# 12 为什么我的MySQL会“抖”一下？

在前面第2篇《日志系统：一条SQL更新语句是如何执行的》中，介绍了WAL机制，InnoDB在处理更新语句时，只做了写日志的一个磁盘操作，这个日志叫（redo log）,在更新完内存写完redo log 之后，就返回给客户端，本次更新完成。

账本是数据文件，粉板是日志文件（redo log） 掌柜的记忆是内存

掌柜要找时间板账本更新一下，对应的是把内存的数据写入磁盘的操作，术语是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

**当内存的数据页跟磁盘的内容不一致的时候，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也的内容时一致的，称为“干净页”。**

示意图展示了“孔乙己赊账”的整个过程，假设原来欠账10文，这次又赊账9文

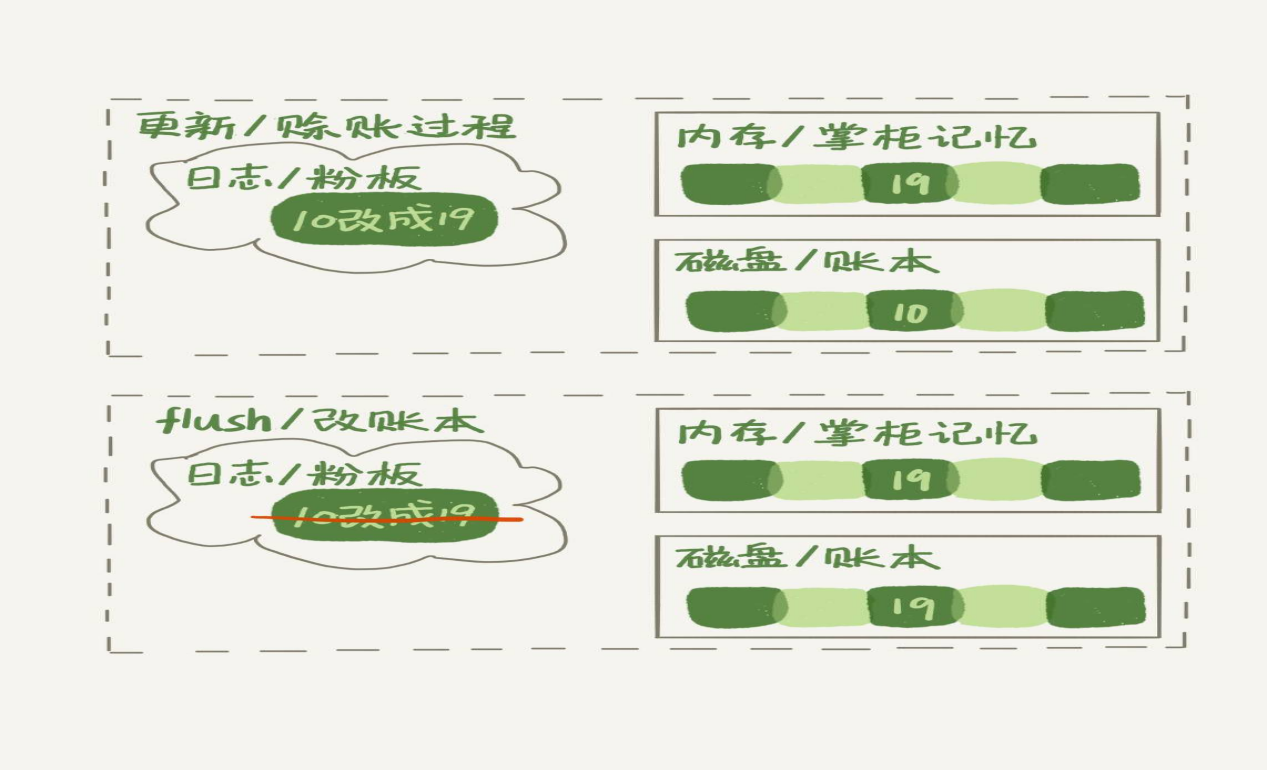
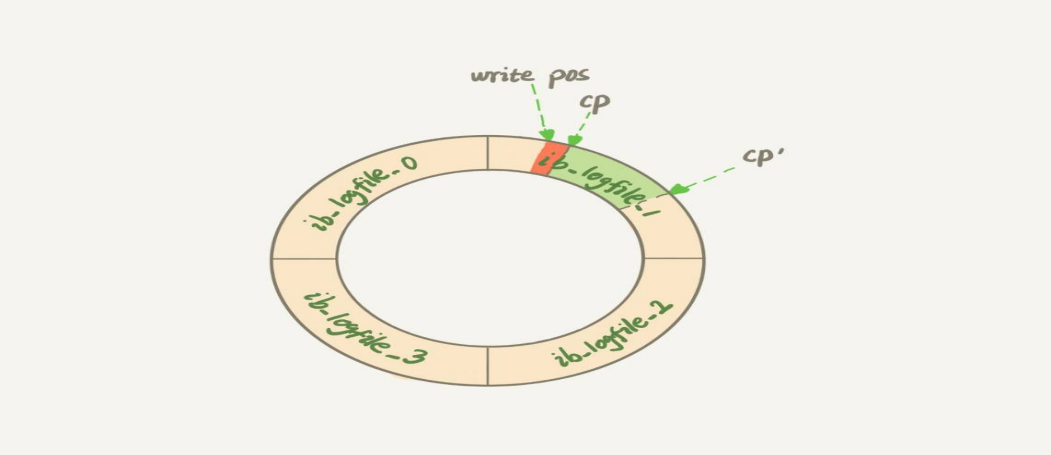


图1 “孔乙己赊账”更新和flush过程

平时执行很快的操作，其实就是写日志和内存，而MySQL偶尔会“抖”一下，的那个瞬间，可能就是在刷脏页（flush）.

**第一种场景,**InnoDB的redo log写满了，此时系统会停止所有的更新操作，吧checkpoint 往前推进，redo log 留出空间可以额继续写，redo log 示意图便于理解，checkpoint可以不是随便的往前修改一下位置就可以了，比如图2中，把checkpoint 位置从P推进到p`,就需要将这两点之间的日志（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redo log的区域



**第二种场景，**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，就要淘汰一部分数据页，空出内存给别的数据页用。如果淘汰的是“脏页”，就要先将脏页写到磁盘。

疑问？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？从性能考虑，如果刷脏一定会写磁盘，就保证了每个数据页有两种状态；

1. 内存里存在，内存里的肯定就是正确的结果，直接返回
2. 内存里没有数据，就可以肯定数据文件是正确的的结果，读入内存返回

这样的效率高

**第三种场景，**对应是MySQL认为系统“空闲”的时候，MySQL这家酒店生意好起来的时候可是很快就把粉版记满的，所有合理的安排时间，即使是“生意好”的时候，也要见缝插针的找时间，只要有机会就刷一点“脏页”

**第四种场景，**对应的是MySQL正常关闭的时候，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。

**上述四种场景对性能的影响**

其中，第三种情况是属于MySQL空闲时候，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，不会关注“性能”问题，所以要分析前两种情况。

第一种“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，整个系统就不再接受更新了，所有的更新都必须堵住，如果从监控上看，这时候更新会跌为0

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

**1** 还没有使用的

2 使用了并且是干净页

3 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使用的库来说，未使用的页很少

而当要读入的数据页没有在内存的时候，就必须到缓冲池申请一个数据页，此时只能把最久不使用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；但是如果是脏页呢？就必须将脏页刷到磁盘，变成干净页后才能复用。

刷脏也虽然是常态，但是出现一下两种情况，都是明显影响性能的

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，是不可接受的

所以 InnoDB 需要控制脏页比例的机制，来尽量避免上面的两种情况

**InnoDB 刷脏页的控制策略**

首先要正确的告诉InnoDB所在主机的IO能力，这样InnoDB才能知道需要全力刷脏也的时候，可以刷多快。

这要用到innodb\_io\_capacity这个参数， 它会告诉InnoDB磁盘能力，这个值建议设置成磁盘的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=500M -numjobs=10 -runtime=10 -group\_reporting -name=mytest

虽然定义了“全力刷脏”的行为，但平时不能一直全力刷脏？毕竟磁盘能力不只用来刷脏页，还需要服务用户请求，一起看看InnoDB怎么控制引擎按照“全力”的百分比来树脏页

**控制刷脏的速度，考虑哪些因素**

一个是脏页比例，一个是redo log 写盘速度

InnoDB根据这两个因素先单独算出来两个数字

参数innodb\_max\_dirty\_page\_pct是脏页比例上限，默认是75%，InnoDB会根据当前脏页比例（假设为M）,算出来一个范围在0-100之间的数字，计算这个数字的伪代码类似于

F1(M)

{

if M>=innodb\_max\_dirty\_pages\_pct then

return 100;

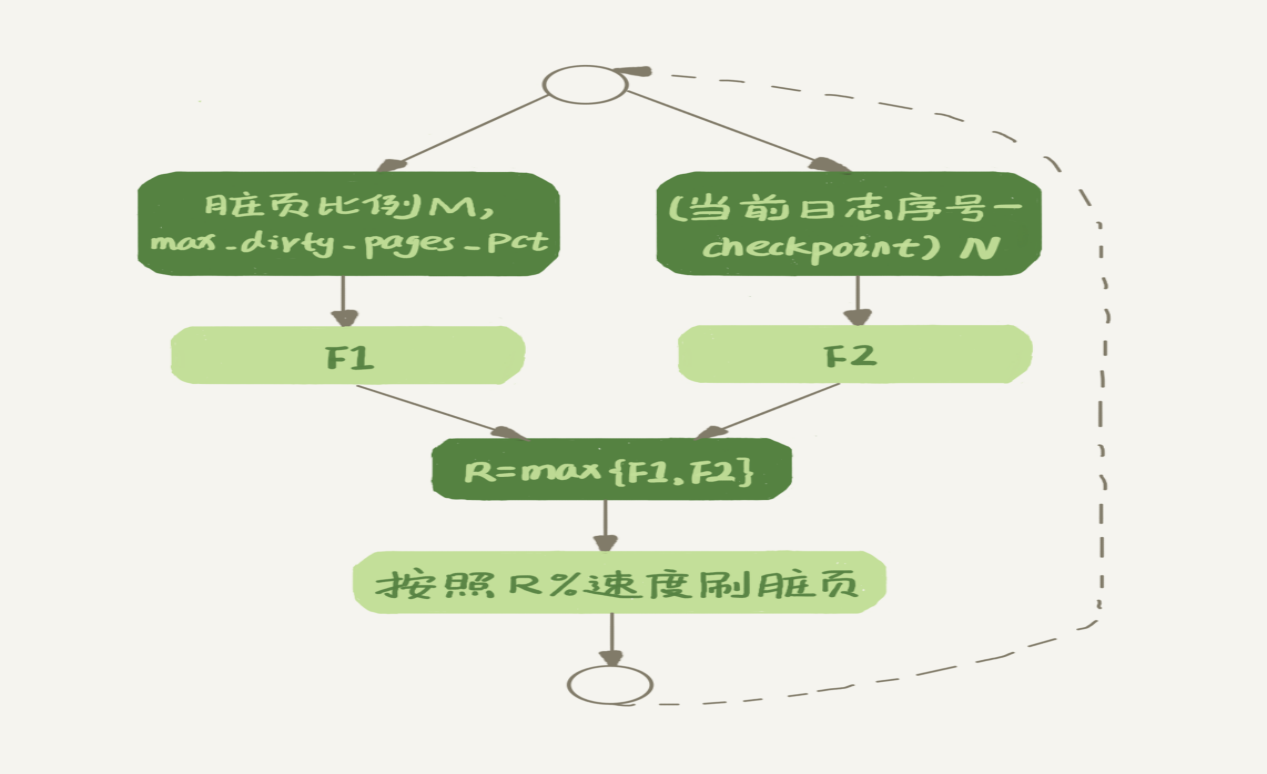
return 100\*M/innodb\_max\_dirty\_pages\_pct;

}

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，假设为N,InnoDB会根据这个N算出一个范围在0-100之间的数字，这计算公式可以即为F2(N),F2(N)算法比较复杂，只要知道N越大，算出来的值越大就可以了

**根据上述算的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，简单的流程图，F1和F2就是上面通过脏页比例和redo log写入速度算出来的值



InnoDB会在后台刷脏页，而刷脏页的过程是要将内存也写入磁盘，所以，无论是查询语句需要内存的时候可能要淘汰一个脏页，还是由于刷脏页的逻辑占用IO资源并可能影响到跟新语句，都可能造成从业务端感知到MySQL“抖”了一下的原因。

要尽量避免这种情况，就要合理的设置innodb\_io\_capacity的值，并且**平时要多关注脏页比例，不要让他经常接近75%。**

其中，脏页比例，通过innodb\_buffer\_pool\_pages\_dirty/innodb\_buffer\_pool\_pages\_total得到，参考代码

mysql> select VARIABLE\_VALUE into @a from global\_status where VARIABLE\_NAME = 'Innodb\_buffer\_pool\_pages\_dirty';

select VARIABLE\_VALUE into @b from global\_status where VARIABLE\_NAME = 'Innodb\_buffer\_pool\_pages\_total';

select @a/@b;

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，这个查询就可能要比平时慢了，而MySQL中的一个机制，可能让查询更慢，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，而这个把“邻居”托下水的逻辑还可以继续蔓延，也就是对于每个邻居数据页，如果跟他相邻的数据页也是脏页的话，也不被一起刷。

在InnoDB中，innodb\_flush\_neighbors参数是控制这个行为的，值为1时会有“连坐”机制，值为0时表示不找邻居，自己刷自己。

这个优化在机械硬盘有意义，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,意味着系统性能的大幅度提升。

而如果是SSD这类IOPS比较高的设备的话，建议把Innodb\_flush\_neighbors的值为0，IOPS往往不是瓶颈， 而值刷自己，就能更快的执行完必要的刷脏页操作，减少SQL语句响应时间。

MySQL 8.0 中，innodb\_flush\_neighbors参数的默认值是0

**思考题**

一个内存配置为128GB,innoDB\_io\_capacity设置为20000的大规格实例，正常会建议将redo log设置为4个1GB的文件

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