**MySQL存储引擎转换方案**

编 写 人：李建军

部 门：数据支持

完成时间：2017.06

Email：[lijianjun@meilele.com](mailto:lijianjun@meilele.com)

版本信息

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 时间 | 提交人 | 描述 |
| 1.0 | 2017.06 | 李建军 | 创建文档 |

1. InnoDB

## 1.1InnoDB介绍

InnoDB，是MySQL的数据库引擎之一，为MySQL AB发布binary的标准之一。

InnoDB由Innobase Oy公司所开发，2006年五月时由甲骨文公司并购。与传统的MyISAM相比，InnoDB的最大特色就是支持了ACID兼容的事务（Transaction）功能。InnoDB是为处理巨大。数据量时的最大性能设计。InnoDB存储引擎完全与MySQL服务器整合，InnoDB存储引擎为在主内存中缓存数据和索引而维持它自己的缓冲池。InnoDB存储它的表&索引在一个表空间中，表空间可以包含数个文件（或原始磁盘分区）。这与MyISAM表不同，比如在MyISAM表中每个表被存在分离的文件中。

InnoDB 给 MySQL 提供了具有事务(transaction)、回滚(rollback)和崩溃修复能力

crash recovery capabilities)、多版本并发控制(multi-versioned concurrency control)的事务安全(transaction-safe (ACID compliant))型表。InnoDB 提供了行级锁(locking on row level)，提供与 Oracle 类似的不加锁读取(non-locking read in SELECTs)。InnoDB锁定在行级并且也在SELECT语句提供一个Oracle风格一致的非锁定读。这些特色增加了多用户部署和性能。没有在InnoDB中扩大锁定的需要，因为在InnoDB中行级锁定适合非常小的空间。InnoDB也支持FOREIGN KEY强制。在SQL查询中，你可以自由地将InnoDB类型的表与其它MySQL的表的类型混合起来，甚至在同一个查询中也可以混合。这些特性均提高了多用户并发操作的性能表现。在InnoDB表中不需要扩大锁定(lock escalation)，因为 InnoDB 的行级锁定(row level locks)适宜非常小的空间。InnoDB 是 MySQL 上第一个提供外键约束(FOREIGN KEY constraints)的表引擎。

在技术上，InnoDB 是一套放在 MySQL后台的完整数据库系统，InnoDB 在主内存

中建立其专用的缓冲池用于高速缓冲数据和索引。

InnoDB 把数据和索引存放在表空间里，可能包含多个文件，这与其它的不一样，举

例来说，在 MyISAM 中，表被存放在单独的文件中。InnoDB 表的大小只受限于操作系统的文件大小，可也可以每个表使用各自独立的表空间，只需要启用选innodb\_file\_per\_table。

## 1.2InnoDB的特点

1.2.1：InnoDB支持行级锁(row level lock),InnoDB最小锁粒度为行，这在极大的

提高MySQL在高并发环境里的应用。同时InnoDB也支持next-key lock也就是GAP间隙锁来防止出现幻读。

1.2.2提供事务ACID支持。但必须显示的以start/begin来开始一个事务。MVCC多

版本来保证事务的隔离性，redo和undo来保证事务的一致性与持久化。

1.2.3提供更优的崩溃恢复能力。不同于MyISAM的repair操作。

1.2.4Insert Buffer插入缓冲，对于非聚集索引插入和更新的时候，首先判断非聚

集索引是否存在于缓存中，如果存在则插入，否则先插入到一个插入缓存中，然后以一定的频率执行插入和合并非聚集索引的叶子节点的操作，来提高插入性能。（使用的2个条件，非聚集索引和非唯一索引）

1.2.5Doubler Write双写，是innodb内存中的double write buffer区域2M，同

时共享表空间里连续的128个页，即2个区也为1+1 M。首先会把数据写入共享表空间的连续的128页2M，完成后再立即刷盘，随机的写入到磁盘上相应的位置。这样做能保证在数据写入时崩溃后的恢复。

1.2.6自适应哈希，INNODB不支持HASH索引，但INNODB会监控表上的索引使用，

如果发现建立HASH索引可以提高效率的时候，则自动建立hash索引。

1.2.7支持外键。

## 1.3InnoDB相对于MyISAM的一些缺点

1.3.1InnoDB双写特性会带来一些性能损失，当然这点性能损失和其带来的优势对比，我们都乐于接受。

1.3.2InnoDB没有类似于MyISAM记录表记录行数的功能，因此在COUNT(\*)会执行全表扫描，因而在实现类似功能的时候速度会比较慢。

1.3.3InnoDB不支持全文索引，MyISAM支持。(5.6开始innodb也支持全文索引，因词为单位)

1.3.4MyISAM在varchar列索引支持前缀压缩，而INNODB不支持此特性，因此通常INNODB占用的空间会大于MyISAM。

1.3.5MyISAM的联合主键索引中，可以在非第一列(非第一个字段)使用自增列，而InnoDB的主键索引中包含自增列时，必须在最前面;这个特性在discuz论坛中，被设计用于“抢楼”功能，因此，若有类似的业务，则无法将该表从MyISAM转成InnoDB;

## 2MyISAM引擎转为InnoDB引擎的方法

2.1导出的方法通过mysqldump导出，并更改相应的

engine=MyISAM替换为engine=InnoDB,然后再导入到数据库中。此种方法需要停机或者停实例的方式来完成。

2.2在线更改的方式。Alter table xxx engine = innodb;此种方式会锁表，影响数据库的使用。也可以在业务低估进行。或者先在从库上更改再切换。

2.3 在线更改的方式。借助pt工具来完成更改，此种方式通过新建表—》更改引擎—》导入原表数据—》删除原表-》重命名新表。此种方式不会锁表。（此种方式在使用online-change-schema 制定参数—charset=字符集，否则可能产生乱码）--alter 'engine = innodb' D=test,t=t -S /tmp/mysql6.sock --execute --charset=utf8mb4

2.4在实用pt工具的时候注意是否有依赖的外键检查使用参数：--alter-foreign-keys-method

2.4.1使用--set-vars来控制是否写日志

2.4.2需要注意pt工具集的相关限制，如原表上不能有触发器等！

## 3存储引擎转换实施过程建议

3.1查看整个数据库系统当前负载情况。根据负载情况决定新机器配置好加入环境还是从现有集群中从从库实施。

3.2根据上一步的配置，参考对业务的影响，优化配置参数。切换主从，直至完全变更。

3.3为了保证业务高可用，替换过程中建议采用逐步灰度替换方式。

## 4 信息收集

4.1评估由于存储引擎的变更带来的业务相关影响：

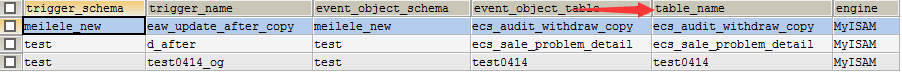
4.1.1首先查看现有业务中的count(\*)相关的业务，由于innodb在执行无条件的count(\*)时会全表扫描，效率低下，分析业务中相关的统计业务是否很高的实时性要求。一般情况下业务中都不有太严格的要求，如果要求很严格，有以下解决方法：a、创建额外的表专门用于统计相关计数，然后通过触发器达到更新；b联系DB架构考虑是否把这部分数据放在redis、memcache中。如果准确性要求不是那么高，那么可以单独放在另外的表，同事通过events来达到更新。

4.2 由于变更为innodb引擎后，5.6以前的版本都不支持全文索引，因此需要查看业务中是否有用到全文索引的语句。类似于 where match(column) aginst(‘key’)的语句。如果没有，则直接跳过。如现有业务有利用到全文索引的地方，这些表不能转到innodb，如果一定也要转到innodb，则需要升级到5.6版本。

4.3 需要运维同事配合收集服务器资源信息，可用磁盘空间、CPU、mem等常规信息。

4.4 由于innodb和myisam联合主键自增信息，需要收集相关信息。（下班后统计）

4.5 统计触发器信息。



4.5.1 event\_object\_schema下的相应表在更改的时候引擎的时候只能通过alter的方式而不能通过pt工具集。

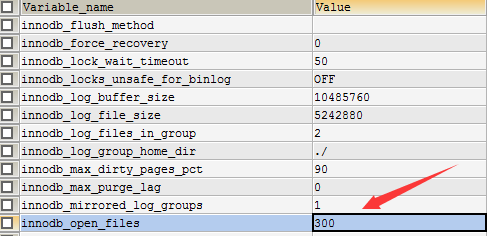
4.6 现有表引擎、占用空间等信息统计：

4.6.1 现有myisam表总数：4921个。Innodb表总数：546个。

4.6.2 现有数据库磁盘占用量。Myisam表总大小：

4.6.3 基于4.6.2所收集到的数据，我们在更改前必须要保证可用磁盘空间大于现有占用空间的2倍以上。

4.7 收集现有innodb相关配置信息：



4.7.1 类似于innodb\_open\_files、io\_threads等一些只读参数必须在服务器初始化的时候设置好。

4.8 数据访问模式，是否通过代理，中间件特性等。

4.9 高可用的架构以及高可用切换功能是否可用。

## 5 具体实施

5.1 实施方式有2种，一是在线更改，二是根据资源使用情况选择新申请服务器或者停掉其中一台从库，进行停机更改维护再加入到环境中。结合我们的实际情况，我们采用第2种方式进行变更。（新服务器的规格需要不低于现有服务器配置。）

5.2 备份现有数据库信息。并上传至用来变更的服务器。

5.3 新服务器上安装并设置好my.cnf相关信息，诸如 server\_id、innodb\_open\_file、innodb\_open\_tables、innodb\_file\_io\_threads等只读参数。有一部分诸如innodb\_sort\_buffer等可动态更改的参数，我们可以后期动态调整。

5.4 在新服务器上还原备份文件，加入测试环境。进行测试和压测。其中测试主要是验证引擎转换后业务正常性测试；压测主要是用于测试配置参数对业务环境的性能。

5.5 基于5.4的结果，我们把新服务器加入生产环境验证。如无问题逐步替换从库并观察数据库状态。如果此时出现异常，我们可以踢掉新加的服务器，从新开始5.4这一步。因为是新加服务器，所以我们此时踢掉新服务器并不会影响业务。

5.6 切换主从，最后我们更改主库。完成全部变更。