关于MySQL checkpoint

Ⅰ、Checkpoint

1.1 checkpoint的作用

- 缩短数据库的恢复时间
- 缓冲池不够用时,将脏页刷到磁盘
- 重做日志不可用时,刷新脏页

1.2 展开分析

page被缓存在bp中,page在bp中和disk中不是时刻保持一致的(page修改一下就刷一次盘是不现实的,是通过checkpoint来玩的)

万一宕机,重启的时候disk上那个page需要恢复到原来bp中page的那个版本

那问题是,两个page版本不一致咋整？没事,我们做到最终一致就行

那我们就说一下这个最终一致是个怎样的过程,通过一个例子来说明：

昵称： 海东潮
园龄： 6年11个月
粉丝： 62
关注： 2
+加关注

< 2025年8月 >

日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

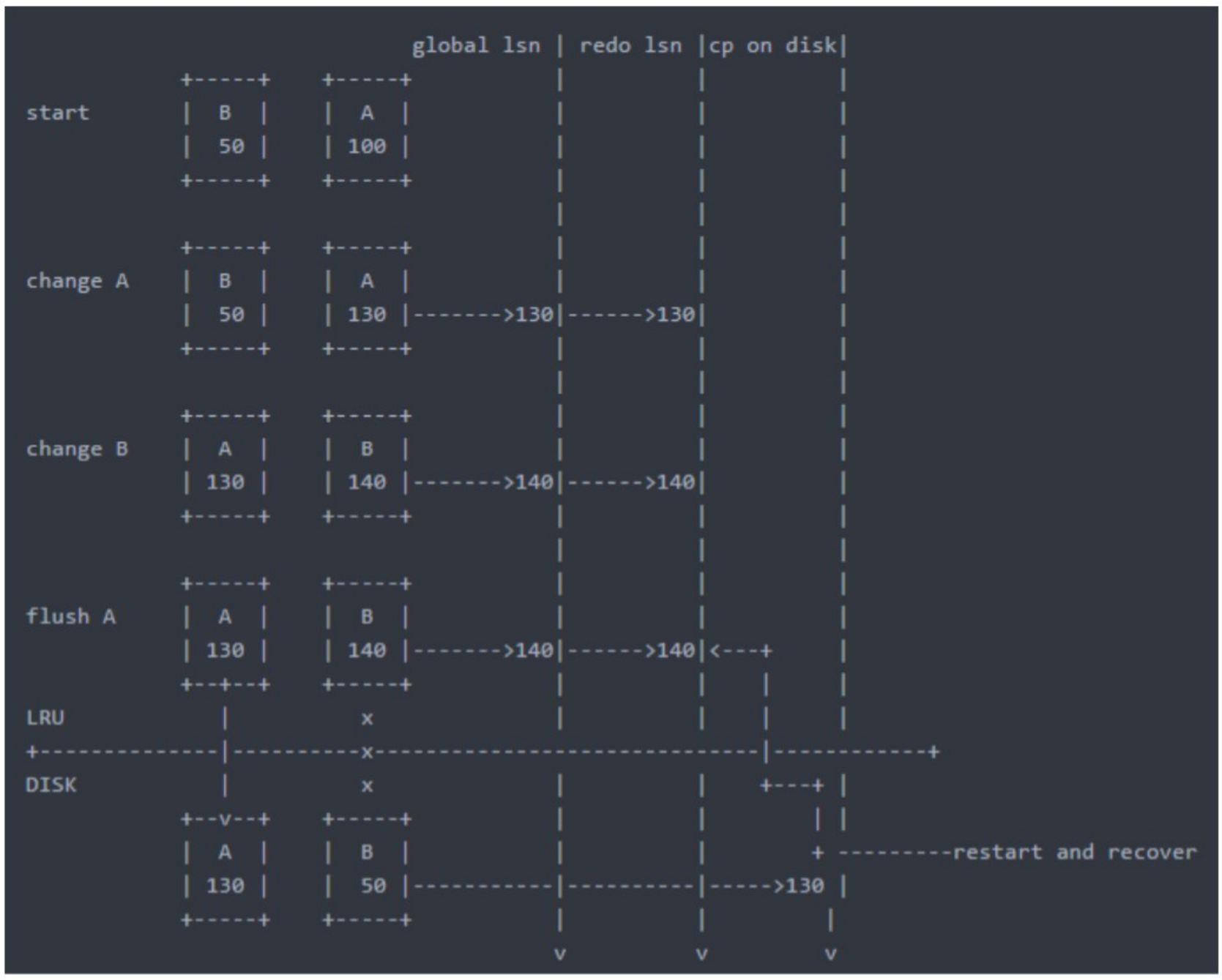
搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

我的标签




```
Step1:
一个page读到bp中时,它的lsn（这个鬼东西待会儿仔细说,先理解为一个flag）是100,然后这个page被modify了,它的lsn变成了13
Step2:
另外一个page之前进bp的时候lsn是50,前面那个page被modify之后,它也被修改,它的lsn变成了140,它这个140的lsn也写到了red
Step3:
关键的一步,假设此时lsn为130的page被刷到disk上了（什么时候刷也是个学问,这里不说）,而lsn为140的那个page还没被刷,磁盘
Step4:
这时候restart数据库,就会从磁盘上cp的位置(130)开始读redo log,一直回放到140,这样没被刷到磁盘的那个page就恢复到宕机之
```

划重点：

①这个130,140其实就是字节数,也就是说你对这个页修改产生了10个字节的日志,那么lsn就加10

②page原来读进bp的lsn甭管,只管它改变了多少字节就行,所以这个lsn的变化肯定是一个单调递增的过程,其实lsn就是日志写了多少字节(之前没理解好,以为各个page的lsn是自己玩自己的)

II 、LSN(log sequence number)——日志序列号

lsn是用来保存checkpoint的,保存现在刷新到磁盘的位置在哪里

这个130,140其实就是字节数,也就是说你对这个页修改产生了10个字节的日志,那么lsn就加10,lsn没有上限,8字节

2.1 lsn存在什么地方?

- 每个page有一个LSN,page更新一下LSN就会更新一下,记录在page header中
- 整个MySQL实例也有一个LSN(这就是checkpoint),记录在第一个重做日志的前2k的块里(就给它用,不会被覆盖)
- redo log里有一个LSN

全局lsn位置之前的内容已经刷磁盘上,只要恢复它后面的日志,数据就恢复了

2.2 查看lsn和整个checkpoint流程梳理

mysql(208)
linux(54)
oracle(24)
performance(16)
新特性(15)
memory(14)
复制(12)
pt(11)
replication(10)
lock(10)
更多

随笔分类 (379)
AWS(1)
Cloud(1)
DataArch(3)
EBS(1)
Golang(2)
Goldengate(2)
Linux(59)
MySQL(222)
NewSQL(1)
Oracle(44)
Percona(3)
python(1)
Python(3)
架构(1)
压力测试(2)
运维(27)
字符编码与存储(6)

随笔档案 (403)
2019年2月(16)
2019年1月(99)
2018年12月(153)
2018年11月(92)
2018年10月(32)
2018年9月(5)
2018年8月(6)

文章分类 (21)

看page中的lsn,page中其实是保存两个lsn的,如下：

```
(root@172.16.0.10) [(none)]>  desc information_schema.INNODB_BUFFER_PAGE_LRU;
+---------------------+---------------------+------+-----+---------+-------+
| Field               | Type                | Null | Key | Default | Extra |
+---------------------+---------------------+------+-----+---------+-------+
| POOL_ID             | bigint(21) unsigned | NO   |     | 0       |       |
| LRU_POSITION        | bigint(21) unsigned | NO   |     | 0       |       |
| SPACE               | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_NUMBER         | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_TYPE           | varchar(64)         | YES  |     | NULL    |       |
| FLUSH_TYPE          | bigint(21) unsigned | NO   |     | 0       |       |
| FIX_COUNT           | bigint(21) unsigned | NO   |     | 0       |       |
| IS_HASHED           | varchar(3)          | YES  |     | NULL    |       |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| OLDEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| ACCESS_TIME         | bigint(21) unsigned | NO   |     | 0       |       |
| TABLE_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| INDEX_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| NUMBER_RECORDS      | bigint(21) unsigned | NO   |     | 0       |       |
| DATA_SIZE           | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED_SIZE     | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED          | varchar(3)          | YES  |     | NULL    |       |
| IO_FIX              | varchar(64)         | YES  |     | NULL    |       |
| IS_OLD              | varchar(3)          | YES  |     | NULL    |       |
| FREE_PAGE_CLOCK     | bigint(21) unsigned | NO   |     | 0       |       |
+---------------------+---------------------+------+-----+---------+-------+
20 rows in set (0.00 sec)

newest_modification 页最新更新完后的lsn
oldest_modification 页第一次更新完后的lsn
page刷到磁盘的时候,全局的check_point保存的是oldest(只保存第一次修改时的lsn),而page中的lsn保存的是newest

(root@172.16.0.10) [(none)]> show engine innodb status\G
...
---
LOG
---
Log sequence number 15151135824    当前内存中最新的LSN
Log flushed up to   15151135824    redo刷到磁盘的LSN
Pages flushed up to 15151135824    最后一个刷到磁盘上的页的最新的LSN（NEWEST_MODIFICATION）
Last checkpoint at  15151135815    最后一个刷到磁盘上的页的第一次被修改时的LSN（OLDEST_MODIFICATION）
...

Log sequence number和Log flushed up这两个LSN可能会不同,运行过程中后者可能会小于 前者,因为redo日志也是先在内存中

最后一个小于前面三个,为什么?
```

脏页会被指向flush list这个就不多赘述了

flush list是根据lsn进行组织的,而且还是用一个page第一次放进来的lsn进行组织的,也就是说这个page再次发生更新,它的位置是不会移动的

分析一波：

bp的LRU列表中,一个page,假设LSN进来的时候是100,当前全局LSN也是100,如果这个page变化了,产生了20字节的日志,这时候page的lsn变成120,并且通过指针指向flush list中去了,但是这个page立马又被更新产生20字节日志,此时page的lsn为140,而此时在flush list中的lsn还是120(这里意思就是page里面保存了两种lsn,一个是第一次修改页的,一个是最后一次修改页的)

当这个lsn为120的page被刷到disk上,那么disk上的cp就是120了,但是上面的三个值都是140,是不是很好理解呢,那就是说,每个page只更新一次,那这四个值就相等了呗,23333!

为什么这么设计?

为了恢复的时候,保证redo回放的过程的连续性,不会出错

page A第一次修改后lsn是120,记录到全局lsn,后面还有个page B被更新,lsn变为140,此时,page A再更新,lsn变为160了。这时候发生宕机,page A被刷到磁盘,page B没刷过去,如果flush list里面记录160的话,发生故障重启时lsn为140的page B怎么恢复？是不是被跳过去了

那从120开始恢复,那个页已经是160了,为什么还要恢复?

数据库会检测,如果page的lsn大于实例的lsn,就不会恢复这个page,跨过去,只将page B从120恢复到140

tips：

①checkpoint不需要实时刷新到磁盘,不是一个页更新了就要更新磁盘上的cp,磁盘上的cp前置一点是没有关系的,大不了多scan一点redo log,读到不回放就是了,而是由master_thread控制,差不多每秒钟更新一次

②回滚问题

回滚不是通过redo来回滚的,所有的page前滚到一个位置(恢复完),这些page对应的事务还是活跃的,还没提交,之后这些事务都会通过undo log来undo回滚,但undo是通过redo来恢复的

比如一个页120-160已经恢复过去了,但是这个事务需要回滚,却又已经刷到磁盘了,没关系,通过undo log往回滚一下就好了

事务活跃列表存放在undo段中,只要事务没提交就在里面,提交后移动到undo的history中,这个历史列表是用来做purge的,这里面的undo会被慢慢回收

III、checkpoint 分类

- Sharp Checkpoint
  将所有的赃页都刷新回磁盘,刷新时系统hang住,InnoDB关闭时使用
  相关参数：innodb_fast_shutdown={1|0}

- Fuzzy Checkpoint
  将部分脏页刷新回磁盘,对系统影响较小
  innodb_io_capacity来控制,最小限制为100,表示一次最多刷新脏页的能力,与IOPS相关
  SSD可以设置在4000-8000,SAS最多设置在800多（IOPS在1000左右）

IV、什么时候刷dirty page

- 以前在master thread线程中(从flush_list中进行刷新)

  现在都在page_cleaner_thread线程中(每一秒,每十秒)

- FLUSH_LRU_LIST 刷新

  5.5以前需要保证在LRU_LIST尾部要有100个空闲页（可替换的页）,即刷新一部分数据 ,保证有100个空闲页。

  由innodb_lru_scan_depth参数来控制,并不只是刷最后一个页,默认探测尾部1024个页（默认）,1024个页中所有脏页会一起刷掉,该参数是应用到每个Buffer Pool,总数即为该值乘以Buffer Pool的个数,总量超过innodb_io_capacity是不合理的,即此参数不得超过innodb_io_capacity/innodb_buffer_pool_instances,ssd的话,可以适当把这个扫描深度调深一点

- Async/Sync Flush Checkpoint
  重做日志重用

- Dirty Page too much
  赃页比例超过bp总量的一定比例,本来是通过page_cleaner_thread来刷,但是脏页太多了,就会强行刷,由innodb_max_dirty_pages_pct参数控制

tips：

①页只会从flush_list中刷新这个观点是不对的,只有page_cleaner_thread定期问flush_list要脏页,一个一个刷,刷到innodb_io_capacity的比例值

②LRU list中既存在干净的页也存在脏页,假设最后一个页,是脏的,另一个线程需要一个页,free list已经空了,lru会把这个页淘汰给这个线程去使用,这时候也需要刷新这个脏页,默认一下探测1024个page,把脏页刷掉

分类: MySQL

标签: mysql, 原理, checkpoint

海东潮
粉丝 - 62 关注 - 2
+加关注

« 上一篇： 缓冲池工作原理浅析
» 下一篇： MySQL重做日志相关

posted @ 2019-01-07 23:49 海东潮 阅读(1548) 评论(2) 收藏 举报

刷新页面 返回顶部

9. 缓冲池工作原理浅析(1)
10. __细看InnoDB数据落盘 图解 MYSQL(1)

推荐排行榜

1. 详细分析MySQL事务日志(redo log和undo log)(7)
2. MySQL 8.0窗口函数(5)
3. 灰度发布：灰度很简单，发布很复杂&灰度发布（灰度法则）的6点认识(2)
4. MySQL: OPTIMIZE TABLE: Table does not support optimize, doing recreate + analyze instead(2)
5. ORACLE DBA应该掌握的9个免费工具(2)

最新评论

1. Re:简单实现MySQL数据库的日志审计
头一次听说审计
--小菜pjy
2. Re:ps命令之排序
对我有用
--lizhenlzlz
3. Re:一个能够编写、运行SQL查询并可视化结果的Web应用：SqlPad
@只往前 你装上了吗...
--临冬城的狮子
4. Re:MySQL 8.0窗口函数
老哥，有sql语句吗
--法外诳图张三
5. Re:MySQL 8.0窗口函数
lag,lead写反了
--full233
6. Re:MySQL 8.0窗口函数
你好！，请问基于范围的动态窗口有什么例子吗，看的不是很明白，我在你们的书上也没有找到例子
--下海搬砖
7. Re:Linux man 命令详细介绍
内容有点乱，而且有的介绍明显看着有问题， 比如 -c 显示使用 cat 命令的手册信息。直接测试会报错，怎么可能是cat命令的信息。 建议整理下，方便看，主要也方便博主自己看！...
--findmoon
8. Re:关于MySQL checkpoint
跟姜老师讲的差不多
--安纳克里昂
9. Re:Linux内存管理（text、rodata、data、bss、stack&heap)
我在： 2021年 5月 13日 10:28:32 看过本篇博客!

编辑推荐：
· 记一次 C# 平台调用中因非托管 union 类型导致的内存访问越界
· ［EF Core］聊聊“复合”属性
· 那些被推迟的 C# 14 特性及其背后的故事
· 我最喜欢的 C# 14 新特性
· 程序员究竟要不要写文章

阅读排行：
· 遭遇疯狂 cc 攻击的一个周末
· C#/.NET/.NET Core技术前沿周刊 | 第 49 期（2025年8.1–8.10）
· 美丽而脆弱的天体运动：当C#遇见宇宙混沌
· 【EF Core】聊聊“复合”属性
· GPT-5 重磅发布

--努力变胖-HWP

10. Re:__细看InnoDB数据落盘 图解MYSQL

太强啦！！还有一个问题请教。MySQL的innodb_flush_method 5.7后是不是默认采用O_DIRECT？如果是，那么这么说MySQL数据库就会绕过【VFS】和【文件系统】，直接对磁盘进...

--Ethan3306

11. Re:一个能够编写、运行SQL查询并可视化结果的Web应用：SqlPad

博主又在本地运行过吗，对node不熟悉，只能docker启动了，但是我还是想本地运行，官方文档看不太明白。

--只往前

12. Re:MySQL: OPTIMIZE TABLE: Table does not support optimize, doing recreate + analyze instead

感谢

--一里天空

13. Re:关于MySQL checkpoint

这个篮色的字都不全啊

--王庆凡

14. Re:MySQL 8.0新特性之原子DDL

深入解析MySQL 8.0新特性：Crash Safe DDL：

--龙隆隆

15. Re:MySQL 8.0新特性之原子DDL

深入解析MySQL 8.0新特性：Crash Safe DDL：

--龙隆隆

16. Re:MySQL 8.0窗口函数

你好，介绍到cume_dist的函数是不是放错图了呀

--Cles

17. Re:缓冲池工作原理浅析

你这个人真的时有意思，专门抄袭，无耻下流卑鄙

--91洲际哥

18. Re:【MySQL】sysbench压测服务器及结果解读

@kun_行者 这人到处抄袭，也不署名一下...

--91洲际哥

19. Re:MySQL 8.0窗口函数

别名

--sun俊

20. Re:MySQL 8.0窗口函数

你好，谢谢你的分享。 但一个地方不明白：select * from (select row_number()over w as row_num, order_id,user_no,amount,cr...

--danica_string