|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.6 5.7 原Online DDL | pt-online-schema-change  我们现在用的就是这种 | gh-ost  这种方式最近也做下调研 |
| 原理 | 1. 建立一个临时文件， 扫描原始表主键的所有数据页；  2. 用数据页中原始表的数据页生成B+Tree，存储到临时文件中；  3. state 2（日志）：生成临时文件的过程中， 将所有对原始表的操作 记录在一个日志文件（row log）中；  4. state 3（应用）：临时文件生成后， 将日志文件中的操作应用到临时文件，得到一个逻辑数据上与A表相同的数据文件；  5. state 4（替换）：用临时文件替换原始表的数据文件。 | 1. 创建一个跟源数据表一样的表结构，称为临时表; 2. 对临时表进行加字段； 3. 在原表中创建触发器，3个触发器分别对应insert,update,delete操作； 4. 以一定块大小从原表拷贝数据到临时表（把大事务拆分成小事务） 5. 数据拷贝完成之后 把源数据表命令为 old表，临时表命令为原始表； 6. 删除 old表，删除3个触发器。 |  |
| 使用限制 |  | 1. 原表必须要有主键或者唯一索引； 2. 原表上不能有触发器； 3. 使用前需要保证有足够的磁盘空间，因为 copy data需要一倍以上的磁盘空间 并且binlog格式为 row的情况下，写binlog需要额外的磁盘空间； 4. 在阿里RDS 上使用需要增加参数no-version-check。 5. 新增字段，NOT NULL必须要指定默认值 |  |
| 重要参数说明 | | | |
| 优点 | 1. Online DDL期间几乎不阻塞读写； | 1. 业界使用广泛 2. **主从延迟很小** 3. 负载不是很大 |  |
| 风险点和性能消耗 | 1. alter table 重建表空间方法都会扫描原表数据、构建临时文件，并把原表数据写入到临时文件;  2. 需要比原表数据一倍以上的磁盘空间;  3. 重建表过程中，很消耗IO和CPU资源;  4. **是主从延迟的杀手;**  5. 如果有慢查询，会有影响 | 1. 执行失败： 如果 max\_binlog\_cache\_size 缓存不下单个 binlog的大小，就会执行失败： execute failed: Multi-statement transaction required more than 'max\_binlog\_cache\_size' bytes of storage; 2. 导致主从中断：Multi-statement transaction required more than 'max\_binlog\_cache\_size' bytes of storage; 从库在应用relay log，生成的binlog大小通过 max\_binlog\_cache\_szie缓存不了； 3. **可能会导致死锁，回滚业务的SQL语句。**   4. 使用不合理，会导致业务插入慢（加上chunk-size参数，影响就会比较小）。 |  |
| 如何选择 | 1. 需要扫描原表的所有数据，并且写入到临时文件，成本高，不宜采用 2. 是主从延迟的杀手，不宜采用 3. 重建表中，很消耗I/O和CPU资源，不宜采用 |  |  |
| 使用场景 | 修改索引、外键、列名时，优先采用online ddl，并指定 ALGORITHM=INPLACE | 添加和删除列、索引、重建表空间 |  |
| 结论 | 无论哪种方式都选择的业务低峰期执行 | | |