62 案例实战：线上数据库不确定性的性能抖动优化实践（上）

案例实战：线上数据库不确定性的性能抖动优化实践（上）

之前我们花费了很大篇幅来给大家深入和细致的讲解数据库在执行增删改这类更新语句时候的底层原理，这里涉及到了很多数据库内核级的概念，比如buffer pool、redo log buffer、lru/flush链表，等等，大家对数据库执行更新语句的原理都有了较为深入的理解。

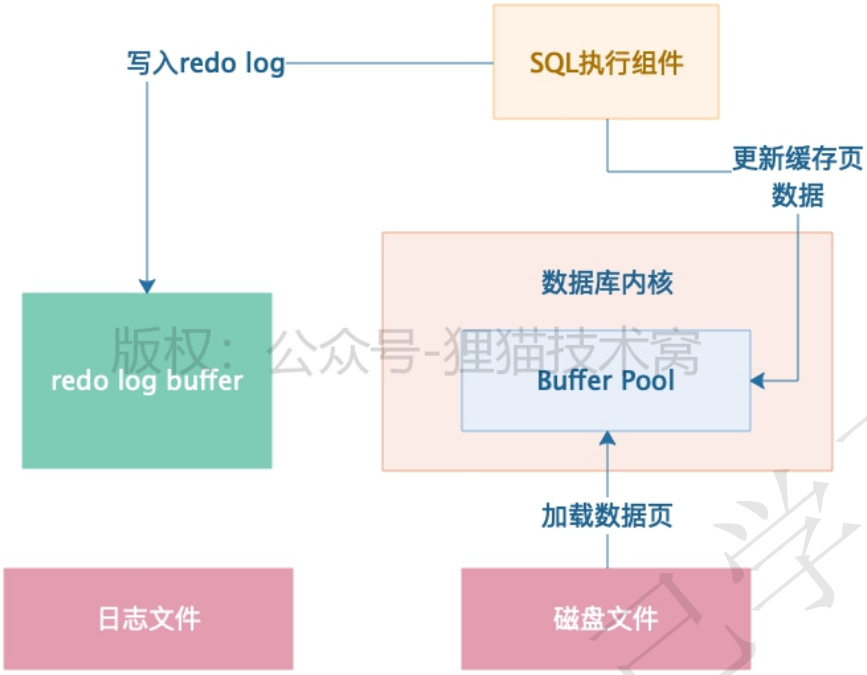
同时我们还在这个基础之上，给大家深入分析了数据库的事务的底层原理，包括事务隔离和mvcc机制的原理和概念，以及多事务并发运行时的锁机制，如何让多个事务合理的在数据库内部并发执行，同时读写共享的数据。

接着我们就要给大家讲解关于数据库更新这块的一些生产实践案例了，主要会讲解数据库更新时候的性能抖动优化、各种奇葩的锁导致性能降低的问题以及死锁问题，包括删库跑路、数据丢失等问题。

相信大家学习完接下来这部分内容之后，对自己日常工作中，线上数据库出现的一些数据更新导致的锁、数据丢失等生产故障，都能自己进行排查、定位和解决了，也就达到了我们希望的大家通过学习专栏掌握生产级优化能力的初衷。

今天我们要给大家讲解的第一个生产案例，就是线上数据库时不时莫名其妙的来一次性能抖动的问题，而且造成性能抖动的还不是之前我们讲过的数据库锂电池充放电的问题，而是另外一个新的问题，跟我们之前讲解的原理是息息相关的。

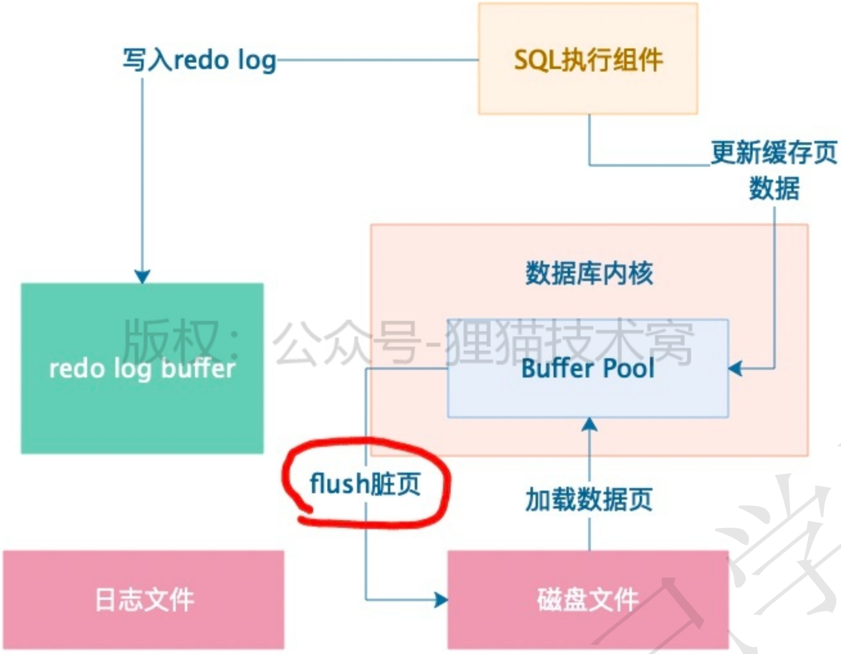
大家都知道一件事情，那就是我们平时在数据库里执行的更新语句，实际上都是从磁盘上加载数据页到数据库内存的缓存页里来，接着就直接更新内存里的缓存页，同时还更新对应的redo log写入一个buffer中，如下图所示。



那么大家都知道，既然我们更新了Buffer Pool里的缓存页，缓存页就会变成脏页，之所以说他是脏页，就是因为缓存页里的数据目前跟磁盘文件里的数据页的数据是不一样的，所以此时叫缓存页是脏页。

既然是脏页，那么就必然得有一个合适的时机要把那脏页给刷入到磁盘文件里去，之前我们其实就仔细分析过这个脏页刷入磁盘的机制，他是维护了一个lru链表来实现的，通过lru链表，他知道哪些缓存页是最近经常被使用的。

那么后续如果你要加载磁盘文件的数据页到buffer pool里去了，但是此时并没有空闲的缓存页了，此时就必须要把部分脏缓存页刷入到磁盘里去，此时就会根据lru链表找那些最近最少被访问的缓存页去刷入磁盘，如下图所示。



那么万一要是你要执行的是一个查询语句，需要查询大量的数据到缓存页里去，此时就可能导致内存里大量的脏页需要淘汰出去刷入磁盘上，才能腾出足够的内存空间来执行这条查询语句。

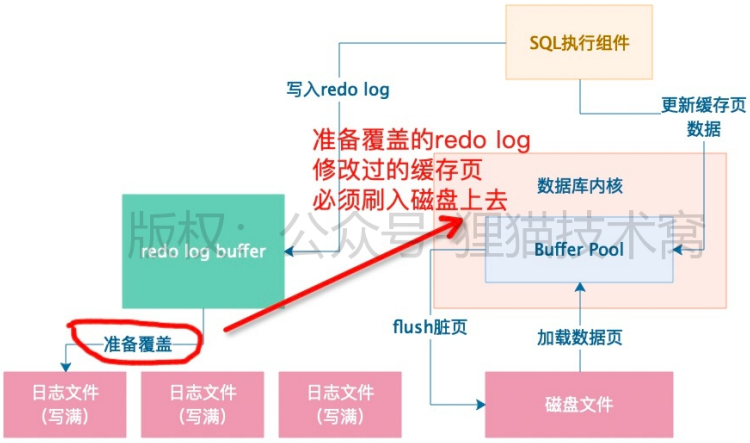
在这种情况下，可能你会发现突然莫名其妙的线上数据库执行某个查询语句就一下子性能出现抖动，平时只要几十毫秒的查询语句，这次一下子要几秒都有可能，毕竟你要等待大量脏页flush到磁盘，然后语句才能执行！

另外还有一种脏页刷磁盘的契机，之前我们并没有给大家提到，就是大家都知道redo log buffer里的redo log本身也是会随着各种条件刷入磁盘上的日志文件的，比如redo log buffer里的数据超过容量的一定比例了，或者是事务提交的时候，都会强制buffer里的redo log刷入磁盘上的日志文件。

然后我们也知道，磁盘上是有多个日志文件的，他会依次不停的写，如果所有日志文件都写满了，此时会重新回到第一个日志文件再次写入，这些日志文件是不停的循环写入的，所以其实在日志文件都被写满的情况下，也会触发一次脏页的刷新。

为什么呢？因为假设你的第一个日志文件的一些redo log对应的内存里的缓存页的数据都没被刷新到磁盘上的数据页里去，那么我问你，一旦你把第一个日志文件里的这部分redo log覆盖写了别的日志，那么此时万一你数据库崩溃，是不是有些你之前更新过的数据就彻底丢失了？

所以一旦你把所有日志文件写满了，此时重新从第一个日志文件开始写的时候，他会判断一下，如果要是你第一个日志文件里的一些redo log对应之前更新过的缓存页，迄今为止都没刷入磁盘，那么此时必然是要把 那些马上要被覆盖的redo log更新的缓存页都刷入磁盘的，如下图。



尤其是在这一种刷脏页的情况下，因为redo log所有日志文件都写满了，此时会导致数据库直接hang死，无法处理任何更新请求，因为执行任何一个更新请求都必须要写redo log，此时你需要刷新一些脏页到磁盘，然后才能继续执行更新语句，把更新语句的redo log从第一个日志文件开始覆盖写。

所以此时假设你在执行大量的更新语句，可能你突然发现线上数据库莫名其妙的很多更新语句短时间内性能都抖动了，可能很多更新语句平时就几毫秒就执行好了，这次要等待1秒才能执行完毕。

因此遇到这种情况，你必须要等待第一个日志文件里部分redo log对应的脏页都刷入磁盘了，才能继续执行更新语句，此时必然会导致更新语句的性能很差。

所以综上所述，导致线上数据库的查询和更新语句莫名其妙出现性能抖动，其实就很可能是上述两种情况导致的执行语句时大量脏缓存页刷入磁盘，你要等待他们刷完磁盘才能继续执行导致的。

下一次我们继续讲解，针对上述两种情况的数据库性能抖动，应该如何来优化数据库的参数配置，来解决上述的性能问题。

End