑备份的组合。在备份 innodb表的时候，它拷贝ibd文件，并一刻不停的监视redo log的变化，append到自己的事务日志文件。在拷贝ibd文件过程中，ibd文件本身可能被写”花”，这都不是问题，因为在拷贝完成后的第一个 prepare阶段，Xtrabackup采用类似于innodb崩溃恢复的方法，把数据文件恢复到与日志文件一致的状态，并把未提交的事务回滚。如果同 时需要备份myisam表以及innodb表结构等文件，那么就需要用flush tables with lock来获得全局锁，开始拷贝这些不再变化的文件，同时获得binlog位置，拷贝结束后释放锁，也停止对redo log的监视。

　　**它的工作原理如下**：



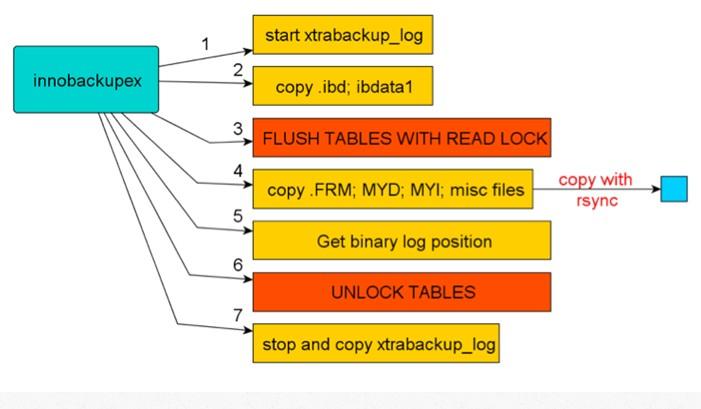
　　由于mysql中不可避免的含有myisam表，同时innobackup并不备份表结构等文件，因此想要完整的备份mysql实例，就少不了要执行 flush tables with read lock，而这个语句会被任何查询(包括select)阻塞，在阻塞过程中，它又反过来阻塞任何查询(包括select)。如果碰巧备份实例上有长查询先 于flush tables with read lock执行，数据库就会hang住。而当flush tables with read lock获得全局锁后，虽然查询可以执行，但是仍会阻塞更新，所以，我们希望flush tables with read lock从发起到结束，持续的时间越短越好。

　　**为了解决这个问题，有两种比较有效的方法**：

　　1. 尽量不用myisam表。

　　2. Xtrabackup增加了–rsync选项，通过两次rsync来减少持有全局锁的时间。

　　优化后的备份过程如下：



　　优点：在线热备，全备+增备+流备，支持限速，支持压缩，支持加密。

　　缺点：需要获取全局锁，如果遇到长查询，等待时间将不可控，因此要做好监控，必要时杀死长查询或自杀;遇到超大的实例，备份过程较长，redo log太大会影响恢复速度，这种情况下最好采用延迟备份。

　　**4. mysqlbinlog 5.6**

　　上述所有的备份方式，都只能把数据库恢复到备份的某个时间点：mysqldump和mydumper，以及snapshot是备份开始的时间 点;Xtrabackup是备份结束的时间点。要想实现point in time的恢复，还必须备份binlog。同时binlog也是实现增备的宝贵资源。

　　幸运的是，mysql 5.6为我们提供了远程备份binlog的选项：

　　mysqlbinlog --raw --read-from-remote-server --stop-never

　　它会伪装成mysql从库，从远程获取binlog然后进行转储。这对线上主库容量不够无法保存较多binlog的场景非常有用。但是，它毕竟不像 真正的mysql从库实例，状态监控和同步都需要单独部署。因此个人觉得采用blackhole来备份全量的binlog是更好的选择。笔者曾经实现过一 个自动搭建blackhole从库的工具，稍加修改，就可以完美搭建出blackhole从库。一旦同步起来，基本一劳永逸，很少出问题，主从切换的时候跟着切了就行。

　　**提示：**

　　不要小看binlog的备份。当5.6的多线程复制大规模使用后，从库追赶主库命令点的耗时将被极大缩短，这样我们把每天一次的全量备份改为每3 天一次、甚至每周一次的全量备份，和持续的binlog增量备份。遇到故障需要恢复数据的时候，重放3、5天的binlog也是极快的。降低备份频率最直 接的好处是，省钱、省事。

　　blackhole对于备份binlog是极好的。一方面可以长久的备份binlog用于恢复数据库，另一方面，在其上配置半同步复制，可以有效防止主库的binlog丢失。

　　**总结**

　　备份方式各有千秋，而对我们来说，面对数千实例，选择合适的备份工具来实现统一配置、统一规划，构建智能调度的备份云平台才是王道。毕竟，多种备份方式共存的运维成本是不容忽视的。

　　从使用经验来看，用Xtrabackup全备数据，用blackhole增备binlog，并定期对备份数据的有效性进行验证，是当下比较好的选择。