并发 FAQ

github 版本 – 18b9548718ec34a93ac759b60eed1547f3e37845

根据 2013-05-07 官方文档快照翻译（v2.4.3）

翻译、编辑 **陈杰（6623662005@163.com）**

2.2 版做了一些改变。

MongoDB 使用了专门的锁机制来保证多客户端在单库上读写的一致性，也避免了多个应用在同一时间修改同一块数据。锁能确保针对单条数据的所有写操作要么全部执行要么都不执行。*(译者注：这句话的意思是指写操作可能同时会伴有对 journaling 的写，或者对复制集主节点写时会伴有对 oplog 的写，这种成对出现的写通过锁来达到要么全写入要么全不写入)*

相关资料 *MongoDB2.2 并发和内部实现介绍*

MongoDB 使用的是什么类型的锁?

MongoDB使用多读单写锁[1]来支持并发的读操作，但是只允许单个写操作排它的持有这个锁。

当存在一个读锁时，多数读操作都能共享该锁。但是，当存在一个写锁时，只有一个写操作能排它的持有这个锁，不会和任何其它的读写操作共享。

锁明显“对写更偏爱”*（译者注：原文为 writer greedy，字面意思为写贪婪的）*，意思是说写比读拥有更高的优先级。当一个读操作和一个写操作同时在等待一个锁时，MongoDB 会将锁分配给写操作。

[1] 你可以将“多读单写锁”看成是“多读锁”或者“共享独占锁”。更多信息可以查看维基百科上的多读单写锁。

MongoDB 中的锁是什么粒度的?

2.2 版本中做了一些改变。

从 2.2 版开始，针对绝大多数读写操作，MongoDB 实现了数据库级的锁。一些全局的操作，比如一个涉及多个数据库的短暂操作仍然需要一个全局的锁。在 2.2 版之前每个 MongoDB 实例都只有一个全局的锁。

例如，假设有 6 个数据库，其中一个库有一个写锁，其它 5 个库依然能提供读写服务。

怎样查看mongod实例的锁状态?

下面几个方法可以查看锁的使用信息：

- *db.serverStatus()*
- *db.currentOp()*
- *mongotop*
- *mongostat*
- *MongoDB监控服务(MMS)*

再具体一点，*serverStatus输出*章节中的**locks** 部分或者是 *currentOp* 章节中的**locks** 部分，都提供了**mongod**实例中锁类型和锁竞争数的信息。

如果要终止某个操作，可以执行**db.killOp()**。

读或写操作会在某个执行时间让出锁吗?

2.0 版新增

当发生*页错误*或目标数据不在内存中时，读写操作会将锁交出。Mongodb会允许执行其它只需要访问内存中的数据就可以完成工作的操作，直到**mongod** 将所有目标数据加载到内存。

此外，影响多条数据的写操作（比如带**multi** 参数的 **update()**），在它相对长时间的执行过程中会周期性的交出锁以便让读操作能执行。同样的，长时间的读锁也会周期性的交出锁让写操作有机会完成。

2.2 版做出的改变：MongoDB2.2 极大的扩展了让出锁的用法，包括“页错误让出锁”。 在执行读操作之前，MongoDB 会跟踪内存并预判数据是否可用。如果 MongoDB 发现数据并没有在内存中，那么读操作会让出锁直接数据被全部加载到内存中。一旦数据在内存中可用，读操作会再次获取这个锁来完成工作。

哪些操作会对数据库产生锁?

2.2 版有所变化

下面的表格列出了常见的数据库操作以及这些操作的锁类型。

操作	锁类型
发起一个查询	读锁
游标 执行getmore操作	读锁
insert 数据	写锁
删除数据	写锁
更新数据	写锁
Map-reduce	读锁和写锁，除非 Map-reduce 操作被声明为非原子性的。部分 map-reduce 任务能够并发的执行。
创建索引	创建索引操作默认在前台执行，会长时间锁住数据库。
db.eval()	写锁。 **db.eval()**会阻塞其它基于JavaScript的操作。
eval	写锁。如果将**nolock**声明为， **eval**操作不会持有写锁也不会向数据库写入数据。
aggregate()	读锁

哪些管理类命令会产生锁?

某些管理类命令会排它的锁住数据库很长时间。在一些大规模集群的部署环境下，可能需要对 **mongod**实例做离线操作避免某些操作影响客户端的访问。例如，某个**mongod**实例是*复制集*的一个成员，可以将这个实例下线来做维护类操作，复制集中的其它成员继续提供服务。

下面这些管理类操作会长时间持有数据库的排它锁(写锁)：

- **db.collection.ensureIndex()**，未将**background**属性设置为**true**
- **reIndex**
- **compact**
- **db.repairDatabase()**
- **db.createCollection()**，当创建一个非常大的固定集合时
- **db.collection.validate()**
- **db.copyDatabase()**，这个操作会锁住所有的数据库。见下文 *有没有什么操作会锁住一个以上的数据库?*

下面的操作只会锁住数据库很短的时间：

- **db.collection.dropIndex()**
- **db.getLastError()**
- **db.isMaster()**
- **rs.status()** (即 **replSetGetStatus**,)
- **db.serverStatus()**
- **db.auth()**
- **db.addUser()**

有没有什么操作会锁住一个以上的数据库?

下面这些操作会锁住多个数据库：

- **db.copyDatabase()**会立即锁住整个**mongod**实例。
- *Journaling*，这是一个内部操作，它会短暂的锁住所有数据库。所有数据库共享一个journal。
- *用户认证* 会锁住**admin** 库以及该用户正在访问的库。
- 对复制集*主节点* 的写操作会锁住目标库和**local**库。**Local**库上的锁让**mongod**进程往主节点的*oplog*表中写数据。

分片上的并发

*分片*通过在多个**mongod**实例上部署分布式集合来提高并发的性能，allowing 允许分片服务器(即 **mongos** 进程)借助下游**mongod** 实例的集群优势执行任意数量的并发操作。

mongod实例之间是相互独立的，并且使用的是MongoDB*多读单写锁*。**mongod**实例上的操作不会阻塞其它实例。

复制集主节点上的并发

在*复制集*中，当往*主节点*上某个表写入时，MongoDB也会对主节点的*oplog*进行写入，这是**local** 库上一个特殊的表。因此，MongoDB会锁住目标表的库和**local库**。**mongod**进程会锁住这两个库来保证数据的一致性，让这两个库的写操作要么全执行要么全不执行。

次级节点上的并发

在*复制集*中，MongoDB不允许往*次级节点*连续写入*（译者注：这里所说的往次级节点写入是指MongoDB内部的写入，客户端是不能往次级点节写入的）*。次级节点会分批的收集oplog信息，这些批次的回写是并行执行的。当在进行写操作时，次级节点上的数据是不可读的，回写操作会按照数据在oplog中的顺序执行。

MongoDB 允许复制集次级节点上的几个写操作并行执行，分两个阶段：

1. 第一个阶段，通过一个读锁，**mongod**要确保所有与写操作有关数据都存在于内存中。在这个阶段，其它客户端可以在这个节点上做查询操作。
2. 一个使用写锁的线程池允许所有的写操作相互协调的成批写入。

MongoDB 支持什么样的 JavaScript 并发操作?

2.4 版做出的改变：MongoDB2.4 增加了JavaScript V8 引擎，允许多个JavaScript 操作同时执行。2.4 版之前，单个**mongod**实例只允许同时执行一个JavaScript 操作。