MySQL Replication之并行复制

小虾米 老叶茶馆 2022-07-01 07:00 发表于福建

传统单线程复制说明

> 众所周知，MySQL在5.6版本之前，主从复制的从节点上有两个线程，分别是I/O线程和SQL线程。

- I/O线程负责接收二进制日志的Event写入Relay Log。
- SQL线程读取Relay Log并在数据库中进行回放。

> 以上方式偶尔会造成延迟，那么可能造成主从节点延迟的情况有哪些?

- 1.主库执行大事务（如：大表结构变更操作）。
- 2.主库大批量变更（如：大量插入、更新、删除操作）。
- 3.ROW同步模式下，主库大表无主键频繁更新。
- 4.数据库参数配置不合理，从节点性能存在瓶颈（如：从节点事务日志设置过小，导致频繁刷盘）。
- 5.网络环境不稳定，从节点IO线程读取binlog存在延迟、重连情况。
- 6.主从硬件配置差异，从节点的硬件资源使用达到上限。(比如：主节点SSD盘，从节点SAS盘)

> 可以对以上延迟原因做个大致分类。

- 1.硬件方面问题（包括磁盘IO、网络IO等）

- 2.配置方面问题。
- 3.数据库设计问题。
- 4.主库大批量变更，从节点SQL单线程处理不够及时。

总结

> 分析以上原因可以看出要想降低主从延迟，除了改善硬件方面条件之外，另外就是需要DBA关注数据库设计和配置问题，最后就是需要提高从节点的并发处理能力，由单线程回放改为多线程并行回放是一个比较好的方法，关键点在于如何在多线程恢复的前提下解决数据冲突和故障恢复位置点的确认问题。

MySQL5.6基于库级别的并行复制

> 在实例中有多个数据库的情况下，可以开启多个线程，每个线程对应一个数据库。该模式下从节点会启动多个线程。线程分为两类 `Coordinator` 和 `WorkThread` 。

- 线程分工执行逻辑

`Coordinator` 线程负责判断事务是否可以并行执行，如果可以并行就把事务分发给 `WorkThread` 线程执行，如果判断不能执行，如 `DDL` ， `跨库操作` 等，就等待所有的worker线程执行完成之后，再由 `Coordinator` 执行。

- 关键配置信息

```
slave-parallel-type=DATABASE
```

- 方案不足点

> 这种并行复制的模式,只有在实例中有多个DB且DB的事务都相对繁忙的情况下才会有较高的并行度，但是日常维护中其实单个实例的的事务处理相对集中在一个DB上。通过观察延迟可以发现基本上都是基于热点表出现延迟的情况占大多数。如果能够提供基于表的并行度是一个很好方法。

MySQL5.7基于组提交的并行复制

组提交说明

> 简单来说就是在双1的设置下，事务提交后即刷盘的操作改为多个事务合并成一组事务再进行统一刷盘，这样处理就降低了磁盘IO的压力。详细资料参考 老叶茶馆 关于组提交的说明推文 https://mp.weixin.qq.com/s/rcPkrutiLc93aTblEZ7sFg

一组事务同时提交也就意味着组内事务不存在冲突，故组内的事务在从节点上就可以并发执行，问题在于如何区分事务是否在同一组中的，于是在binlog中出现了两个新的参数信息 last_committed 和 sequence_number

- 如何判断事务在一个组内呢?

> 解析binlog可以发现里面多了 last_committed 和 sequence_number 两个参数信息，其中 last_committed 存在重复的情况。

- sequence_number # 这个值指的是事务提交的序号，单调递增。
- last_committed # 这个值有两层含义，1.相同值代表这些事务是在同一个组内，2.该值同时又是代表上一组事务的最大编号。

```
[root@mgr2 GreatSQL]# mysqlbinlog mysql-bin.0000002 | grep last_committed
GTID last_committed=0 sequence_number=1
GTID last_committed=0 sequence_number=2
GTID last_committed=2 sequence_number=3
GTID last_committed=2 sequence_number=4
GTID last_committed=2 sequence_number=5
GTID last_committed=2 sequence_number=6
GTID last_committed=6 sequence_number=7
GTID last_committed=6 sequence_number=8
```

- 数据库配置

```
slave-parallel-type=LOGICAL_CLOCK
```

- 方案不足点

> 基于组提交的同步有个不足点，就是当主节点的事务繁忙度较低的时候，导致时间段内组提交fsync刷盘的事务量较少，于是导致从库回放的并行度并不高，甚至可能一组里面只有一个事务，这样从节点的多线程就基本用不到，可以通过设置下面两个参数，让主节点延迟提交。

- binlog_group_commit_sync_delay # 等待延迟提交的时间，binlog提交后等待一段时间再 fsync。让每个 group 的事务更多，人为提高并行度。
- binlog_group_commit_sync_no_delay_count # 待提交的最大事务数，如果等待时间没到，而事务数达到了，就立即 fsync。达到期望的并行度后立即提交，尽量缩小等待延迟。

MySQL8.0基于writeset的并行复制

> writeset 基于事务结果冲突进行判断事务是否可以进行并行回放的方法，他由 `binlog-transaction-dependency-tracking` 参数进行控制，默认采用 `WRITESET` 方法。

关键参数查看

Command-Line Format	--binlog-transaction-dependency-tracking=value
System Variable	binlog_transaction_dependency_tracking
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Enumeration
Default Value	COMMIT_ORDER
Valid Values	COMMIT_ORDER WRITESET WRITESET_SESSION

参数配置项说明

- `COMMIT_ORDER` # 使用 5.7 Group commit 的方式决定事务依赖。
- `WRITESET` # 使用写集合的方式决定事务依赖。
- `WRITESET_SESSION` # 使用写集合，但是同一个session中的事务不会有相同的last_committed。

> writeset 是一个HASH类型的数组，里面记录着事务的更新信息，通过 `transaction_write_set_extraction` 判断当前事务更新的记录与历史事务更新的记录是否存在冲突，判断过后再采取对应处理方法。writeset储存的最大存储值由 `binlog-transaction-dependency-history-size` 控制。

> 需要注意的是，当设置成 `WRITESET` 或 `WRITESET_SESSION` 的时候，事务提交是无序状态的，可以通过设置 `slave_preserve_commit_order=1` 强制按顺序提交。

- binlog_transaction_dependency_history_size

> 设置保存在内存中的行哈希数的上限，用于缓存之前事务修改的行信息。一旦达到这个哈希数，就会清除历史记录。

Command-Line Format	**--binlog-transaction-dependency-history-size=#**
System Variable	binlog_transaction_dependency_history_size
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	25000
Minimum Value	1
Minimum Value	1000000

- transaction_write_set_extraction

> 该模式支持三种算法，默认采用XXHASH64，当从节点配置writeset复制的时候，该配置不能配置为OFF。该参数已经在MySQL 8.0.26中被弃用，后续将会进行删除。

Command-Line Format	**--transaction-write-set-extraction[=value]**
Deprecated	8.0.26
System Variable	binlog_transaction_dependency_history_size

Command-Line Format	--transaction-write-set-extraction[=value]
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Enumeration
Default Value	XXHASH64
Valid Values	OFF MURMUR32 XXHASH64

数据库配置

```
slave_parallel_type = LOGICAL_CLOCK
slave_parallel_workers = 8
binlog_transaction_dependency_tracking = WRITESET
slave_preserve_commit_order = 1
```

免责申明：

- 个人水平有限，如有错漏欢迎评论区留言指出，有其他想法建议亦欢迎留言讨论。

引用资料：

- 阿里内核月报： http://mysql.taobao.org/monthly/2018/06/04/
- 官方文档： https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html

Enjoy GreatSQL :)

小程序

https://www.bilibili.com/medialist/play/1363850082?business=space_collection&business_id=343928&desc=0

文章推荐：

- Linux环境监控工具汇总
- MySQL主从复制之半同步(semi-sync replication)
- MySQL复制主从实例表DDL不一致导致失败案例
- 如何分析 mysqld crash 的原因
- 简单学习一下ibd数据文件解析

想看更多技术好文，点个**“在看”**吧！

阅读原文

喜欢此内容的人还喜欢

总监又来了，人狠话不多，这篇 gRPC，小弟佩服！

楼仔

PostgreSQL 数据库统计信息概览

yangyidba

Linux 下让工作效率翻倍的四个实用技巧

Linux学习