1

6 0

0

05 从零实现 KV 存储一数据读写流程 写数据流程

读数据流程

05 从零实现 KV 存储—数据读写流程



前面一节主要了解 bitcask 存储引擎总体的内存和磁盘设计,接下来我将会介绍详细的 bitcask 存储引擎数据读写 流程。

写数据流程

按照 bltcask 论文的设定,其实写数据就非常简单了,总体就是两步,先写磁盘数据文件,再更新内存索引。

首先我们将数据封装到一个结构体中,可以命名为 LogRecord ,表示追加写到数据文件中的日志记录。

LogRecord 只有三个字段,分别是 Key、Value、Type ,表示用户写入的 k/v 数据,以及标识数据类型的字段, 因为在删除数据的时候,需要一个墓碑值,所以可以用这个 Type 来表示。

```
1 // LogRecord 写入劉獻韶文件的日志记录
2 // 之所以叫日志,是因为都是追加写入的,类似日志的情式
3 type LogRecord struct {
                 Key []byte
Value []byte
Type LogRecordType
```

LogRecord 构造好之后,可以再增加一个追加写入到数据文件的方法 appendLogRecord ,这个方法在删除数据 的时候也会用到, 所以单独提取出来。

appendLogRecord 方法逻辑

这个方法的伪代码逻辑大致如下:

```
1 if 当前活跃文件 == NULL {
     initActiveFile(); // 初始化活跃文件
5 if 活跃文件 >= 文件阈值 {
6 Sync 活跃文件();
     OpenNewActiveFile(); // 打开新的活跃文件
11 Write(); // 向活跃文件中写入记录
13 return index_pos; // 返回索引位置
```

首先做一个判断,看当前活跃文件是否存在,因为当数据库处于初始状态时,可能是没有任何数据文件存在的。 如果不存在的话,就新增一个数据文件。

然后需要再做一个判断,回想一下 bltcask 的论文,当活跃文件写到阈值大小之后,需要关闭活跃文件并打开一个 新的数据文件。

所以我们需要增加一个 if 条件,当前活跃文件时候写到阈值的话,需要持久化活跃文件,这一步持久化是为了保证 将当前数据都写到磁盘中。然后将活跃文件转为旧的数据文件,并打开一个新的活跃文件。

上述判断完成后,就开始真正的往活跃文件中追加写数据了。

注意这里需要将 LogRecord 进行二进制编码,具体的编码逻辑我们可以先忽略,只是先定义一个方法即可,这个编 码的功能是将 LogRecord 结构体编码并返回一个字节数组。

然后再调用数据文件的 Write 方法写入编码后的字节数据,在 Write 之前我们先记录一下目前活跃数据文件的写偏 移 offset,表示这次写数据的起始位置,后面讲这个 offset 返回出去存储到内存索引中。

最后需要判断,如果用户指定了每次写都需要进行持久化刷盘的话,则还需要调用 Sync 方法。

对于标准系统 IO, 向文件写数据实际上会先写到内存缓冲区, 并且等待操作系统进行调度, 将其刷到磁盘当中。

如果缓冲区的内容还没来得及持久化,而此时操作系统发生了崩溃,可能会导致缓冲区的数据丢失。

所以为了避免数据丢失,我们可以手动指定 fsync 系统调用,强制将缓冲区的内容刷到磁盘,但是这样会导致效 率较低,所以我们可以提供配置选项,让用户选择是否每次写都刷盘。

磁盘数据文件写完之后,会返回一个数据位置信息,主要就是两个字段,一个是文件 id,表示写到了哪个文件当 中,二是偏移量 offset,表示写到了数据文件中的哪个位置。

我们可以把这个位置信息封装到一个结构体中,可以叫 LogRecordPos (上一讲中已经定义好了)。

```
1 // LogRecordPos 索引存放的信息,表示数据的位置
2 type LogRecordPos struct {
```

3 FAV VAILUSA:
4 Offset int64
5 }

拿到这个位置信息以后,调用更新内存索引的方法,这里需要判断更新内存索引是否成功,如果不成功的话,则直

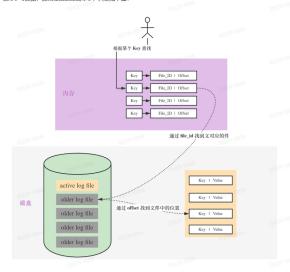
0

内存索引更新之后,写数据流程就完成了。

读数据流程

接返回错误。

相对于写数据,读流程就更加简单了,大致如下图:



首先,根据用户传入的 key 去内存索引数据结构中直找数据,如果没有找到,则说明对应的 key 根本不存在,直接返回一个 key 不存在的错误即可。

如果找到了,则取出对应的位置信息,这个位置信息就是前面存入的 LogRecordPos。

根据位置信息中的文件 Id,找到对应的数据文件。这里可以先判断这个文件是不是当前活跃文件,如果是的话,则直接用活跃文件,否则从旧的数据文件中去查找。

如果根据这个文件 id 没找到数据文件的话,则直接抛出数据文件不存在即可。

最后,根据位置索引的 offset 偏移信息,从数据文件中读取出数据,这里只需要调用数据文件的读取 LogRecord 方法,这个方法的具体逻辑可以先暂时忽略。