图文 24 我们写入数据库的一行数据，在磁盘上是怎么存储的?

详情 评论

我们写入数据库的一行数据，在磁盘上是怎么存储的?

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑
如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：**《MySQL专栏付费用户如何加群》**（购买后可见）

1、承上启下：在Buffer Pool之后，为什么要学习MySQL物理数据模型?

之前的一些文章我们已经深入的给大家分析了当你执行crud操作的时候，MySQL是如何把磁盘上的数据页加载到内存中的Buffer Pool的缓存页里去的，以及对Buffer Pool是如何进行一整套复杂的管理机制的。

相信现在每个人都对缓存页加载到Buffer Pool中，更新和读取缓存页里的数据，这个过程中的free链表、flush链表以及lru链表的维护，都有了一个深刻的理解，包括后台线程是如何定时根据flush链表以及lru链表将部分被更新的缓存页刷入磁盘的，以及缓存页都用完了以后是如何根据lru链表将一些冷数据缓存页刷入磁盘的。

按理来说，在讲解完上述内容之后，我们下一步就应该是要给大家讲解undo log和redo log以及事务机制了，但是现在还不行，实际上我们在理解了MySQL中的数据缓存机制以及内存数据更新机制，包括缓存到磁盘的数据刷新机制之后，我们还得来理解一下MySQL中的物理数据结构。

之前很多朋友可能会发现我在文章里提到了表空间、区、数据页、一个区中的连续数据页、表空间号以及数据页号，这些概念，这些概念，我们之前即使不理解，其实也不妨碍我们去理解MySQL的Buffer Pool缓存机制。

因为大家之前可能脑子里大致都有一个概念，就是数据页这个概念，起码知道每一行数据都是放在数据页里的，我们是按照数据页为单位把磁盘上的数据加载到内存的缓存页里来，也是按照页为单位，把缓存页的数据刷入磁盘上的数据页中。

但是我之前也问过大家，我们平时写SQL语句的时候脑子里都有一个表、行和字段的概念，但是为什么跑到MySQL内部，就出现了一堆表空间、数据区、数据页这些概念呢?

其实很多人都说了，表、行和字段是逻辑上的概念，而表空间、数据区和数据页其实已经落实到物理上的概念了。

实际上表空间、数据页这些东西，都对应到了MySQL在磁盘上的一些物理文件了。

所以接下来，我们要用一些文章来逐步逐步的讲解MySQL的表空间、数据区、数据页、磁盘上的物理文件这些概念。当大家看明白这些东西之后，你就会理解，当我们执行SQL语句的时候，是从MySQL机器上的哪些磁盘文件里加载数据页到缓存页里来的，数据页是如何由数据区这个概念来组织起来的，表空间这个概念是怎么回事

很多朋友之前还在评论区提问，你一个SQL语句仅仅指定了你要查询或者更新哪个表的哪些数据，那你怎么知道这些数据在哪个表空间里?在哪个数据区里?在哪些数据页里?对应是在MySQL机器上的哪些磁盘文件里呢?

当我们学习完接下来的一部分内容后，上述问题的答案都会迎刃而解。

2、之前遗留思考题解答：为什么不能直接更新磁盘上的数据?

之前我们留过一个思考题，让大家思考一下，为什么MySQL要设计这么一套复杂的数据存取机制，要基于内存、日志、磁盘上的数据文件来完成数据的读写呢？为什么对insert、update请求，不直接更新磁盘文件里的数据呢？

很多人都在评论区给出了自己的思考和回答，我觉得每个人的思考都特别的好，大家可以多去评论区里看看别人的思考以及进行交流。

这里我简单一句话总结，为什么不能直接更新磁盘上的数据，因为来一个请求就直接对磁盘文件进行随机读写，然后更新磁盘文件里的数据，虽然技术上是可以做到的，但是那必然导致执行请求的性能极差。

因为磁盘随机读写的性能是最差的，所以直接更新磁盘文件，必然导致我们的数据库完全无法抗下任何一点点稍微高并发一点的场景。

所以MySQL才设计了如此复杂的一套机制，通过内存里更新数据，然后写redo log以及事务提交，后台线程不定时刷新内存里的数据到磁盘文件里

通过这种方式保证，你每个更新请求，尽量就是更新内存，然后顺序写日志文件。

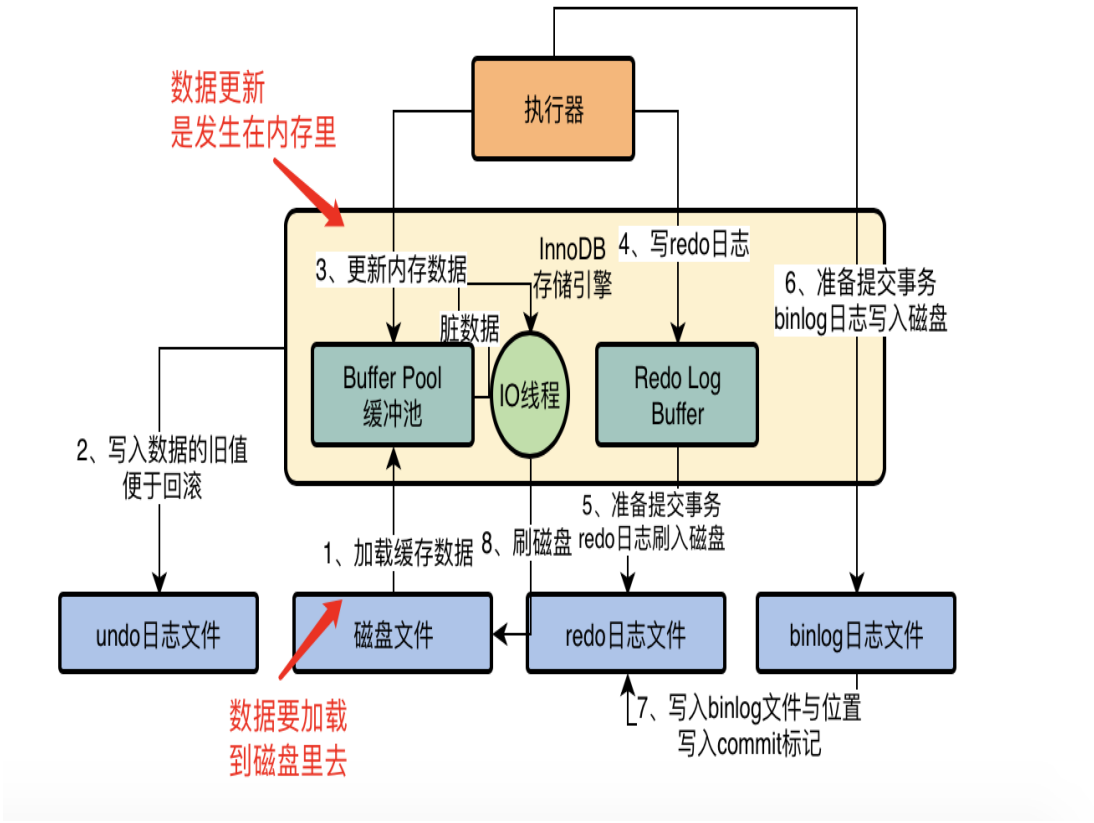
更新内存的性能是极高的，然后顺序写磁盘上的日志文件的性能也是比较高的，因为顺序写磁盘文件，他的性能要远高于随机读写磁盘文件。

也正是通过这套机制，才能让我们的MySQL数据库在较高配置的机器上，每秒可以抗下几千的读写请求。

3、复习巩固：MySQL为什么要引入数据页这个概念?

首先我们先考虑一下，刚才是不是已经给大家讲过，当我们要执行update之类的SQL语句的时候，必然涉及到对数据的更新操作？那么此时对数据是在哪里更新的?

此时并不是直接去更新磁盘文件，而是要把磁盘上的一些数据加载到内存里来，然后对内存里的数据进行更新，同时写redo log到磁盘上去，我们看下图回忆一下。

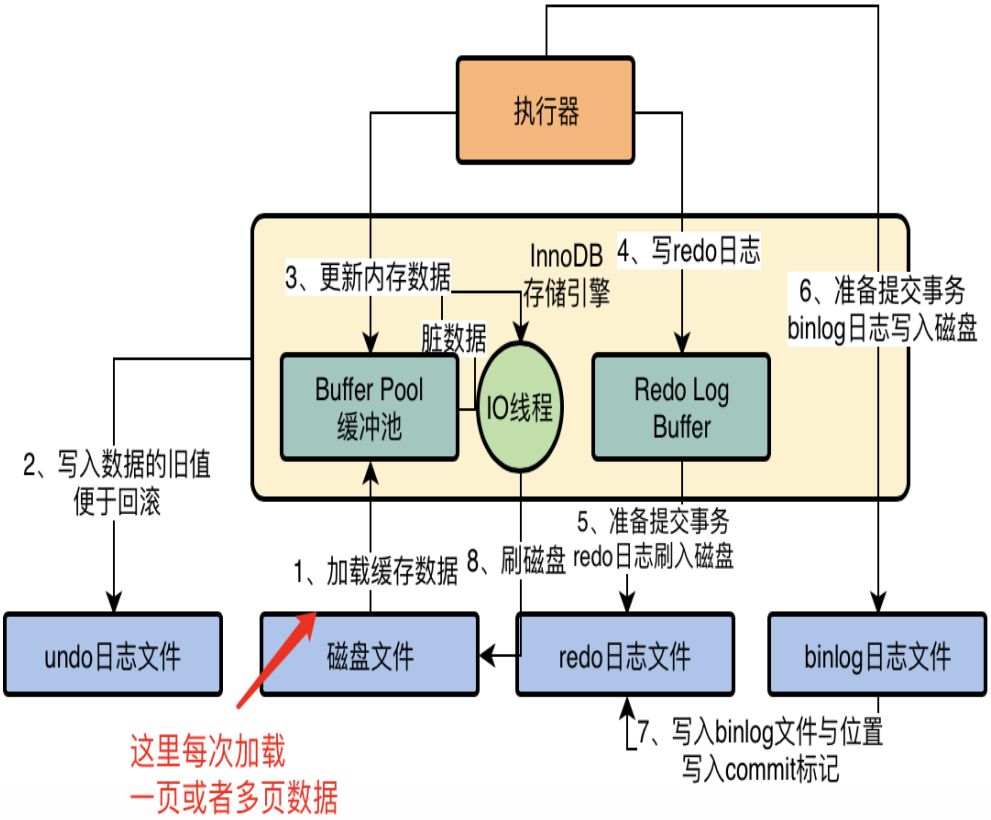


但是这里就有一个问题了，难道我们每次都是把磁盘里的一条数据加载到内存里去进行更新，然后下次要更新别的数据的时候，再从磁盘里加载另外一条数据到内存里去?

这样每次都是一条数据一条数据的加载到内存里去更新，大家觉得效率高吗?

很明显是不高的

所以innodb存储引擎在这里引入了一个**数据页**的概念，也就是把数据组织成一页一页的概念，每一页有16kb，然后每次加载磁盘的数据到内存里的时候，是至少加载一页数据进去，甚至是多页数据进去，我们看下图



假设我们有一次要更新一条id=1的数据：

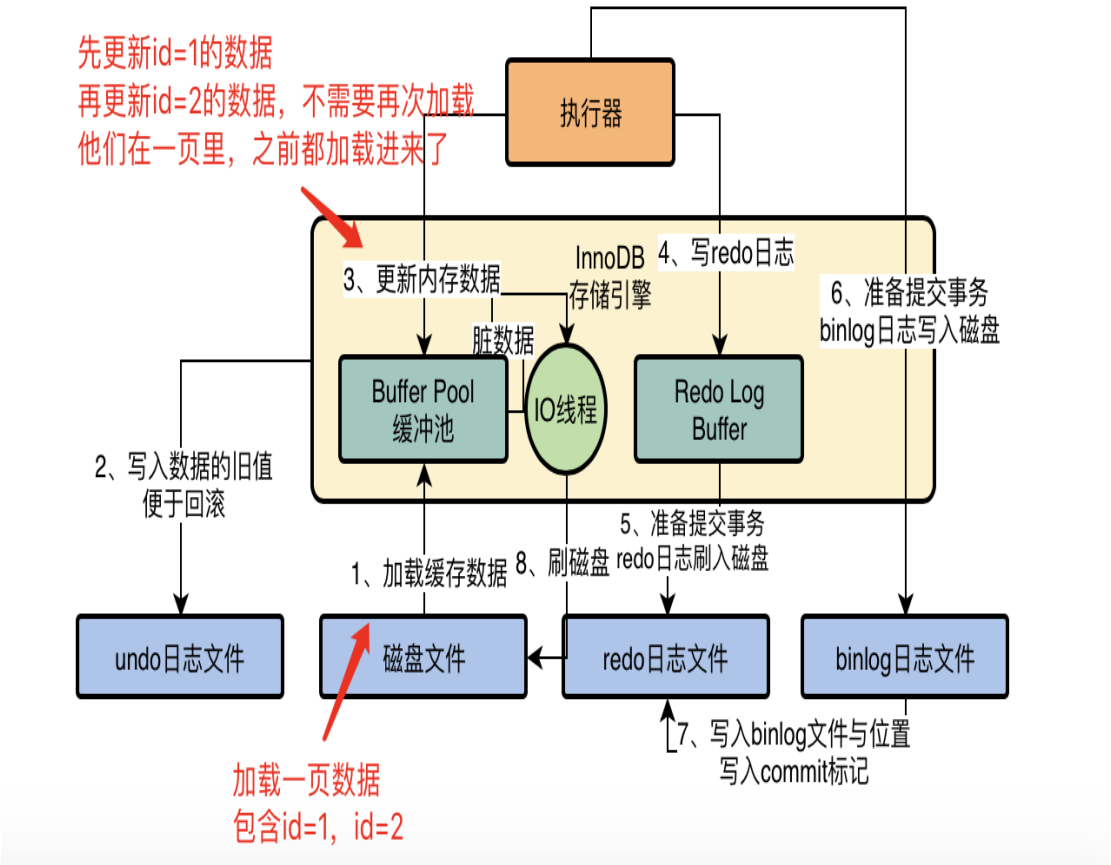
update xxx set xxx=xxx where id=1

那么此时他会把id=1这条数据所在的一页数据都加载到内存里去，这一页数据里，可能还包含了id=2，id=3等其他数据。

然后我们更新完id=1的数据之后，接着更新id=2的数据，那么此时是不是就不用再次读取磁盘里的数据了？

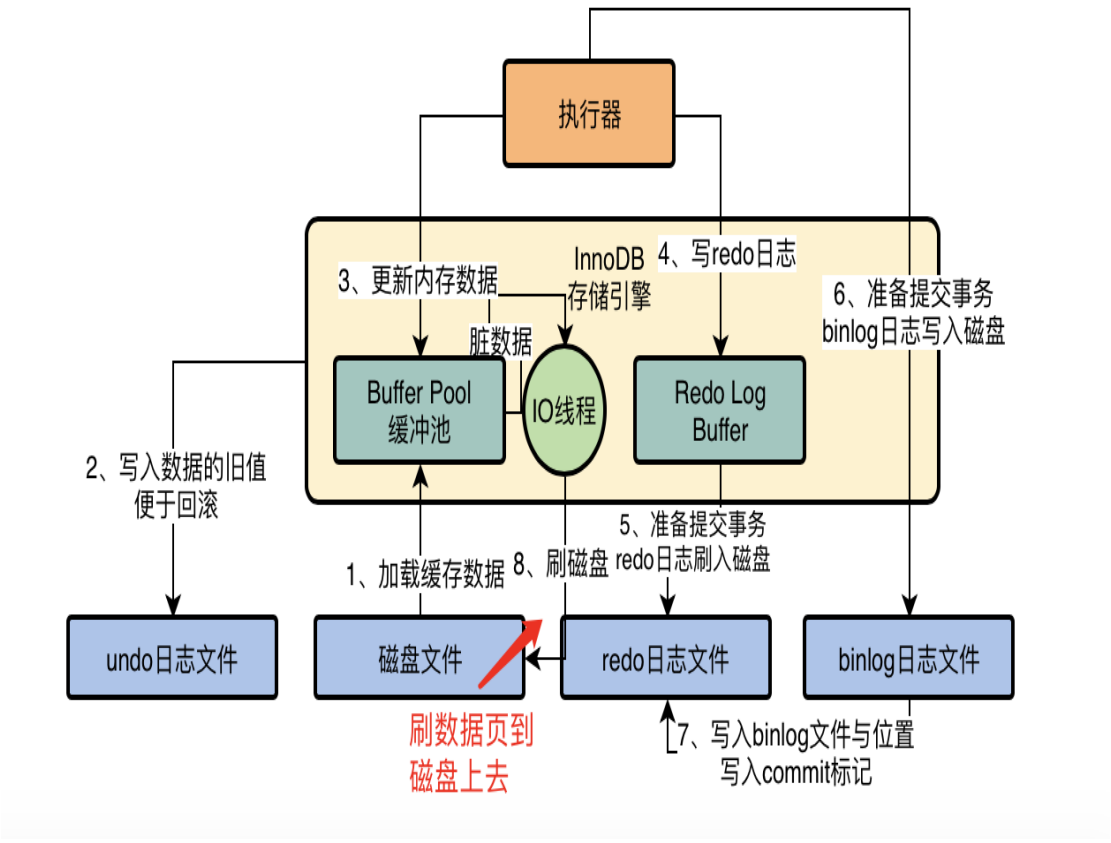
因为id=2本身就跟id=1在一页里，之前这一页数据就加载到内存里去了，你直接更新内存里的数据页中的id=2这条数据就可以了。

我们看下图，这就是数据页的意义，磁盘和内存之间的数据交换通过数据页来执行，包括内存里更新后的脏数据，刷回磁盘的时候，也是至少一个数据页刷回去。



我们再看下图，要明白的一点是，我们不是一直在内存里更新各种数据吗？当IO线程把内存里的脏数据刷到磁盘上去的时候，也是以数据页为单位来刷回去的

下图中有这个刷数据的图示：



4、初涉MySQL物理数据存储格式：一行数据在磁盘上是如何存储的?

那么接着我们可以来思考一下，对数据页中的每一行数据，他在磁盘上是怎么存储的?

其实这里涉及到一个概念，就是行格式。我们可以对一个表指定他的行存储的格式是什么样的，比如我们这里用一个COMPACT格式。

```
CREATE TABLE table_name (columns) ROW_FORMAT=COMPACT
ALTER TABLE table_name ROW_FORMAT=COMPACT
```

你可以在建表的时候，就指定一个行存储的格式，也可以后续修改行存储的格式。这里指定了一个COMPACT行存储格式，在这种格式下，每一行数据他实际存储的时候，大概格式类似下面这样：

变长字段的长度列表，null值列表，数据头，column01的值，column02的值，column0n的值.....

对于每一行数据，他其实存储的时候都会有一些头字段对这行数据进行一定的描述，然后再放上他这一行数据每一列的具体的值，这就是所谓的行格式。除了COMPACT以外，还有其他几种行存储格式，基本都大同小异。

大家可能对一行数据实际存储的时候，他里面的一些东西到底都是什么含义，感觉都很好奇，大可不必着急，我们明天的文章会继续讲解的。

5、今天学习的要点总结

今天我们主要是做了一些承上启下的复习和巩固，告诉了大家innodb存储引擎在存储数据的时候，是通过数据页的方式来组织数据的

然后我们初步的开始尝试切入了MySQL的物理数据存储格式，讲解了对于数据页中的每一行数据，其实他都有对应的行格式

接下来，我们将会深入探索这个每一行数据到底是怎么存储的。

End
