**目录**

**jar cvfm MyWordCount.jar manifest.data \*.class**

**你好**

本篇源码为MongoDB2.6.12，以阅读Linux环境为准。

# 概述

前面大致了解了MongoDB的存储结构，本篇概要介绍记录的插入过程。

待插入的数据最终回落到Record中，而Record由Extent管理，Extent被ExtentManager将其与DataFile关联。因此1条数据的插入涉及了多个对象的创建。（此处我们假设ns文件和db文件已经预创建好了）

1. 相关数据结构

首先介绍一下相关的数据结构。涉及的对象有Database、Collection、ExtentManager、Extent、Record等，这些数据结构的关系基本如下：



因为是采用内存映射文件，因此从ns file和data file处进入的结构既是内存数据结构也是磁盘数据结构。

从Database角度到空间分配。



从ExtentManager到Record之间的关联关系：



从ExtentManager计算定位Record方式：



从如上几个图可以看出：空间分配操作就由allocRecord方法落实，此处我们只分析SimpleRecordStoreV1类的分配动作。（CappedRecordStoreV1是循环存储结构，在此不做分析）

1. 记录插入流程

数据记录的插入可分为：插入记录请求、空间分配、数据写入三个过程。

3.1记录插入请求



请求在服务实体接收到之后，几经辗转到Collection::insertDocument方法，如下：



3.2空间分配和数据写入

空间分配和数据写入是一个方法中的流程，由于流程复杂在此将其拆开。空间分配由上面的Collection::insertDocument方法进入：

3.2.1空间分配

* 有空间，直接复用



* 空间不足，扩展空间



3.2.2数据写入

关于数据写入问题，这里主要探讨数据写在了内存什么地方，此处不考虑数据持久化。

持久化相关的数据结构：



从Collection::insertDocument方法进入：

关于为什么有\_view和view\_private？

如下是来自MongoDB的开发者的解释：

The reason that we have a private copy on write view of the memory mapped file is that we cannot risk the OS writing modified areas of the memory-mapped file back to disk before the entire write transaction (WriteUnitOfWork) is complete and written to the journal. Only after the journal is durable on disk (synced) is it safe to start writing the data to disk. We do that by copying the data back on the write view. Then periodically we remap our private view so we don't keep duplicate copies indefinitely.

See [here](https://github.com/mongodb/mongo/blob/2bd286acef2fdb035f1d45253f6e6e4c24a2dc04/src/mongo/db/storage/mmap_v1/dur.cpp#L30-L52)（https://github.com/mongodb/mongo/blob/2bd286acef2fdb035f1d45253f6e6e4c24a2dc04/src/mongo/db/storage/mmap\_v1/dur.cpp#L30-L52） for more detail.

Doing updates this way ensures that replaying the journal after a hard crash (unclean shutdown) will leave the data in a consistent state.