原创

MySQL 5.7 特性：Online DDL

2020-09-14 阅读 1.4K

前言

DDL 一向是业务的痛点，尤其是对大型表的 DDL 操作，具有操作时间久，对性能影响大，可能影响业务正常使用等问题。

本文详细解释 MySQL DDL 的原理，以及尽可能减少 DDL 对业务的影响的办法。

MySQL DDL 的方法

MySQL 的 DDL 有很多种方法。

MySQL 本身自带三种方法，分别是：copy、inplace、instant。

> copy 算法为最古老的算法，在 MySQL 5.5 及以下为默认算法。
>
> 从 MySQL 5.6 开始，引入了 inplace 算法并且默认使用。inplace 算法还包含两种类型：rebuild-table 和 not-rebuild-table。MySQL 使用 inplace 算法时，会自动判断，能使用 not-rebuild-table 的情况下会尽量使用，不能的时候才会使用 rebuild-table。当 DDL 涉及到主键和全文索引相关的操作时，无法使用 not-rebuild-table，必须使用 rebuild-table。其他情况下都会使用 not-rebuild-table。
>
> 从 MySQL 8.0.12 开始，引入了 instant 算法并且默认使用。目前 instant 算法只支持增加列等少量 DDL 类型的操作，其他类型仍然会默认使用 inplace。

有一些第三方工具也可以实现 DDL 操作，最常见的是 percona 的 pt-online-schema-change 工具（简称为 pt-osc），和 github 的 gh-ost 工具，均支持 MySQL 5.5 以上的版本。

各类工具的对比

方法	copy	inplace not-rebuild-table	inplace rebuild-table	instant	pt-osc	gh-ost
DDL 过程中读取数据	允许	允许	允许	允许	允许	允许
DDL 过程中写入数据	不允许	允许	允许	允许	允许	允许
需要 MDL	需要	需要	需要	需要	需要	需要
需要额外空间	大	小	中	小	大	大
执行时间	非常长	短	非常长	非常短	长	长

目录

作者介绍

王文安@DBA

腾讯MySQL DBA

关注 专栏

文章	阅读量	获赞	作者排名
47	38.6K	647	1069

方法	copy	rebuild-table	rebuild-table	instant	osc	ost
IO 负载	大	小	大	非常小	非常大	大
导致主从同步延时	非常大	大	大	非常小	小	小
其他						支持临时暂停

一般情况下的建议：

如果使用的是 MySQL 5.5 或者 MySQL 5.6，推荐使用 gh-ost

如果使用的是 MySQL 5.7，索引等不涉及修改数据的操作，建议使用默认的 inplace 算法。如果涉及到修改数据（例如增加列），不关心主从同步延时的情况下使用默认的 inplace 算法，关心主从同步延时的情况下使用 gh-ost

如果使用的是 MySQL 8.0，推荐使用 MySQL 默认的算法设置，在语句不支持 instant 算法并且在意主从同步延时的情况下使用 gh-ost

各类工具的使用方法

copy

MySQL 5.5 及以下，直接正常 DDL 即可。

MySQL 5.6 及以上，如果希望使用 copy 算法，需要使用 `ALGORITHM=COPY` 指定算法类型，例如：

```
ALTER TABLE table1 ADD COLUMN column1 int ALGORITHM=COPY;
```

inplace

MySQL 5.7，直接正常 DDL 即可。

MySQL 8.0 及以上，如果希望使用 inplace 算法，需要使用 `ALGORITHM=INPLACE` 指定算法类型。

instant

MySQL 8.0 ，直接正常 DDL 即可。

pt-online-schema-change

比 gh-ost 落后很多，不推荐使用此工具。

gh-ost

参考其他的文章

MySQL DDL 的使用注意事项

MySQL 在大型表上的 DDL 会带来耗时较久、负载较高、额外空间占用、MDL、主从同步延时等情况。需要特别引起重视。

大部分情况在上面的性能对比表格已经描述，这里不再重复，这里着重讲一些重点问题：

DDL 的所需时间

DDL 的执行时间，和很多因素相关，如果需要比较精确的时间预估，建议在测试环境提前做测试。

可以新建一个测试实例，将备份数据导出到测试实例，执行 DDL 操作，判断执行时间，作为对线上执行的一个估计。但是请注意，该估计仍然可能不准确，因为线上实例

目录

MySQL 5.7 可以开启 DDL 监控，使用以下语句查看 DDL 执行进度：

```
SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED FROM performance_s
```

MySQL 自带的监控也是估算值，因此进可作参考。

负载

所有方式对大表做 DDL 都会增加负载，只是程度的不同，主要为 IO 的负载。如果是 IO 使用非常高的实例，建议在 IO 较小的时间段执行 DDL 操作。

额外空间占用

copy、inplace rebuild-table、gh-ost、pt-online-schema-change，都会将表完整复制一份出来再做 DDL 变更，因此会使用和原表空间一样大（甚至更大，如果是加列的操作的话）的额外空间，另外还会生成大量的临时日志。要特别注意剩余空间，确保空间充裕，不然可能导致 DDL 过程中磁盘写满。

主从同步延时

所有方式做 DDL 均会引发主从同步延时。其中 copy 和 inplace 算法，只有主完成了 DDL 操作之后，binlog 才会同步给从库，从库才能开始操作 DDL，从库操作完 DDL 之后才能开始操作其他语句，因此会造成巨大的（大概两倍 DDL 操作时间）的延时。其他方法产生的延时较小，但仍然可能有几秒的延时。

MDL

所有方式做 DDL 均会产生 MDL（metadata lock）。除了 copy 模式会有持续性的锁（DDL 的整个过程期间无法向该表写入任何数据）之外，其他方式的 MDL 均为短暂的锁。

除了 copy 模式之外的所有模式，MDL 如下：

1. 在 DDL 的开始阶段，申请该表的 EXCLUSIVE-MDL 锁，禁止读写
2. 降级 EXCLUSIVE-MDL 锁，允许读写
3. 在 DDL 的最终 COMMIT 阶段，升级 EXCLUSIVE-MDL 锁，禁止读写

其中的阶段一和阶段三，其 MDL 的持续时间都是非常短暂的，也就是申请到了 MDL 锁之后会在很快的时间（一般小于一秒）处理完成相关操作并释放锁，一般情况下是不会影响业务的。只有阶段二是真正在处理数据，持续时间一般较长。

但是，有可能出现在阶段一和阶段三，无法申请到 MDL 的情况。这是因为 MDL 和所有的读写语句都可能会产生冲突，如果是在申请 MDL 的时候，之前有读写的事务一直没有执行完成（或者执行完成之后一直没有 COMMIT），MDL 就会无法立刻申请到，这个时候，DDL 语句，**以及所有在该 DDL 语句之后的读写事务**，都会阻塞并等待之前的读写事务完成，导致整个实例处于不可用状态。这个时候 `SHOW PROCESSLIST` 看到的语句状态为 `waiting for metadata lock` 。

由于目前所有的 DDL 语句都会产生 MDL，无法避免，因此，在执行 DDL 操作期间，尽可能确保不要有未执行完成的长事务。如果发生了 `warting for metadata lock` 导致的阻塞，一般有以下三种处理方法：

1. 耐心等待之前的事务全部执行完成
2. 将之前未执行完成的事务全部 kill 掉
3. kill 掉 DDL 语句

其他

MySQL 的 inplace 算法虽然支持在 DDL 过程中间的读写，但是对写入的数据量有上限，不能超过 `innodb_online_alter_log_max_size` （默认为 128M）。如果超过上限可能导致执行失败。

MySQL DDL 的原理简析

copy 算法

目录

替换掉源表。这个算法主要被早期（<=5.5）版本所使用。

inplace 算法

从 5.6 开始，常用的 DDL 都默认使用这个算法。inplace 算法包含两类：inplace-no-rebuild 和 inplace-rebuild，两者的主要差异在于是否需要重建源表。

inplace 算法的操作阶段主要分为三个：

Prepare阶段：
- 创建新的临时 frm 文件(与 InnoDB 无关)。
- **持有 EXCLUSIVE-MDL 锁，禁止读写。**
- 根据 alter 类型，确定执行方式（copy，online-rebuild，online-not-rebuild）。
更新数据字典的内存对象。
- 分配 row_log 对象记录数据变更的增量（仅 rebuild 类型需要）。
- 生成新的临时ibd文件 new_table（仅rebuild类型需要）。

Execute 阶段：

- **降级EXCLUSIVE-MDL锁，允许读写。**
- 扫描old_table聚集索引（主键）中的每一条记录 rec。
- 遍历new_table的聚集索引和二级索引，逐一处理。
- 根据 rec 构造对应的索引项。
- 将构造索引项插入 sort_buffer 块排序。
- 将 sort_buffer 块更新到 new_table 的索引上。
- 记录 online-ddl 执行过程中产生的增量（仅 rebuild 类型需要）。
- 重放 row_log 中的操作到 new_table 的索引上（not-rebuild 数据是在原表上更新）。
- 重放 row_log 中的DML操作到 new_table 的数据行上。

Commit阶段：

- 当前 Block 为 row_log 最后一个时，**禁止读写，升级到 EXCLUSIVE-MDL 锁。**
- 重做 row_log 中最后一部分增量。
- 更新 innodb 的数据字典表。
- 提交事务（刷事务的 redo 日志）。
- 修改统计信息。
- rename 临时 ibd 文件，frm文件。
- 变更完成，**释放 EXCLUSIVE-MDL 锁。**

instant 算法

MySQL 8.0.12 才提出的新算法，**目前只支持添加列等少量操作**，利用 8.0 新的表结构设计，可以直接修改表的 metadata 数据，省掉了 rebuild 的过程，极大的缩短了 DDL 语句的执行时间。

pt-online-schema-change

借鉴了 copy 算法的思路，由外部工具来完成临时表的建立，数据同步，用临时表替换源表这三个步骤。其中数据同步是利用 MySQL 的触发器来实现的，会少量影响到线上业务的 QPS 及 SQL 响应时间。

gh-ost

详细信息参考其他的文章

MySQL 云数据库 MySQL 数据库 举报

目录