主要讲解复制的参数优化

． ' 00m10n 还 0m 
． 复 制 和 能 相 关 的 0 数 
· Mas@配 肯 优 化 
io 」 h 。 0 羽 《 置 伉 化 
· sq 凵 h 「 “ 0 配 置 优 化 
大 纲 
． 复 制 中 詎 要 功 能 启 所 及 注 意 事 项 
复 制 过 谑 及 实 战 中 注 怠 申 项 
． a 蝻 a 倦 化 帥 0a00n 配 置 〔 安 全 } 
． sync 过 e000g 、 相 关 配 肯 （ 安 全 〕 
· 迂 迟 皂 制 
嗶 多 源 复 制 
嗶 增 强 半 同 步 复 制 配 置 及 注 意 事 项 

.•mshutang.c-x.m 
• Mastcrådßfftft 
• 
binlog_row image—lull 
• 
• Glid_mode 
• enfcrce_gtid_consistency 
• 
• 
• 
• 
• 
max_binlog_size- 1 G 
sync_binIog=O 
—on 
bin 
bin 10 
ependency_tracking RDER 

横线下面的部分是5.7.22之后的版本才有。

transaction\_write\_set\_extraction 不会对主库性能有任何提升，但是对从库有帮助，但是要求从库要开启并行复制，建议从库开4到8个并行复制

rheshutang.c•x.m 
json 
• binlog_row_image=image 
• binlog_row_value_options=PARTIAL_JSON 
binlog_row_value_options=PARTlAL—JSON 

5.7的json更新是把全部的json字段写到binlog中，8.0针对这项进行了优化，只是把更新的json字段写到binlog中，但是这个功能默认是没有开启的，需要我们经过参数来做；

将binglog\_row\_image=image，binlog\_row\_value\_options=PARTIAL\_JSON就可以启用了更新部分json字段功能了。

================json 结束====================

binlog group commit 针对的主库的写日志的优化。

才 00j0 n 《 0m 
复 制 中 重 要 参 数 
0 0 0 忙 
SET 螽 A 湖 献 
V为桕@ 
inimu 而 唁 0 
0 n m 地 0 
SET 戳 R Hi' 讶 
000u0 ． ． 妇 丛 
0 
瑟 小 m 刂 m 
0 
Maximum V 的 《 
1000000 
5 

binlog group commit功能：（针对主库提交binlog进行优化，提高主库的并行度）

什么是binlog group commit？举例说明，长途汽车站北京通天津的长途客车，准许乘坐55人，如果发车满足条件1或者2或者3：

条件1：坐满55人了，但是没有到发车时间

条件2：到发车时间，即使没有坐满55人

条件3：到了发车时间，刚好55人 ----- 自己想到的

通过例子抽象到binlog group commit，第一：等到多长时间，到了时间commit，第二：等待数量，事物数量一但满了commit。都会有参数对应。

参数一：

参数二：带有no的

如果这项功能之前，来一个commit一个，比较浪费。

.•mshutang.c-x.m 
binlog group commit 
• Binlog Group Commitfi5NE 
flush Stage 
• 
• Sync Stage 
Commit Stage 
• 
LSÅüfj: 
InnoDB prepare 
write/sync Binlog 
InnoDB commit 
6 

没有使用组提交，就是先写redo，filename position合并到redo里面。

现在使用组提交，排成了三个队列。

flush stage

sync stage

commit stage

binglog group commit 
• binlogcP31j\: 
• sequence_numher 9äY3%lD 
• last_commited 
•Group 18 

解析binlog，在binlog中看到last\_commited和sequence\_number。

那么last\_commited 是怎么来的呢？上一个事物提交的最大值。

以前没有使用group commit在做sync\_binlog=1，每次写binlog的时候都要做一次f sync写入。

在启动group commit之后变成每一个队列做一个f sync写入，这样就提高了一次提交的事物的数量，一次的io增多了。

-----------f sync指的是flush sync？

断 掉 了 ， 系 统 c “ 里 面 数 据 再 杩 ？ 
0 
断 电 了 
0 
启 交 0 邴 判 “ 一 的 了 吧 
才 没 听 设 舀 彎 吡 b 阍 = 
． 三 ． 北 点 
， 打 开 了 oup 稿 ， 是 不 就 变 成 一 龃 一 提 交 了 
开 扈 $ “ 。 一 № 地 驴 1 后 ， 每 个 9- p 脚 就 进 了 一 次 櫪 叠 其 且 别 新 0 凶 · 而 不 开 启 9 。 c “ 训 之 
的 个 事 务 提 交 刷 醣 收 一 次 。 这 样 没 间 把 师 
0 
每 个 骱 段 宕 机 的 情 况 昵 

设置sync\_binlog=1，打开group commit 就会组提交。

==============组提交完事=====================

writeset针对从库的sql\_thread进行的优化，

rh.snutang.t•x.m 
8.0fi5WriteSet 
• binlog group commitÆ57GlÄ, 
GSlave 
WriteSet 
• binlog_transaction_dependency_tracking=commit_orderlwritesetlwriteset_session 
• transaction_write_set_extraction (hashfii±) 
• binlog_transaction_dependency_history_size (hash*'h) 
8 
5.7 binlog_group_commit_sync_delay fiB3i9Å 
8.0 

在 8 ． 0 中 根 据 数 据 库 配 置 高 低 设 置 binlog-transaction-dependency-hlstory-size, 性 能 有 富 余 的 实 例 可 
以 适 当 调 大 该 爹 数 ， 找 到 更 小 的 commit parent, 提 高 备 库 回 放 并 行 度 。 内 存 手 PU 紧 张 的 实 例 最 好 避 免 
在 w 诅 eset 上 消 耗 太 身 资 源 ， binlog-transaction-dependency-history-size 过 大 不 光 消 耗 内 存 ， 还 会 
降 低 冲 突 查 询 的 效 率 。 

ppt的备注部分来自和淘宝的交流。

这个是针对slave起作用的，但是要slave要开启并行复制。如果没有开启并行复制，这个参数没什么作用，还要开启基于逻辑时钟的复制，mysql变成了一组参数控制一个功能。

binlog\_transaction\_dependency\_tracking：

commit\_Order ： 依赖主库binlog group commit，依赖binlog group commit 的顺序。

writeset：依赖thread\_id。

writeset\_session：推荐使用，避免了两个链接去改同一张表

要使用writeset，还需要transaction\_write\_set\_extraction，这个参数起的作用就是怎么把sql分配到并行的线程里面，假如有4个线程，那么那个sql该去那个线程去执行呢？这里就需要这个参数来控制，利用hashcode，拿到sql的hashcode，对线程数取模（只要结果在线程个数内就可以），假如取模等于2，那么这个sql就去2号线程去执行。这个参数在配置文件中不用动。

binlog\_transaction\_dependency\_history\_size ：这个不用动，但是什么意思，起了什么作用。数值设置大了可能会导致更新丢失。

复 制 中 重 要 参 数 
Command-LineFormat 
In 0 
namie 
SET VAR Hint 
Valid Values 
V 以 ， 
一 到 ； 一 t 葛 秘0《支0身一过•歸傘葛@@00y一 
《 t 葛 孬 孬 g 《 04 
NO 

rh.snutang.c•x.m 
Property 
Command-Line Format 
Introduced 
System Variable 
namic 
SET VAR Hint A 
pe 
Default Value 
Minimum Value 
Maximum Value 
lies 
Value 
- -binlog- transaction—dependency-history- 
8.0.1 
binlog transaction dependency history size 
Global 
Yes 
No 
integer 
1 
10 
Sets an upper limit on the number of row hashes which are kept in memory and used for looking up 
the transaction that last modified a given row. Once this number of hashes has been reached, the 
history is purged. 

AIOOO ． 能 · 上@ 
希 试 可 以 监 控 te 1 么 

===================writeset结束================

rms,hutang.c-x.m 
• io_thread 
•o *20130 
• io_threadßfifftftt change master to 
• MASTER_DeIay-O 

slave\_net\_timeout ：感知和主库断开，假如这个参数设置20，那么和主库断开20秒slave才会有反应。

master\_bind ： 如果使用的是ssd等高速盘，那么千兆的网卡可能会支撑不住，那么可以利用这个参数绑定一个网卡。

SQL_Thread 
• slave_preserve_commit_order=on 
slave_preserve_commit_order 
log-bin 
slave-log-updates 

====================参数结束======================

复制过滤，建议在从库上进行配置。

• CHANGE AEPLICATION FILTER 
= {dh_list} 
1-1 
REPLICATE_DO DB (db_list) 
I (db_list) 
I (tbl_list) 
I (tbl_list) 

复制的过滤规则使用在什么场景下？知道mysql支持这样的功能即可。

2）--还是针对从库

使用start slave启动从库，但是relay-log损坏了，无法启动，这个时候怎么办？

手工方法解决问题。

如果使用了gtid，可以在从库上 reset slave，再start slave。

如果没有使用gtid呢？执行到那个位置，然后再从那个位置重新去拉取。

执行到那个位置怎么来判断？

sql\_thread 对应relay\_ma…..

sql_thread- > relay_master_log_file, exec_master_log_pos 
9 
purge relay logÄfil 
stop slave ; change master to 
start slave io_thread; 

在做change master 会把本地的relay-log全部清理掉，然后告诉mysql从那个文件那个位置重新拉取一份到本地，作为新的relay-log。

为了防止slave挂掉，relay-log损坏，默认没有开启。

2%lJ crash-safe replication 
• relay_log_info_repository=table 
• relay_log_recovery=l 
START TRANSACTION 
- Statement 1 
-- Stat•nt N 
- Update replication info 
COMMIT: 

3）

Command-Line Format 
System Variable 
Permitted Values (32-båt platforms) 
Values (64-bit platforn;ts} 
show slave 
http://wubx.net/faq-show-slave- 
8D%AE/ 
— — —eel —I 
variable Scope I 
Dyrvarn,ic Variable I Yes 
Jype—J 
Default 
Min Value 
Max value 
10000 
-Type—J integer 
Default 
Min Value 
Max Vilue 
] 10000 
184 467440737095516151 

4)

• sync_relay_log_info=l & 
relay_log_info_repository=file 
• & relay_log_info_repository=table 
• replication 

如果改成1和file的情况下是性能杀手。

5）延迟复制---针对从库

mysql>stop slave sql_thread; 
mysql>change master to \ 
mysql>start slave sql_thread; 
mysql>stop slave sql_thread; 
mysql>change master to 
mysql>starl slave sql_thread; 

延迟在什么场景下使用？

灾备，T+1,如果是数据量比较大，可以把从库作为备份库，防止误操作，为了争取手动干预的时间。

备份是防灾难的，数据审计的。

． 中 r 旧 1 n000m 
复 制 中 重 要 功 能 启 用 
· 延 迟 复 制 
02 、 
主 库 挂 了 ， 延 迟 复 制 会 不 会 丢 相 应 的 数 据 ？ 
延 迟 复 制 有 什 么 戽 用 场 景 ？ 

binlog里面有时间戳，根据时间戳来判断。

6）多源复制

change master to \ 
for channel 'channelnamei 
enmulti-source replicationßN%lJ 
+GTID 
gtid_modc=an 
22 

gtid _purge 
stop slave sql_thread; 
show master status; 
set global new_dump_gtid'; 

在set 设置值之前要做resrt master操作，但是具体在什么位置reset可以自己控制。

就是 先备份一下gtid，然后reset master，然后导入，在把备份的和新的一起加进去，在开启---来子A779

start slave for channel isl-3306'; 
change master to master_auto_position=l for channel 
62-3307'; 
start slave for channel M-3307'; 
mydump#ÄÆZ 
reset masterg 

老师没有搭建多源复制，这里要自己探索一下。

复 制 中 重 要 功 能 启 用 
0 № 0 Chun 1 （ 命 習 通 、 
Nan 0 C 00 
， 复 制 
Ma T•r 产 0 
Channel 
no 呷 
insert 

在A库里面单独定义一个channel1，会单独分配 io-thread和sql-thread

ibdata1没有办法合并到一块，只能用逻辑备份。

表空间传输是什么回事？

7）增强半同步

． ' 0 》 “ 1 0 《 0m 
复 制 中 重 要 功 能 启 用 
增 强 司 十 
· 加 *iplugirt 
庸 逮 《 酉 LL 「 飞 L•C 的 吝 00 ． 矿 鹵 黼 SO 》 匚 ： ” 000 忙 0 
· SET GLOBAL 0 semi 00 ． 酽 s 地 0 乍 圊 = 1 ， 
· S00L00 為 L 0 一 mi_sy: 忙 一 s 地 七 禰 t —N 
． SET 03 庶 L 0 mL500 一 0 」 彎 丽 」 00 蚓 a 脚 n N 
· SET GLOB 《 》 一 一 」 a 飞 0 一 皂 nac 地 d 。 1 

在主库从库都会加载这两个插件，原因如下。

身 € 一 0 
囍 北 0 
动 ． 赵 岍 上 陶 
有 高 可 用 切 挨 的 ， 必 须 都 装 
蛔 背 上 
插 件 可 以 加 上 ， 但 是 数 不 能 都 开 

增强半同步对网络要求比较高。ping的时候time 小于2

000g ． 《 0m 
复 制 中 重 要 功 能 启 用 
· 增 强 半 同 步 
· 如 果 Master 上 全 部 从 库 挂 掉 ， 可 以 临 时 禁 用 半 同 
步 ， 让 Maste 树 外 提 供 服 务 
· 如 果 拿 一 个 备 份 新 建 一 个 从 库 ， 确 认 ℃ 」 ead 追 上 
主 库 后 再 开 始 半 同 步 
rpl_semi_sync master_timeout 多 合 舌 ？ 

今天的都是重点。

划 重 点 
· 异 步 复 制 中 参 数 优 化 
。 增 强 半 同 步 参 数 
。 复 制 过 滤 
． 多 源 复 制 
· 理 *binlog group commit &wirteset 

参数优化1

• 
transaction_write_set_extraction=on 
• gtid_mode 
enforce_gtid_consistency 
• 100 
binlog_group_commit_sync_no_delay_count=10 
slave_parallel_type 
slave_parallel_workers 
=on 
—on 
—4 
binlog_transaction_dependency_trackingswriteset_session 

参数优化2

• 
• 
• 
• 
• 
plugin_load_add 
=semisync_master.so 
—semisync_slave.so 
rpl_semi 
rpl_semi 
rpl_semi 
rpl_semi 
master enabled 
_sync_ 
_sync_slave_enabled 
_sync_master_timeout 
_sync_master_wait_point 
= 1000 
=after_commit 

参数优化3

• set global 
• set global 
• set global super_read_only=0; 
• set global 1 
• set global 
• set global 
• set global 
• set global 
set global 
set global 
31 

强调数据安全，又强调性能，从库跟不上主库：

从库配置可以松一点，性能为王，主库上安全性更高一点。主要是优化画圆圈的四个参数。