当主的量很大，导致主从同步延迟的时候

有个参数调整一下　会在性能方面提高的比较多

调整从库的这个参数

innodb_flush_log_at_trx_commit=2

默认是1

Renolei

Live Free or Die

博客园　首页　新随笔　联系　订阅　管理

[MySQL] 参数: innodb_flush_log_at_trx_commit和sync_binlog

MySQL参数: innodb_flush_log_at_trx_commit和sync_binlog

`innodb_flush_log_at_trx_commit` 和 `sync_binlog` 是MySQL控制磁盘写入策略的重要参数.

- **innodb_flush_log_at_trx_commit**

Command-Line Format	`--innodb_flush_log_at_trx_commit[=#]`	
Option-File Format	`innodb_flush_log_at_trx_commit`	
System Variable Name	`innodb_flush_log_at_trx_commit`	
Variable Scope	Global	
Dynamic Variable	Yes	
	Permitted Values	
	Type	`enumeration`
	Default	1
		0
		1
	Valid Values	2

- 当 `innodb_flush_log_at_trx_commit=0` 时, log buffer将每秒一次地写入log file, 并且log file的flush(刷新到disk)操作同时进行. 此时, 事务提交是不会主动触发写入磁盘的操作.
- 当 `innodb_flush_log_at_trx_commit=1` 时(默认), 每次事务提交时, MySQL会把log buffer的数据写入log file, 并且将log file flush(刷新到disk)中去.
- 当 `innodb_flush_log_at_trx_commit=2` 时, 每次事务提交时, MySQL会把log buffer的数据写入log file, 但不会主动触发flush(刷新到disk)操作同时进行. 然而, MySQL会每秒执行一次flush(刷新到disk)操作.

> 然而, 每秒flush并不能确保100%每秒发生, 因为os调度问题.

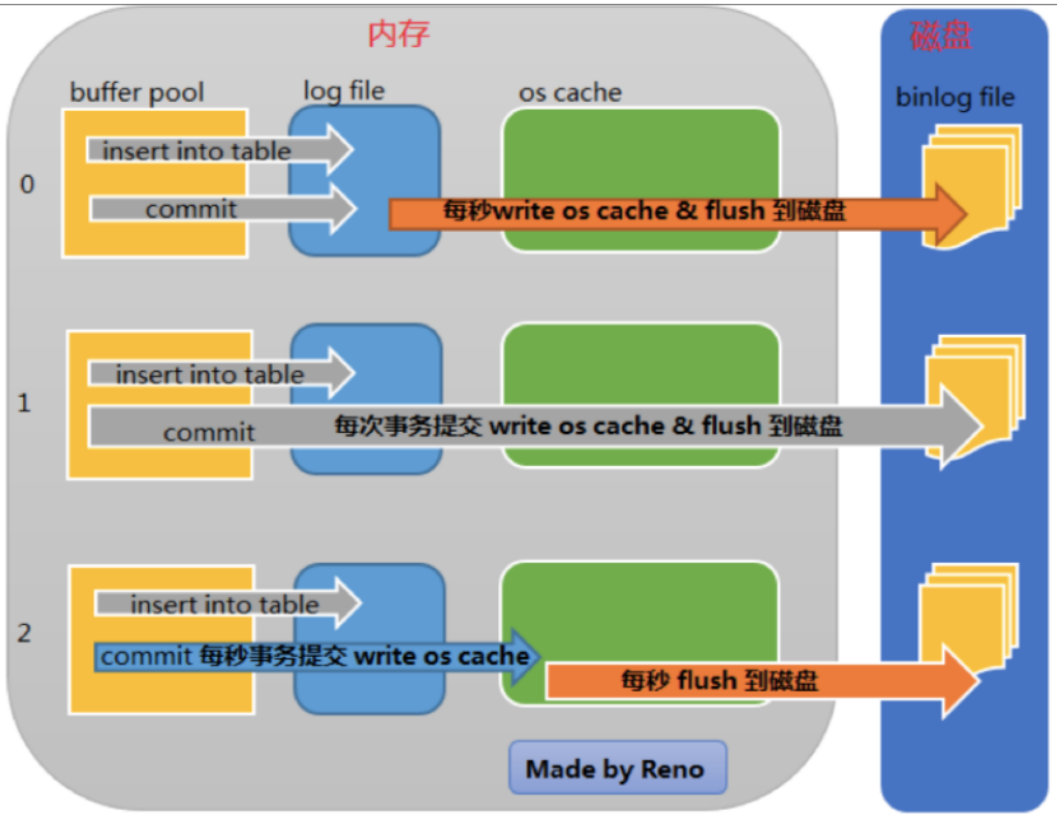
默认的1可以获得更好地数据安全, 但性能会打折扣. 不过非1时, 在遇到crash可能会丢失1秒的事务; 设置为0时, 任何mysqld进程crash会丢失上1秒的事务; 设置为2时, 任何os crash或者机器掉电会丢失上1秒的事务; InnoDB的crash recovery运行时会忽略这些数据.

- **sync_binlog**

```
![](http://images2015.cnblogs.com/blog/748358/201603/748358-20160327131735761-1330837796.png)
```

  - 当 `sync_binlog=0` 时(默认), 如os刷新其他文件的机制一样, MySQL不会刷新log buffer到disk中去, 而是依赖os机制刷新log buffer数据到binary log中.
  - 当 `sync_binlog=1` 时, MySQL在写1次二进制日志binary log时, 会使用fdatasync()函数将二进制binary log同步到disk中去.(安全性最高的配置)
  - 当 `sync_binlog=N(N>1)` 时, MySQL在写N次二进制日志binary log时, 会使用fdatasync()函数将二进制binary log同步到disk中去.

当两个参数设置为双1的时候, 写入性能最差. 当sync_binlog=N(N>1) && innodb_flush_log_at_trx_commit=2, MySQL的写操作才能达到最高性能.



---

mysql主从同步延迟问题解决方法

银河zlm2017-03-20 17:39:59

mysql主从同步延迟问题解决方法：

在从服务器上执行show slave status;可以查看到很多同步的参数，我们需要特别注意的参数如下：

Master_Log_File：SLAVE中的I/O线程当前正在读取的主服务器二进制日志文件的名称

Read_Master_Log_Pos：在当前的主服务器二进制日志中，SLAVE中的I/O线程已经读取的位置

Relay_Log_File：SQL线程当前正在读取和执行的中继日志文件的名称

Relay_Log_Pos：在当前的中继日志中，SQL线程已读取和执行的位置

Relay_Master_Log_File：由SQL线程执行的包含多数近期事件的主服务器二进制日志文件的名称

Slave_IO_Running：I/O线程是否被启动并成功地连接到主服务器上

Slave_SQL_Running：SQL线程是否被启动

Seconds_Behind_Master：从属服务器SQL线程和从属服务器I/O线程之间的时间差距，单位以秒计。

从库同步延迟情况出现的

1、show slave status显示参数Seconds_Behind_Master不为0，这个数值可能会很大

2、show slave status显示参数Relay_Master_Log_File和Master_Log_File显示bin-log的编号相差很大，说明bin-log在从库上没有及时同步，所以近期执行的bin-log和当前IO线程所读的bin-log相差很大

3、MySQL的从库数据目录下存在大量mysql-relay-log日志，该日志同步完成之后就会被系统自动删除，存在大量日志，说明主从同步延迟很厉害

a、MySQL数据库主从同步延迟原理

mysql主从同步原理：

主库针对写操作，顺序写binlog，从库单线程去主库顺序读”写操作的binlog”，从库取到binlog在本地原样执行（随机写），来保证主从数据逻辑上一致。

mysql的主从复制都是单线程的操作，主库对所有DDL和DML产生binlog，binlog是顺序写，所以效率很高，slave的Slave_IO_Running线程到主库取日志，效率比较高，下一步，问题来了，slave的Slave_SQL_Running线程将主库的DDL和DML操作在slave实施。DML和DDL的IO操作是随即的，不是顺序的，成本高很多，还可能可slave上的其他查询产生lock争用，由于Slave_SQL_Running也是单线程的，所以一个DDL卡主了，需要执行10分钟，那么所有之后的DDL会等待这个DDL执行完才会继续执行，这就导致了延时。

有朋友会问：“主库上那个相同的DDL也需要执行10分，为什么slave会延时？”，答案是master可以并发，Slave_SQL_Running线程却不可以。

b、 MySQL数据库主从同步延迟是怎么产生的？

当主库的TPS并发较高时，产生的DDL数量超过slave一个sql线程所能承受的范围，那么延时就产生了，当然还有就是可能与slave的大型query语句产生了锁等待。

首要原因：数据库在业务上读写压力太大，CPU计算负荷大，网卡负荷大，硬盘随机IO太高

次要原因：读写binlog带来的性能影响，网络传输延迟。

c、 MySQL数据库主从同步延迟解决方案。

架构方面

1.业务的持久化层的实现采用分库架构，mysql服务可平行扩展，分散压力。

2.单个库读写分离，一主多从，主写从读，分散压力。这样从库压力比主库高，保护主库。

3.服务的基础架构在业务和mysql之间加入memcache或者Redis的cache层。降低mysql的读压力。

4.不同业务的mysql物理上放在不同机器，分散压力。

5.使用比主库更好的硬件设备作为slave

总结，mysql压力小，延迟自然会变小。

硬件方面

1.采用好服务器，比如4u比2u性能明显好，2u比1u性能明显好。

2.存储用ssd或者盘阵或者san，提升随机写的性能。

3.主从间保证处在同一个交换机下面，并且是万兆环境。

总结，硬件强劲，延迟自然会变小。一句话，缩小延迟的解决方案就是花钱和花时间。

mysql主从同步加速

1、sync_binlog在slave端设置为0

2、 -logs-slave-updates 从服务器从主服务器接收到的更新不记入它的二进制日志。

3、直接禁用slave端的binlog

4、slave端，如果使用的存储引擎是innodb，innodb_flush_log_at_trx_commit =2

从文件系统本身属性角度优化

master端

修改linux、Unix文件系统中文件的etime属性， 由于每当读文件时OS都会将读取操作发生的时间回写到磁盘上，对于读操作频繁的数据库文件来说这是没必要的，只会增加磁盘系统的负担影响I/O性能。可以通过设置文件系统的mount属性，组织操作系统写atime信息，在Linux上的操作为：

打开/etc/fstab，加上noatime参数

/dev/sdb1 /data reiserfs noatime 1 2

然后重新mount文件系统

#mount -oremount /data

PS：

主库是写，对数据安全性较高，比如sync_binlog=1，innodb_flush_log_at_trx_commit = 1 之类的设置是需要的

而slave则不需要这么高的数据安全，完全可以讲sync_binlog设置为0或者关闭binlog，innodb_flushlog也可以设置为0来提高sql的执行效率

1、sync_binlog=1 o

MySQL提供一个sync_binlog参数来控制数据库的binlog刷到磁盘上去。

默认，sync_binlog=0，表示MySQL不控制binlog的刷新，由文件系统自己控制它的缓存的刷新。这时候的性能是最好的，但是风险也是最大的。一旦系统Crash，在binlog_cache中的所有binlog信息都会被丢失。

如果sync_binlog>0，表示每sync_binlog次事务提交，MySQL调用文件系统的刷新操作将缓存刷下去。最安全的就是sync_binlog=1了，表示每次事务提交，MySQL都会把binlog刷下去，是最安全但是性能损耗最大的设置。这样的话，在数据库所在的主机操作系统损坏或者突然掉电的情况下，系统才有可能丢失1个事务的数据。

但是binlog虽然是顺序IO，但是设置sync_binlog=1，多个事务同时提交，同样很大的影响MySQL和IO性能。

虽然可以通过group commit的补丁缓解，但是刷新的频率过高对IO的影响也非常大。对于高并发事务的系统来说，

“sync_binlog”设置为0和设置为1的系统写入性能差距可能高达5倍甚至更多。

所以很多MySQL DBA设置的sync_binlog并不是最安全的1，而是2或者是0。这样牺牲一定的一致性，可以获得更高的并发和性能。

默认情况下，并不是每次写入时都将binlog与硬盘同步。因此如果操作系统或机器(不仅仅是MySQL服务器)崩溃，有可能binlog中最后的语句丢失了。要想防止这种情况，你可以使用sync_binlog全局变量(1是最安全的值，但也是最慢的)，使binlog在每N次binlog写入后与硬盘同步。即使sync_binlog设置为1,出现崩溃时，也有可能表内容和binlog内容之间存在不一致性。

2、innodb_flush_log_at_trx_commit （这个很管用）

抱怨Innodb比MyISAM慢 100倍？那么你大概是忘了调整这个值。默认值1的意思是每一次事务提交或事务外的指令都需要把日志写入（flush）硬盘，这是很费时的。特别是使用电池供电缓存（Battery backed up cache）时。设成2对于很多运用，特别是从MyISAM表转过来的是可以的，它的意思是不写入硬盘而是写入系统缓存。

日志仍然会每秒flush到硬 盘，所以你一般不会丢失超过1-2秒的更新。设成0会更快一点，但安全方面比较差，即使MySQL挂了也可能会丢失事务的数据。而值2只会在整个操作系统

挂了时才可能丢数据。
3、ls(l) 命令可用来列出文件的 atime、ctime 和 mtime。
atime 文件的access time 在读取文件或者执行文件时更改的
ctime 文件的create time 在写入文件，更改所有者，权限或链接设置时随inode的内容更改而更改
mtime 文件的modified time 在写入文件时随文件内容的更改而更改
ls -lc filename 列出文件的 ctime
ls -lu filename 列出文件的 atime
ls -l filename 列出文件的 mtime
stat filename 列出atime，mtime，ctime
atime不一定在访问文件之后被修改
因为：使用ext3文件系统的时候，如果在mount的时候使用了noatime参数那么就不会更新atime信息。
这三个time stamp都放在 inode 中.如果mtime,atime 修改,inode 就一定会改，既然 inode 改了,那ctime也就跟着改了.
之所以在 mount option 中使用 noatime，就是不想file system 做太多的修改，而改善读取效能
mysql主从同步延迟问题解决方法