12 | 为什么我的MySQL会“抖”一下?

2018-12-10 林晓斌

12 | 为什么我的MySQL会“抖”一下?
朗读人：林晓斌　15’37’’ | 14.30M

平时的工作中，不知道你有没有遇到过这样的场景，一条 SQL 语句，正常执行的时候特别快，但是有时也不知道怎么回事，它就会变得特别慢，并且这样的场景很难复现，它不只随机，而且持续时间还很短。

看上去，这就像是数据库“抖”了一下。今天，我们就一起来看一看这是什么原因。

你的 SQL 语句为什么变“慢”了

在前面第 2 篇文章《日志系统：一条 SQL 更新语句是如何执行的？》中，我为你介绍了 WAL 机制。现在你知道了，InnoDB 在处理更新语句的时候，只做了写日志这一个磁盘操作。这个日志叫作 redo log（重做日志），也就是《孔乙己》里咸亨酒店掌柜用来记账的粉板，在更新内存写完 redo log 后，就返回给客户端，本次更新成功。

做下类比的话，掌柜记账的账本是数据文件，记账用的粉板是日志文件（redo log），掌柜的记忆就是内存。

掌柜总要找时间把账本更新一下，这对应的就是把内存里的数据写入磁盘的过程，术语就是 flush。在这个 flush 操作执行之前，孔乙己的赊账总额，其实跟掌柜手中账本里面的记录是不一致的。因为孔乙己今天的赊账金额还只在粉板上，而账本里的记录是老的，还没把今天的赊账算进去。

当内存数据页跟磁盘数据页内容不一致的时候，我们称这个内存页为“脏页”。内存数据写入到磁盘后，内存和磁盘上的数据页的内容就一致了，称为“干净页”。

不论是脏页还是干净页，都在内存中。在这个例子里，内存对应的就是掌柜的记忆。

接下来，我们用一个示意图来展示一下“孔乙己赊账”的整个操作过程。假设原来孔乙己欠账 10 文，这次又要赊 9 文。

图 1 “孔乙己赊账”更新和 flush 过程

回到文章开头的问题，你不难想象，平时执行很快的更新操作，其实就是在写内存和日志，而 MySQL 偶尔“抖”一下的那个瞬间，可能就是在刷脏页（flush）。

那么，什么情况会引发数据库的 flush 过程呢？

我们还是继续用咸亨酒店掌柜的这个例子，想一想：掌柜在什么情况下会把粉板上的赊账记录改到账本上？

- 第一种场景是，粉板满了，记不下了。这时候如果再有人来赊账，掌柜就只得放下手里的活儿，将粉板上的记录擦掉一些，留出空位以便继续记账。当然在擦掉之前，他必须先将正确的账目记录到账本中才行。
  这个场景，对应的就是 InnoDB 的 redo log 写满了。这时候系统会停止所有更新操作，把 checkpoint 往前推进，redo log 留出空间可以继续写。我在第二讲画了一个 redo log 的示意图，这里我改成环形，便于大家理解。

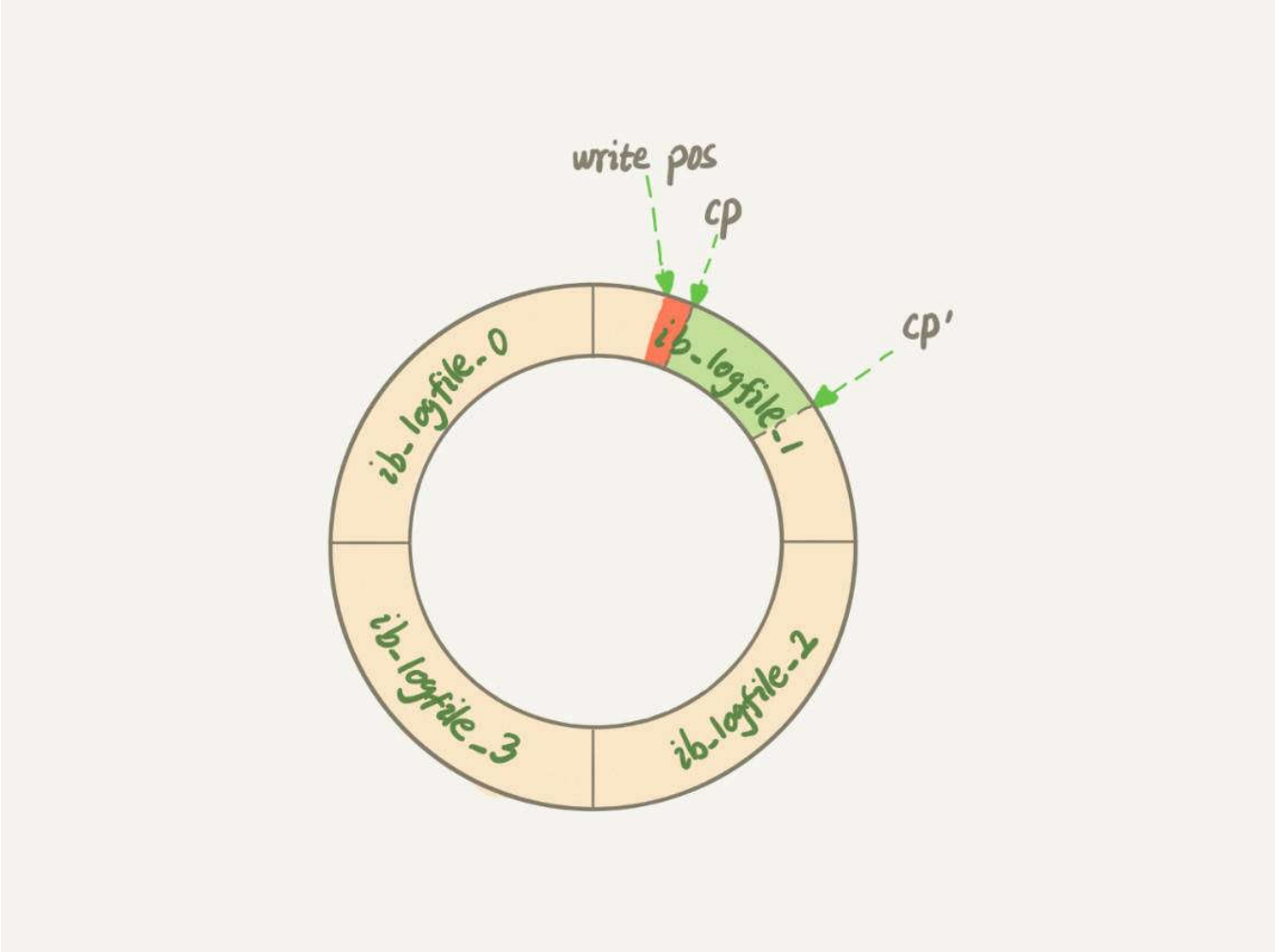

图 2 redo log 状态图

checkpoint 可不是随便往前修改一下位置就可以的。比如图 2 中，把 checkpoint 位置从 CP 推进到 CP’ ，就需要将两个点之间的日志（浅绿色部分），对应的所有脏页都 flush 到磁盘上。之后，图中从 write pos 到 CP’ 之间就是可以再写入的 redo log 的区域。

- 第二种场景是，这一天生意太好，要记住的事情太多，掌柜发现自己快记不住了，赶紧找出账本把孔乙己这笔账先加进去。

这种场景，对应的就是系统内存不足。当需要新的内存页，而内存不够用的时候，就要淘汰一些数据页，空出内存给别的数据页使用。如果淘汰的是“脏页”，就要先将脏页写到磁盘。

你一定会说，这时候难道不能直接把内存淘汰掉，下次需要请求的时候，从磁盘读入数据页，然后拿 redo log 出来应用不就行了？这里其实是从性能考虑的。如果刷脏页一定会写盘，就保证了每个数据页有两种状态：

  - 一种是内存里存在，内存里就肯定是正确的结果，直接返回；
  - 另一种是内存里没有数据，就可以肯定数据文件上是正确的结果，读入内存后返回。这样的效率最高。

- 第三种场景是，生意不忙的时候，或者打烊之后。这时候柜台没事，掌柜闲着也是闲着，不如更新账本。

  这种场景，对应的就是 MySQL 认为系统“空闲”的时候。当然，MySQL“这家酒店”的生意好起来可是会很快就能把粉板记满的，所以“掌柜”要合理地安排时间，即使是“生意好”的时候，也要见缝插针地找时间，只要有机会就刷一点“脏页”。

- 第四种场景是，年底了咸亨酒店要关门几天，需要把账结清一下。这时候掌柜要把所有账都记到账本上，这样过完年重新开张的时候，就能就着账本明确账目情况了。

  这种场景，对应的就是 MySQL 正常关闭的情况。这时候，MySQL 会把内存的脏页都 flush 到磁盘上，这样下次 MySQL 启动的时候，就可以直接从磁盘上读数据，启动速度会很快。

接下来，**你可以分析一下上面四种场景对性能的影响。**

其中，第三种情况是属于 MySQL 空闲时的操作，这时系统没什么压力，而第四种场景是数据库本来就要关闭了。这两种情况下，你不会太关注“性能”问题。所以这里，我们主要来分析一下前两种场景下的性能问题。

第一种是“redo log 写满了，要 flush 脏页”，这种情况是 InnoDB 要尽量避免的。因为出现这种情况的时候，整个系统就不能再接受更新了，所有的更新都必须堵住。如果你从监控上看，这时候更新数会跌为 0。

第二种是“内存不够用了，要先将脏页写到磁盘”，这种情况其实是常态。**InnoDB 用缓冲池（buffer pool）管理内存，缓冲池中的内存页有三种状态：**

- 第一种是，还没有使用的；
- 第二种是，使用了并且是干净页；
- 第三种是，使用了并且是脏页。

InnoDB 的策略是尽量使用内存，因此对于一个长时间运行的库来说，未被使用的页面很少。

而当要读入的数据页没有在内存的时候，就必须到缓冲池中申请一个数据页。这时候只能把最久不使用的数据页从内存中淘汰掉：如果要淘汰的是一个干净页，就直接释放出来复用；但如果是脏页呢，就必须将脏页先刷到磁盘，变成干净页后才能复用。

所以，刷脏页虽然是常态，但是出现以下这两种情况，都是会明显影响性能的：

1. 一个查询要淘汰的脏页个数太多，会导致查询的响应时间明显变长；
2. 日志写满，更新全部堵住，写性能跌为 0，这种情况对敏感业务来说，是不能接受的。

所以，InnoDB 需要有控制脏页比例的机制，来尽量避免上面的这两种情况。

InnoDB 刷脏页的控制策略

接下来，我就来和你说说 InnoDB 脏页的控制策略，以及和这些策略相关的参数。

首先，你要正确地告诉 InnoDB 所在主机的 IO 能力，这样 InnoDB 才能知道需要全力刷脏页的时候，可以刷多快。

这就要用到 innodb_io_capacity 这个参数了，它会告诉 InnoDB 你的磁盘能力。这个值我建议你设置成磁盘的 IOPS。磁盘的 IOPS 可以通过 fio 这个工具来测试，下面的语句是我用来测试磁盘随机读写的命令：

```
fio -filename=$filename -direct=1 -iodepth 1 -thread -rw=randrw -ioengine=psync -bs=16k -size=5
```

复制代码

其实，因为没能正确地设置 innodb_io_capacity 参数，而导致的性能问题也比比皆是。之前，就曾有其他公司的开发负责人找我看一个库的性能问题，说 MySQL 的写入速度很慢，TPS 很低，但是数据库主机的 IO 压力并不大。经过一番排查，发现罪魁祸首就是这个参数的设置出了问题。

他的主机磁盘用的是 SSD，但是 innodb_io_capacity 的值设置的是 300。于是，InnoDB 认为这个系统的能力就这么差，所以刷脏页刷得特别慢，甚至比脏页生成的速度还慢，这样就造成了脏页累积，影响了查询和更新性能。

虽然我们现在已经定义了“全力刷脏页”的行为，但平时总不能一直是全力刷吧？毕竟磁盘能力不能只用来刷脏页，还需要服务用户请求。所以接下来，我们就一起看看 InnoDB 怎么控制引擎按照“全力”的百分比来刷脏页。

根据我前面提到的知识点，试想一下，**如果你来设计策略控制刷脏页的速度，会参考哪些因素呢？**

这个问题可以这么想，如果刷太慢，会出现什么情况？首先是内存脏页太多，其次是 redo log 写满。

所以，InnoDB 的刷盘速度就是要参考这两个因素：一个是脏页比例，一个是 redo log 写盘速度。

InnoDB 会根据这两个因素先单独算出两个数字。

参数 innodb_max_dirty_pages_pct 是脏页比例上限，默认值是 75%。InnoDB 会根据当前的脏页比例（假设为 M），算出一个范围在 0 到 100 之间的数字，计算这个数字的伪代码类似这样：

复制代码

```
F1(M)
{
  if M>=innodb_max_dirty_pages_pct then
      return 100;
  return 100*M/innodb_max_dirty_pages_pct;
}
```

InnoDB 每次写入的日志都有一个序号，当前写入的序号跟 checkpoint 对应的序号之间的差值，我们假设为 N。InnoDB 会根据这个 N 算出一个范围在 0 到 100 之间的数字，这个计算公式可以记为 F2(N)。F2(N) 算法比较复杂，你只要知道 N 越大，算出来的值越大就好了。

然后，**根据上述算得的 F1(M) 和 F2(N) 两个值，取其中较大的值记为 R，之后引擎就可以按照 innodb_io_capacity 定义的能力乘以 R% 来控制刷脏页的速度。**

上述的计算流程比较抽象，不容易理解，所以我画了一个简单的流程图。图中的 F1、F2 就是上面我们通过脏页比例和 redo log 写入速度算出来的两个值。

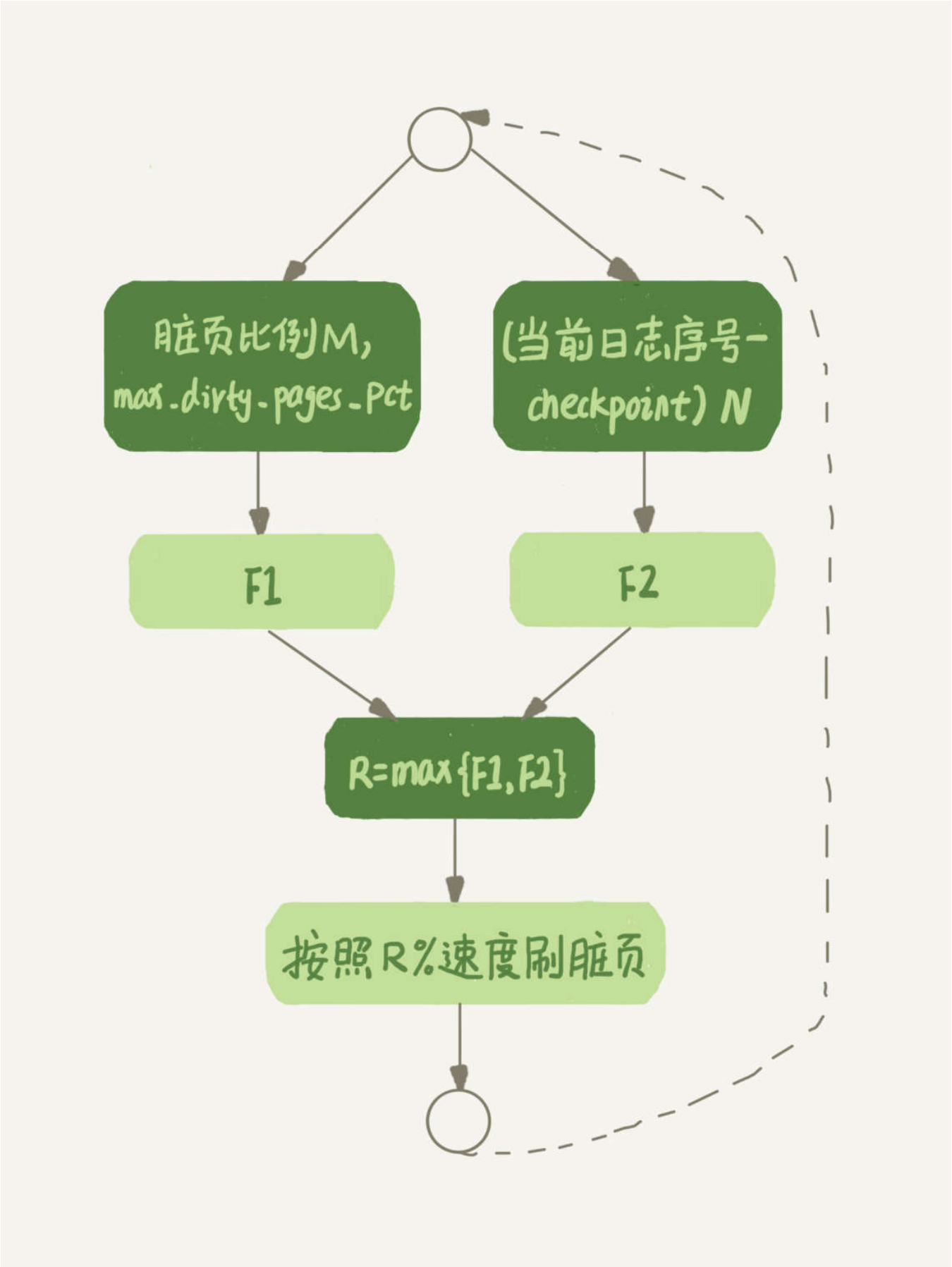

图 3 InnoDB 刷脏页速度策略

现在你知道了，InnoDB 会在后台刷脏页，而刷脏页的过程是要将内存页写入磁盘。所以，无论是你的查询语句在需要内存的时候可能要求淘汰一个脏页，还是由于刷脏页的逻辑会占用 IO 资

源并可能影响到了你的更新语句，都可能是造成你从业务端感知到 MySQL“抖”了一下的原因。

要尽量避免这种情况，你就要合理地设置 innodb_io_capacity 的值，并且**平时要多关注脏页比例，不要让它经常接近 75%**。

其中，脏页比例是通过 Innodb_buffer_pool_pages_dirty/Innodb_buffer_pool_pages_total 得到的，具体的命令参考下面的代码：

```
mysql> select VARIABLE_VALUE into @a from global_status where VARIABLE_NAME = 'Innodb_buffer_poo
select VARIABLE_VALUE into @b from global_status where VARIABLE_NAME = 'Innodb_buffer_pool_pages
select @a/@b;
```

复制代码

接下来，我们再看一个有趣的策略。

一旦一个查询请求需要在执行过程中先 flush 掉一个脏页时，这个查询就可能要比平时慢了。而 MySQL 中的一个机制，可能让你的查询会更慢：在准备刷一个脏页的时候，如果这个数据页旁边的数据页刚好是脏页，就会把这个“邻居”也带着一起刷掉；而且这个把“邻居”拖下水的逻辑还可以继续蔓延，也就是对于每个邻居数据页，如果跟它相邻的数据页也还是脏页的话，也会被放到一起刷。

在 InnoDB 中，innodb_flush_neighbors 参数就是用来控制这个行为的，值为 1 的时候会有上述的“连坐”机制，值为 0 时表示不找邻居，自己刷自己的。

找“邻居”这个优化在机械硬盘时代是很有意义的，可以减少很多随机 IO。机械硬盘的随机 IOPS 一般只有几百，相同的逻辑操作减少随机 IO 就意味着系统性能的大幅度提升。

而如果使用的是 SSD 这类 IOPS 比较高的设备的话，我就建议你把 innodb_flush_neighbors 的值设置成 0。因为这时候 IOPS 往往不是瓶颈，而“只刷自己”，就能更快地执行完必要的刷脏页操作，减少 SQL 语句响应时间。

在 MySQL 8.0 中，innodb_flush_neighbors 参数的默认值已经是 0 了。

小结

今天这篇文章，我延续第 2 篇中介绍的 WAL 的概念，和你解释了这个机制后续需要的刷脏页操作和执行时机。利用 WAL 技术，数据库将随机写转换成了顺序写，大大提升了数据库的性能。

但是，由此也带来了内存脏页的问题。脏页会被后台线程自动 flush，也会由于数据页淘汰而触发 flush，而刷脏页的过程由于会占用资源，可能会让你的更新和查询语句的响应时间长一些。在文章里，我也给你介绍了控制刷脏页的方法和对应的监控方式。

文章最后，我给你留下一个思考题吧。

一个内存配置为 128GB、innodb_io_capacity 设置为 20000 的大规格实例，正常会建议你将 redo log 设置成 4 个 1GB 的文件。

但如果你在配置的时候不慎将 redo log 设置成了 1 个 100M 的文件，会发生什么情况呢？又为什么会出现这样的情况呢？

你可以把你的分析结论写在留言区里，我会在下一篇文章的末尾和你讨论这个问题。感谢你的收听，也欢迎你把这篇文章分享给更多的朋友一起阅读。

上期问题时间

上期我留给你的问题是，给一个学号字段创建索引，有哪些方法。

由于这个学号的规则，无论是正向还是反向的前缀索引，重复度都比较高。因为维护的只是一个学校的，因此前面 6 位（其中，前三位是所在城市编号、第四到第六位是学校编号）其实是固定的，邮箱后缀都是 @gamil.com，因此可以只存入学年份加顺序编号，它们的长度是 9 位。

而其实在此基础上，可以用数字类型来存这 9 位数字。比如 201100001，这样只需要占 4 个字节。其实这个就是一种 hash，只是它用了最简单的转换规则：字符串转数字的规则，而刚好我们设定的这个背景，可以保证这个转换后结果的唯一性。

评论区中，也有其他一些很不错的见解。

评论用户 @封建的风 说，一个学校的总人数这种数据量，50 年才 100 万学生，这个表肯定是小表。为了业务简单，直接存原来的字符串。这个答复里面包含了“优化成本和收益”的思想，我觉得值得 at 出来。

@小潘 同学提了另外一个极致的方向。如果碰到表数据量特别大的场景，通过这种方式的收益是很不错的。

评论区留言点赞板：

> @lttzzlll，提到了用整型存“四位年份 + 五位编号”的方法；
> 由于整个学号的值超过了 int 上限，@老杨同志 也提到了用 8 个字节的 bigint 来存的方法。

上一篇　11 | 怎么给字符串字段加索引?

下一篇　13 | 为什么表数据删掉一半，表文件大小不变?

写留言

精选留言

Tony Du　9

当内存不够用了，要将脏页写到磁盘，会有一个数据页淘汰机制（最久不使用），假设淘汰的是脏页，则此时脏页所对应的redo log的位置是随机的，当有多个不同的脏页需要刷，则对应的redo log可能在不同的位置，这样就需要把redo log的多个不同位置刷掉，这样对于redo log的处理不是就会很麻烦吗？（合并间隙，移动位置？）

另外，redo log的优势在于将磁盘随机写转换成了顺序写，如果需要将redo log的不同部分刷掉（刷脏页），不是就在redo log里随机读写了么?

2018-12-10

作者回复

好问题。

其实由于淘汰的时候，刷脏页过程不用动redo log文件的。

这个有个额外的保证，是redo log在“重放”的时候，如果一个数据页已经是刷过的，会识别出来并跳过。

2018-12-10

某、人 6

redo log是关系型数据库的核心啊,保证了ACID里的D。所以redo log是牵一发而动全身的操作
按照老师说的当内存数据页跟磁盘数据页不一致的时候,把内存页称为'脏页'。如果redo log设置得太小,redo log写满.那么会涉及到哪些操作呢,我认为是以下几点:
1.把相对应的数据页中的脏页持久化到磁盘,checkpoint往前推
2.由于redo log还记录了undo的变化,undo log buffer也要持久化进undo log
3.当innodb_flush_log_at_trx_commit设置为非1,还要把内存里的redo log持久化到磁盘上
4.redo log还记录了change buffer的改变,那么还要把change buffer purge到idb以及merge change buffer.merge生成的数据页也是脏页,也要持久化到磁盘
上述4种操作,都是占用系统I/O,影响DML,如果操作频繁,会导致'抖'得向现在我们过冬一样。但是对于select操作来说,查询时间相对会更快。因为系统脏页变少了,不用去淘汰脏页,直接复用
干净页即可。还有就是对于宕机恢复,速度也更快,因为checkpoint很接近LSN,恢复的数据页相对较少
所以要控制刷脏的频率,频率快了,影响DML I/O,频率慢了,会导致读操作耗时长。
我是这样想的这个问题,有可能不太对,特别是对于第4点是否会mergel以及purge,还需要老师的解答

2018-12-10

作者回复

抖得像过冬一样，😄👍

你说得很对，第4点没错的，出现这种情况的时候，连change buffer的优化也没意义了

2018-12-11

yesir 1

我观察了下公司的数据库确实发现了抖动现象，有几个问题，
1）Innodb_buffer_pool_pages_total这个值很大，百万级别的，而且数值不像是人为设置上去的，是怎么来的呢?
2）Innodb_buffer_pool_pages_dirty达到4万多的时候就开始flush了，脏页比例是75，这肯定是远达不到的，ssd磁盘，innodb_io_capacity是200，肯定可以提高。文章中说flush的触发条件有2个，一个是内存不够了，一个是redo log 满了，那么我这个场景是哪种情况呢

2018-12-11

作者回复

1） 这个是innodb 数据页总是，过百万是正常的，16K一个，Bufree pool size 16G 就是100万了

2）你这个例子就是io_capacity设太小了...

2018-12-12

Ryoma 0

关于粉板和redo log的类比我觉得有一点不太合适：redo log记录的是实时欠款，比如账本中是10文，又欠了9文，此时redo log 记录的是19；而粉板的话，只会追加某人欠款+9文，不会关注原来已欠款多少（不然某人赊账时，我还需要找到账本中的这个人，才知道他之前欠款多少，我觉得这个场景跟MySQL中的场景还是有区别的）

2018-12-12

作者回复

Redo log里也是记的+9哦😁

2018-12-12

岁月安然 4

“内存不够用了，要先将脏页写到磁盘“和“redo log 写满了，要 flush 脏页”可以理解为一个脏页本身占用内存，释放内存需要将脏页写入到磁盘才能释放。而redo log写满只有当redo log对应的脏页flush到磁盘上才能释放对应空间。有几个问题：

1、“内存不够用了，要先将脏页写到磁盘“redo log对应的空间会释放嘛？“redo log 写满了，要 flush 脏页”对应的内存页会释放嘛？

2、将脏页flush到磁盘上是直接将脏页数据覆盖到对应磁盘上的数据？还是从磁盘上取到数据后取根据redo log记录进行更新后再写入到磁盘？

3、redo log是怎么记录对应脏页是否已经flush了？如果断电了重启导致内存丢失，前面几章说通过redo log进行数据恢复那redo log又怎么去释放空间？

2018-12-10

作者回复

1. Redolog 的空间是循环使用的，无所谓释放。 对应的内存页会变成干净页。但是等淘汰的时候才会逐出内存

2. 好问题，前者

3. 不用记，重启了就从checkpoint 的位置往后扫。 如果已经之前刷过盘的, 不会重复应用redi log。 好问题

2018-12-10

melon 3

又思考了一下，请老师帮忙看一下理解的对不对：buffer pool里维护着一个脏页列表，假设现在redo log 的 checkpoint 记录的 LSN 为 10，现在内存中的一干净页有修改，修改后该页的LSN为12，大于 checkpoint 的LSN，则在写redo log的同时该页也会被标记为脏页记录到脏页列表中，现在内存不足，该页需要被淘汰掉，该页会被刷到磁盘，磁盘中该页的LSN为12，该页也从脏页列表中移除，现在redo log 需要往前推进checkpoint，到LSN为12的这条log时，发现内存中的脏页列表里没有该页，且磁盘上该页的LSN也已经为12，则该页已刷脏，已为干净页，跳过。

2018-12-11

作者回复

对的。👍

2018-12-11

jimmy 3

老师，我想问一下，innodb是如何知道一个页是不是脏页的，是有标记位还是通过redolog的ckeckpoint来确定的?

2018-12-10

作者回复

每个数据页头部有LSN，8字节，每次修改都会变大。

对比这个LSN跟checkpoint 的LSN，比checkpoint小的一定是干净页

2018-12-10

Ying 3

redolog 设置过小，会导致频繁刷脏页，还可能引发连坐，这样抖的频率可能会明显变高，系统会不断卡死的

2018-12-10

Tony Du 2

当内存不够用了，要将脏页写到磁盘，会有一个数据页淘汰机制（最久不使用），假设淘汰的是脏页，则此时脏页所对应的redo log的位置是随机的，当有多个不同的脏页需要刷，则对应的redo log可能在不同的位置，这样就需要把redo log的多个不同位置刷掉，这样对于redo log的处理不是就会很麻烦吗？（合并间隙，移动位置？）
另外，redo log的优势在于将磁盘随机写转换成了顺序写，如果需要将redo log的不同部分刷掉（刷脏页），不是就在redo log里随机读写了么?

作者回复
好问题。

其实由于淘汰的时候，刷脏页过程不用动redo log文件的。

这个有个额外的保证，是redo log在“重放”的时候，如果一个数据页已经是刷过的，会识别出来并跳过。

我的回复
这个额外保证是如何做到的？能不能稍微解释下
通过刷脏页时数据页更新的timestamp来对比redo log的timestamp?

2018-12-11

作者回复

LSN，每次写redo log都带的一个数字， 数据页上也有，对比大小的，因为太细节没有写到文章中。

2018-12-11

lionetes 1

很多测试人员再做压力测试的时候 出现刚开始 insert update 很快 一会 就出现很慢,并且延迟很大，大部分是因为redo log 设置太小 引起的,完美诠释

2018-12-11

作者回复

常见的误用场景

2018-12-11

lnon 1

老师您好，一直有个问题，想请教下：就是远程通过JDBC一次查询可能查询出上万条数据。但我在客户端迭代读出来的时候都是一条一条取的。我想问，数据库端一次查询，会将查询的上万条真实数据结果一次性放在一个缓冲中，等客户端来取。还是构建一个查询数据结构，只是定位查询结果位置的，每次客户端通过游标通过查询结果位置实时读磁盘取数据的?

2018-12-10

克己过 1

老师！！！fio这条命令是会破坏硬盘的！而且百度搜出来不加硬盘坏关键词去搜，搜出来的文章没有一篇会告诉你这个事情！！！不说了，我去恢复数据了😭

2018-12-10

作者回复

没有吧，怎么会破坏硬盘？我和以前同事一直这么用的呀...

你确定是这个命令导致的吗😕

2018-12-12

信信 1

老师，你好，针对上一期问题的解答中有个疑问：为啥201100001要从字符串转数字，本来只需要2字节，转成数字需要4字节。是因为相同的内容，数字查找比字符串查找有优势吗?

2018-12-10

作者回复

为什么说这是两个字节？按字符一个个算是9个

2018-12-10

阿建 1

redo-log内存满了，不停的要刷脏页回磁盘。现象就会是发现机器io不高，但是mysql明显的卡顿。

2018-12-10

峰 1

redo日志设置太小，内存又比较大，导致innodb缓存的脏页还没多少就开始大量flush，刷写频率增大。感觉有点像jvm中，年轻代内存设置小点，导致频繁younggc。当然这就是个权衡，毕竟redo和内存不能无限大。

2018-12-10

作者回复

👍引申不错

2018-12-10

夹心面包 1

老师请问下 innodb_io_capacity我们默认是200,在高并发插入场景下,cpu使用率很低,但是cpu iowait高,这种情况下 调大 innodb_io_capacity 是否有用呢,我用了您的测试方法,iops随机读写都为1K

2018-12-10

作者回复

那说明你们磁盘不太好...

2018-12-10

董航 1

怎么说呢，越看越明白，哈哈，前面不懂的，后面就懂了

2018-12-10

作者回复

👍

你领会到实践篇的“奥义”了😄

一边引入新知识点，一边应用前面的

2018-12-10

Ryoma 0

如果redo log也是+9，那原图中的19是不是有问题呢

2018-12-12

Leon📷 0

fush脏页是略过redo-log，直接到磁盘，算是把mysql逼急的应急状态吧，正常情况下还是后台线程慢条斯理的慢慢从redo-log同步到磁盘

2018-12-12

作者回复

淘汰脏页是常态，一般不多就还好

2018-12-12

Leon📷 0

老师，是否可以理解redo_log就是内存buffer-pool的脏页的高速缓存，redo-log就是脏页的镜像，一荣俱荣一损俱损，最终也就是要高速缓存的数据刷新到磁盘。所谓后台的线程刷新其实还是把redo-log的操作还原成数据，然后写进ibd，所谓抖了一下，就是buffer-pool一抖带着redo-log一起抖，连带着磁盘也跟着有刷新动作

2018-12-12

作者回复

前面这个描述不好，容易造成误解。 redo log不是脏页的镜像，当数据页是脏页的时候，redo log 其实是 “磁盘页到内存页的diff”

2018-12-12

发条橙子 。 0

总结

innodb_io_capacity 决定刷新缓存的频率 当值小， 缓存数据不断累积， 刷新少， 脏页会越来越多 。当脏页累计会影响更新 和 查询

影响更新： 因为缓冲池内存不够，下次查询需要将历史脏页刷新 耗费时间

影响查询： 查询数据页不在缓冲池，所以走磁盘并更新数据页， 因为缓冲池内存不够，下次查询需要将历史脏页刷新 耗费时间

问题：

1: 系统内存不够 和 缓冲池内存的关系是？ 缓冲池可以无限扩大内存将系统内存沾满么?

2: 更新操作只是 更新redo log 更新磁盘 加 更新内存 。 可是redo log 也有缓冲区， 是不是会存在都是操作内存，不会操作磁盘的更新的情况

2018-12-12

作者回复

1. 好问题。不能扩大，buffer pool size 设定好就只能用这么多，不会再多吃系统内存

2. Innodb_flush_log_at_trx_commit 设置为0的时候就是

2018-12-12

老鱼头 0

那个说fio命令会破坏硬盘的兄弟，是没用对命令。估计把-filename=/dev/sdb1 。。。这个的意思是从 分区 sdb1 的第一个扇区开始写入随机数据，去判断这个磁盘的写入速度。如果指定路径+文件名就不会出这事了~比如老师给的例子~

2018-12-12

作者回复

嗯嗯，你说的对

写文章的时候，我还故意用变量，这样直接拷贝会出错，然后自己再写个路径，已经考虑了安全了😉

2018-12-12

BD 0

老师请问下，在一个事务回滚操作里如果有DDL操作会怎样。例如如下步骤:1.开始事务及设置手动提交 2修改表Aid为2的行 3动态创建一张表B 4修改表C里id为5的行 6提交事务 及回滚代码编写。如果在步骤5时发生异常数据库会怎样做

2018-12-11

作者回复

第三步会自动提交事务的...

如果你是autocommit=1的话，其实4也是单独事务提交了😉

2018-12-11

风萧雨瑟 0

老师帮忙分析一个问题：
刚搭建的一个新的从库，没有负载，机器规格和参数都是和其它从库一样的。但其从库系统是centos6.有问题这台是centos7。innodb_flush_log_at_trx_commit 当值设置为1的时候(其它从库正常)，复制延迟增大，iostat查看结果如下：
Device r/s w/s rMB/s wMB/s rrqm/s wrqm/s %rrqm %wrqm r_await w_await aqu-sz rareq-sz wareq-sz svctm %util
sda1 4.50 369.00 0.07 3.80 0.00 0.00 0.00 0.00 1.67 0.77 0.29 16.00 10.54 0.70 26.05
其它正常的从库：
Device: rrqm/s wrqm/s r/s w/s rMB/s wMB/s avgrq-sz avgqu-sz await svctm %util
sda1 0.00 0.00 18.00 451.50 0.28 4.60 21.31 0.08 0.17 0.12 5.65
但当innodb_flush_log_at_trx_commit 设置为2的时候就正常了。 ssd为独占。 fio测试硬盘iops与其它从库区别不大。机器上两块硬盘，data目录换到另一块上，问题一样。
通过perf分析信息片断：
- 100.00% 0.00% mysqld libpthread-2.17.so [.] start_thread
- start_thread
- 63.21% io_handler_thread
- 63.21% fil_aio_wait
- 45.23% os_aio_handler
- 28.35% os_file_write_page
- 25.05% 0xf0c3
- 25.05% system_call_fastpath
- 25.05% sys_pwrite64
- 25.05% vfs_write
- 25.05% do_sync_write
- 18.44% xfs_file_aio_write
- 18.44% xfs_file_dio_aio_write
- 18.44% __blockdev_direct_IO
- 18.44% do_blockdev_direct_IO
- 7.11% io_schedule
- io_schedule_timeout
- 7.11% schedule_timeout
- 7.11% schedule

- 7.11% __schedule
- 4.17% deactivate_task
- 4.17% dequeue_task
- 4.17% dequeue_task_fair
4.17% dequeue_entity
- 2.94% call_function_single_interrupt
- 2.94% smp_call_function_single_interrupt
irq_exit
do_softirq
call_softirq
__do_softirq
rebalance_domains
top信息
%Cpu19 : 3.4 us, 2.0 sy, 0.0 ni, 58.4 id, 36.2 wa, 0.0 hi, 0.0 si, 0.0 st
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
111111 mysql 20 0 21.4g 17.9g 15020 S 17.2 28.7 4538:15 mysqld
通过这些看的话象是卡在写这。但参数值不一样，差别很大。innodb方面参数如下：
read_io_threads/write_io_threads = 12
io_capacity = 4000
thread_concurrency = 48
commit_concurrency= 48
flush_neighbors = 0
flush_method=O_DIRECT
purge_threads=4
请老师指点怎么排查

2018-12-11

作者回复

你fio工具测试一下放redolog的盘，分别测试一下随机写性能和顺序写性能

2018-12-11

算不出流源 0

思考题:redo log较小，此时F2(N)就容易较大，后台刷脏页的速度就会较快，但是机器iops很高，所以即使在mysql负载较高的情况下，也不会造成redo log写满，因为innodb会照着最大iops的目标去请求io，而读写数据库的业务请求也机器朝最大iops能力去挤，最终两类io需求相互竞争达到一个动态平衡，redo log不会被灌满，但业务请求也没有空闲的时候执行速度快。
所以我认为应该不是造成抖动，而是业务负载较大的情况下，完成业务请求的平均时间会较系统空闲时加倍。不知道理解有没有问题，请老师指正

2018-12-11

作者回复

这个“动态平衡”确实是用“间歇性地卡住业务请求”作为代价的

2018-12-11

算不出流源

0

老师，请问check point何时会推进？难道是只有在redo log写满时或者停止mysql的时候才会推进，而平时innodb后台刷脏页的时候并不会推进吗？

2018-12-11

作者回复

会的，文中说的算io_capcity*R% 就算控制后台刷的速度的

2018-12-11

melon

0

老师，您在回复 jimmy 的问题时提到通过对比数据页的 LSN 和 checkpoint 的 LSN，比checkpoint LSN 小的就是干净页，那在重放 redo log 时是如何跳过那些由于内存不足而刷脏的页面的呢？刷脏的时候会将 LSN 重置为0之类的？

2018-12-11

作者回复

判断一下发现是干净页就跳过。

不是重置为0，是把磁盘上数据页的LSN也一起改掉了

2018-12-11

天王

0

1redo log满了和内存不够了，都会触发脏页flush到磁盘。脏页和干净页区别，是否刷新到磁盘。relog 满了，会checkpoint，停止更新操作，把脏页刷到磁盘。内存不足也会把很久不使用的刷新到磁盘，如果里面包括有脏页，要触发flush操作。后台有一个线程专门做刷脏页的工作，是根据设置的比例来刷的。内存刷脏页，redo log会打上刷过的标记。relog刷脏页，内存中也会把脏页标记为干净页。

2018-12-11

天王

0

增删改操作会变慢。因为redo log设置比较小，容易满，满了会checkpoint，寻找新的空间，频繁的把redo log的数据刷到磁盘。checkpoint的时候会停止更新操作，所以会变慢

2018-12-11

黄楚门的世界

0

"一个内存配置为 128GB、innodb_io_capacity 设置为 20000 的大规格实例，正常会建议你将 redo log 设置成 4 个 1GB 的文件。"

"redo log 设置成 4 个 1GB 的文件"，这个建议是怎么算出来的？

2018-12-10

作者回复

其实这里意思是，“用了好机器”😄

2018-12-11

斜面镜子 Bill

0

要分业务场景来看

1. 如果业务以写为主或者读写流量差不多，因为读cache相对较大，我理解会频繁出现因日

志写满，更新全部堵住，写性能跌为 0的情况
2. 如果以读为主，我理解redo_log本身设置小并没有明显的影响

2018-12-10

斜面镜子 Bill

0

要分业务场景来看
1. 如果业务以写为主或者读写流量差不多，因为读cache相对较大，我理解会频繁出现因日志写满，更新全部堵住，写性能跌为 0的情况
2. 如果以读为主，我理解redo_log本身设置小并没有明显的影响

2018-12-10

张永志

0

100M的redo很容易写满，系统锁死，触发检查点推进，导致写操作卡住。由于主机IO能力很强，检查点会很快完成，卡住的写操作又很快可以执行。循环往复，现象就是写操作每隔一小段时间执行就会变慢几秒。

2018-12-10

作者回复

准确

2018-12-10

didiren

0

用fio压测的结果，读写的iops是一样的，那么io_capacity按照这个值设，还是设为读写相加

2018-12-10

作者回复

一般参考写能力

2018-12-10

梁中华

0

接着Tony Du的问题讨论，脏页被淘汰的时候，只是redo log上checkpoint指针前移，应该不存在redo log随机读写的问题。

2018-12-10

作者回复

也不会马上就前移动哈

2018-12-10

Bennet

0

老师，一直有个疑问：看了这几期的文章，我理解 redolog 和 undolog 是在有数据更新时同时写的，redolog 作为 crash-safe 机制，每次全部读入做恢复就好。但是像在做脏页flush或者 mvcc版本回退的时候，难道是按照涉及的数据行来定位日志行的吗？如果是，是怎么做到这种“定位”的呢？如果不是，InnoDB 的机制又是怎样的？

2018-12-10

作者回复

Flush 和 mvcc不能放一起说哈，不一样的。

每一行存了一个位置可以直接找到undo log, 这个MVCC的基础

Redolog里面记录了“这是哪个页面的修改”

2018-12-10

崔根禄 0

老师，我又把第02看了一遍

有个疑问：

根据WAL机制，当记录需要更新时，先写redo，然后更新内存，生成脏页，这时候更新就算完成，之后在空闲时刷脏页到磁盘。

1.那么一个含有update的事务，在执行commit时，是不是就触发刷脏页的操作?

2.如果事务中，先update，后select，最后commit；那么update生成脏页，还未刷磁盘，select读到更新之后的数据，是从哪儿读?

2018-12-10

作者回复

1. 不会触发,不需要触发的

2. 内存中

2018-12-10

慕塔 0

再请教另一个问题啊 机器内存是16G,将设置为nnodb_buffer_pool_size=100G，竟然能启动成功！！！我记得bp在系统启动的时候是事先分配好的啊

2018-12-10

作者回复

是，但是如果你数据和压力跑起来,很快就OOM了

2018-12-10

石建磊 0

老师好，对于写入的数据来说，内存中的记录和redo log中的记录是对应的吗？还是说比如redo log的记录已经少了， 但是数据还在内存中？或者说内存中数据少了，redo log中还有记录?

2018-12-10

作者回复

我没理解你说的“对应”的意思。

内存被淘汰，可能redo log还在的；

而即使checkpoint 往前推进了，只是把脏页变成干净页，内存里这个数据页也还在的

2018-12-10

没时间了ngu 0

老师好，小白请教下：
上述201100001一共9位数字，Int，4位就可以存储，是因为2^31-1=2147483647 ，这个能理解。
但是，Int设置了大于4位的长度，最大值还是这个值。假如长度设置11， 和长度4位感觉没区别呀。
请问，是Int本身的机制还是怎么回事呀?

2018-12-10

作者回复

那个（11）不是长度的意思... int 固定四个字节的

2018-12-10

慕塔 0

源码中有计算redo文件大小，有好几个阈值判断，以lsn的差值为判断依据，不同的阈值会导致同步异步checkpoint,同步异步刷脏页。如果page clean刷脏不足，太小会导致用户线程刷脏，single page flush，用户线程等待，频繁做checkpoint，sysbench的表现是tps掉的厉害。page clean优化点有很多，percona的优化不错，专门一个线程做lru链刷脏。
我想进一步问下啊，1.刷脏页是写os cache啊还是disk？2.再一个就是写redo，是不是也是调fil io先写os cache,然后调用fsync刷到disk，这两个接口也没看到是如何同步的(虽说都是在log_write_up_to下)，事务提交时，那个参数1又是如何保证redo落盘？好多接口没有返回值，看的有点晕😁 3.我在物理机和虚拟机下分别部署mysql(其他配置都一样，内存 cpu，测试环境)，虚拟机比物理机的tps(insert)高5倍，很不能理解啊😭大佬能指点不 谢谢！

2018-12-10

作者回复

1. 一般建议是直接设置o_direct ， 直接写文件
2. Fio带fsync就是直接落盘；
3. 你试下把物理机的双1改掉试试

2018-12-10

极客童 0

梳理一下之前的知识，存储引擎接到一个更新操作时，操作过程是：
1、引擎将这行新数据更新到change buffer中，同时将这个更新操作记录到redo log 里面；
2、此时 redo log 处于 prepare 状态，再生成写入binlog；
3、最后把redo log commit。change buffer找时机merge到buffer pool。

对本题目，如果redo log过小，会频繁触发purge操作，purge操作即buffer pool刷脏页的同时把redolog的checkpoint往前移动，导致频繁的磁盘io，结果就是mysql频繁的抖动。我的理解对吗?

2018-12-10

作者回复

1. 这个要说明下，change buffer只对非唯一索引有效

2018-12-10

范永康 0

redo log 固定大小，存在于什么位置：内存、磁盘或者缓存中？

2018-12-10

作者回复

不是说文件？文件肯定在磁盘

2018-12-10

小确幸 0

有两个疑问：

1、数据库的页，与书本的页在概念上是类似的吗？

2、在内存中的页数据，与查询缓存有什么区别？

2018-12-10

作者回复

1. 数据页是“数据的存储单位”，每次要读写都是整页读入内存的。

2. 这里说的内存是innodb buffer pool, 查询缓存是server层的，存的是SQL语句的查询结果

2018-12-10

Gavin 0

文章中说“InnoDB在处理更新语句的时候，只做了写日志这一个磁盘操作。这个日志叫做 redo log（重做日志）”，我想问下斌哥，undo log 的持久化时机是什么时候或者能详细说下 undo log 的刷新机制吗？

2018-12-10

作者回复

Undo log跟这文章中说的数据一样

2018-12-10

算不出流源 0

老师，请问同一个事务中有多条更新语句时，是每一条更新语句都走第2讲中说的写redo log prepare -->binlog-->redo log commit的步骤，还是整个事务只走一遍这个步骤？如果是每条语句各走一遍，也就是说redo log commit和事务的commit并不对应，那事务的整体性在redo log中是怎么体现的？

2018-12-10

作者回复

只走一遍

2018-12-10

lemon 0

一个内存配置为 128GB、innodb_io_capacity 设置为 20000 的大规格实例，正常会建议你将 redo log 设置成 4 个 1GB 的文件。

请问这三个参数的设置有某种计算关系么？您的这个推荐有没有对业务场景的考虑(比如读写比例等)

2018-12-10

作者回复

这个配置其实说的是“高配”😄

倒没有直接的关系

2018-12-10

高枕

0

早

2018-12-10