MongoDB 数据更新

李焕贞

河北师范大学软件学院

本章大纲

- MongoDB文档插入和删除
- MongoDB文档修改
- MongoDB写安全机制

MongoDB文档插入和删除

- MongoDB数据更新简介
- Insert函数介绍与使用
- Bulk函数介绍与使用
- Remove函数介绍与使用
- MMAPv1的内存分配策略

MongoDB文档插入和删除-数据更新简介

MongoDB的数据更新操作主要包括三种，分别是文档插入insert、文档删除remove以及文档修改update。

关系型数据库	MongoDB
INSERT INTO student(Sno,Sname,Sage) VALUES(101,'Joe',23)	db.student.insert({ _id:101, Sname:'Joe', Sage:23 })
Delete FROM student where Son=101	db.student.remove({Son:101})
UPDATE student SET Sage=22 WHERE Sno=101	db.student.update({Sno:101},{$set:{Sage:22} })

MongoDB文档插入和删除-insert函数

db.集合名.insert（obj，opts）

- 主要有两个参数：

1、obj要插入的文档

2、ops可选参数 可用来设置写安全级别

- 返回值为WriteResult对象

例如：WriteResult({“nInserted”:1})

MongoDB文档插入和删除-insert函数

insert函数使用说明：

- insert函数只能作用于一个集合
- 如果集合不存在，数据库服务会自动创建目标集合
- 插入文档时，如果没有指定_id字段，数据库服务会自动创建ObjectId对象作为_id的值
- 可以使用save()、update()以及findAndModify()插入文档

MongoDB文档插入和删除-Bulk函数

Bulk可以将多个数据更新操作（包括插入、修改以及删除等）放到一个待执行的列表中批量来执行。

Bulk分为两种：顺利执行的Bulk和并行的Bulk

顺序Bulk：按照**预先定义**的操作顺序（向Bulk中添加操作的顺序）来执行每一个操作

并行Bulk：以**随机的方式**并行地执行添加到执行列表中的操作

MongoDB文档插入和删除-Bulk函数

1、初始化Bulk

db.集合名.initializeUnorderedBulkOp() 并行Bulk

db.集合名.initializeOrderedBulkOp()　顺序Bulk

2、向Bulk中添加数据更新操作

Bulk.insert()、Bulk.find.update()、Bulk.find.remove()

例如：bulk.find({name:”tom”}).remove();

3、执行更新操作

调用Bulk.execute()

MongoDB文档插入和删除-Remove函数

db.集合名.remove（query， justOne）

- 主要有两个参数：

1、query查询条件，指明要删除文档的条件 相当于where语句

如果为空的，会删除所有文档 例如：db.student.remove({})

2、justOne可选参数，使用该参数（boolean），只会删除满足条件的一个文档

例如：db.student.remove({name:”tom”},true)

- 返回值为WriteResult对象

例如：WriteResult({“nRemoved”:11})

MongoDB文档插入和删除-MMAPv1的内存分配策略

- 如果一个更新操作超过了文档在磁盘上预分配的空间，MongoDB会重新在磁盘上为其分配一块更大的连续空间
- MongoDB3.0 使用“2的n次方”的方式来分配内存

例如：32,64,128,256,512…2MB,4MB…

采用这种方式的优点：

1、有利于内存的重用，降低系统碎片数量

2、减少数据移动频次，提高系统数据写效率

本章大纲

- MongoDB文档插入和删除
- MongoDB文档修改
- MongoDB写安全机制

MongoDB文档修改

- Update函数介绍与使用
- Update函数的更新操作符
- 内嵌文档的修改
- 数组元素的修改

MongoDB文档修改-Update函数

db.集合名.update(query,obj,upsert,multi)

- 主要有四个参数：

1、query　查询条件，指明要更新的文档，相当于SQL中的where语句

2、obj　　更改的内容 相当于SQL中的set语句

3、upsert 当查询条件query指明的文档不存在时，是否需要插入一条新文档{upsert:true}

4、multi　当查询条件query返回多个文档时，是否需要一次更新所有满足条件的文档{multi:true}

MongoDB文档修改-Update函数

db.集合名.update(query,obj,upsert,multi)

- 返回值为WriteResult对象

例如：

WriteResult({“nMatched”:1,“nUpserted”:0,” nModified”:1})

1、nMatched 待更新集合中，满足query条件的文档个数

2、nUpserted 当使用{upsert:true}选项时，插入文档的个数

3、nModified 实际修改的文档个数

MongoDB文档修改-更改操作符

MongoDB提供了众多原子性的更新操作符，它们拥有十分强大的功能，用于对文档的某些字段进行更新。

名称	作用	名称	作用
$inc	为文档中某个字段增加一定的值	$set	修改文档中字段的值
$mul	为文档中某个字段乘以一定的值	$unset	删除文档中的字段
$rename	为文档中的字段重命名	$min	当前更新值小于文档中的值，就将其替换
$currentDate	更新文档中的日期类型为当前时间	$max	当前更新值大于文档中的值，就将其替换

MongoDB文档修改-内嵌文档

1、修改整个内嵌文档

{$set:{field1:新内嵌文档}}

2、修改内嵌文档的某个字段

通过使用**点号**操作符来修改内嵌文档的某个字段

{$某个修改操作符:{field1.field2:value}}

MongoDB文档修改-数组元素

针对数组，MongoDB提供了大量的特定操作符，使得数组既可以作为栈、队列等有序对象使用，也可以当作集合等无序对象来使用。

名称	作用	名称	作用
$	占位符，定位已经匹配的数组元素并进行更新	$pull	用于删除指定值的元素(只能指定单个值)
$push	用于向已有的数组末尾添加一个元素	$addToSet	数组中存在相同的元素，则不会插入
$pop	从数组两端来删除元素	$pullAll	用于删除指定值的元素(指定多个值)

MongoDB文档修改-数组元素

除了更新操作符外，MongoDB为$push和$addToSet提供了一组修改器（modifiers）。通过将操作符和修改器结合使用，可以实现更多复杂的功能。

名称	作用	名称	作用
$each	可以一次插入多个数组元素 {$each: [value1,value2,...]}	$sort	对数组元素排序 { $each: [value1,value2,...] $sort:<filedName> }
$slice	取数组元素的子集 { $each: [value1,value2,...] $slice:<num> }	$position	指定元素的插入位置 { $each: [value1,value2,...] $position:<num> }

本章大纲

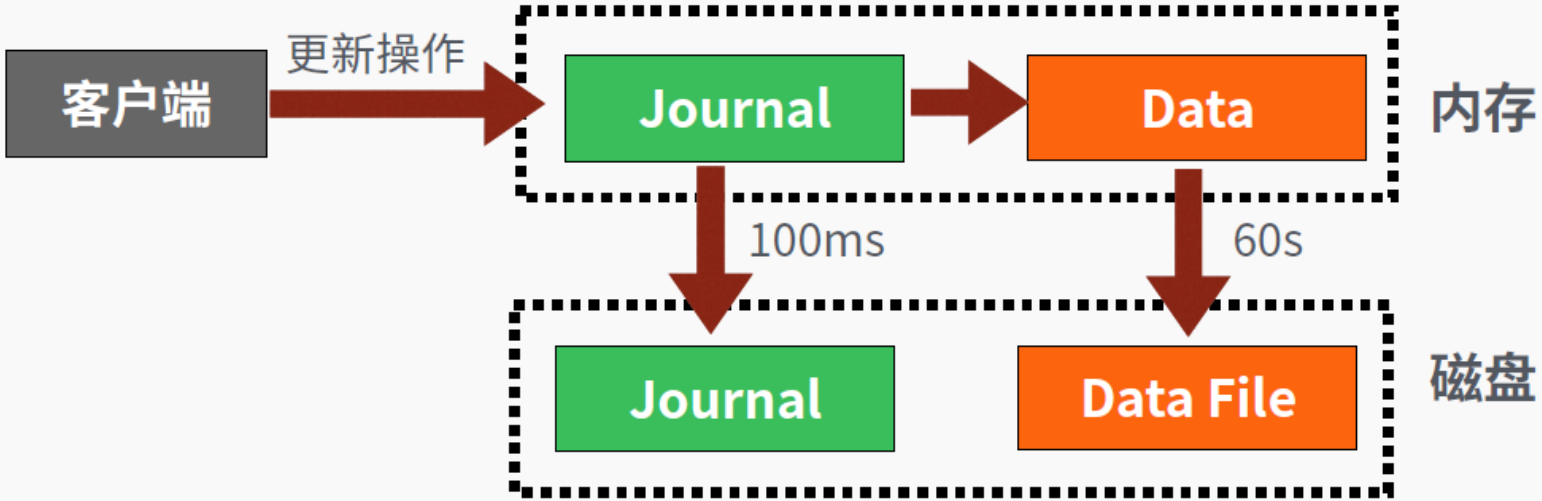
- MongoDB文档插入和删除
- MongoDB文档修改
- MongoDB写安全机制

MongoDB文档修改

- MongoDB写操作的执行过程
- MongoDB写安全级别的介绍
- MongoDB写安全级别的使用

MongoDB的写安全机制-写过程介绍

当时用insert/update/remove/save等操作更新集合中的数据时，只是修改了数据在内存中的映像，数据更新并没有同步地保存到磁盘上，而且在更新内存中的数据之前，更新操作会被记录到journal日志文件中。



MongoDB的写安全机制-写安全级别介绍

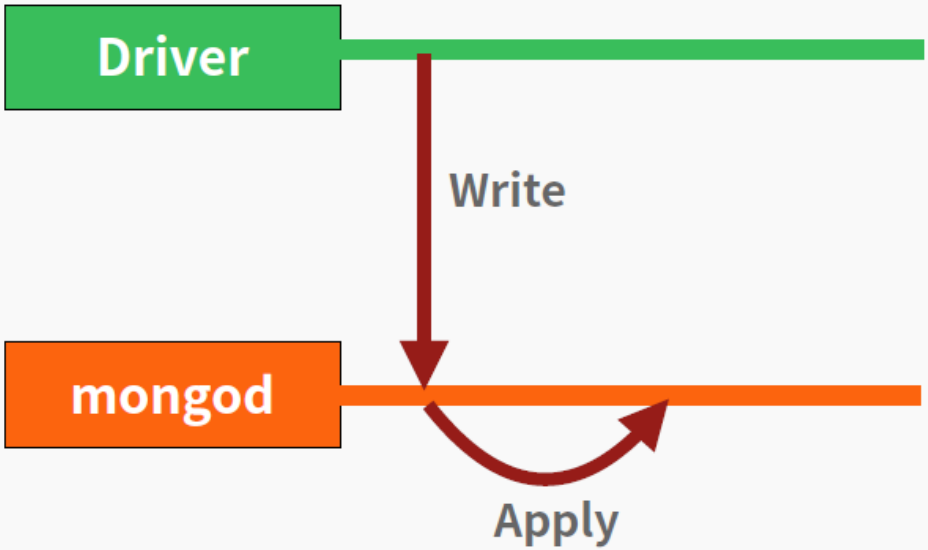
写入安全（Write Concern）是一种由客户端设置的，用于控制写入安全级别的机制，通过使用写入安全机制可以提高数据的可靠性。

MongoDB提供了四种写入级别，分别是：

- （Unacknowledged）　　非确认式写入
- （Acknowledged）　　确认式写入
- （Journaled）　　日志写入
- （Replica Acknowledged）复制集确认式写入

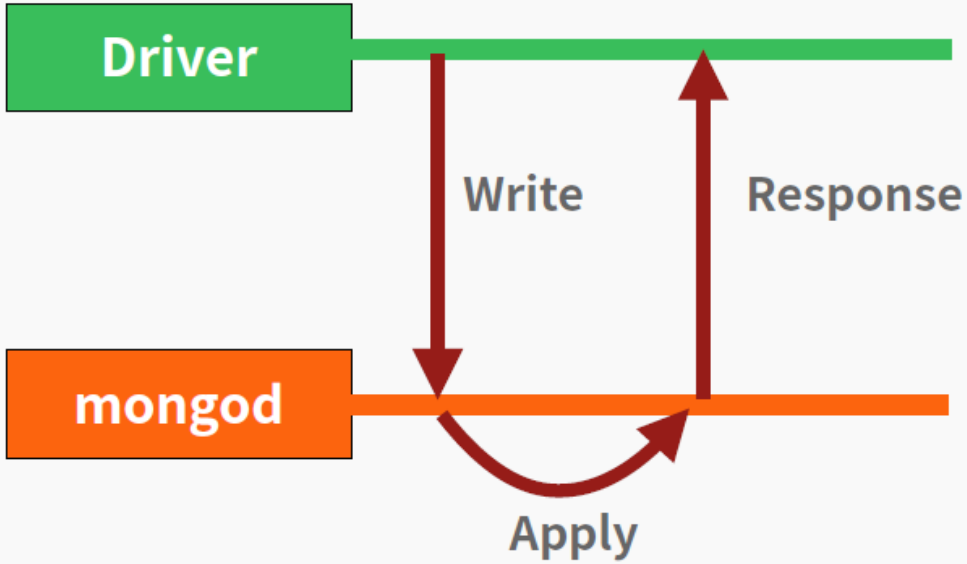
MongoDB的写安全机制-非确认式写入

非确认式写入**不会返回任何结果**，对于写操作，在没有得到服务器写入确认的情况下就立即返回，所以无法知道是否写入成功。



MongoDB的写安全机制-确认式写入

写操作必须得到MongoDB服务器的写入确认，如果写入失败，服务器会返回异常，比如：常见的DuplicateKey Error

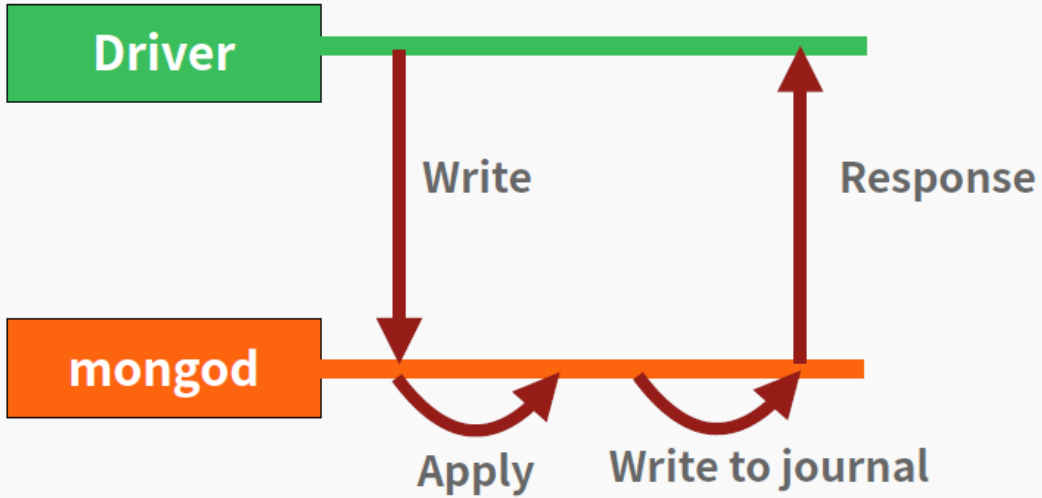


MongoDB的写安全机制-Journal日志简介

- journal日志作用相当于Oracle中的redo日志文件，用于故障恢复和持久化
- 64位机器上，MongoDB 2.0以上版本默认情况下是开启journal
- journal文件位于journal目录中，只能以追加方式添加数据，
- 数据库正常关闭时（例如：db.shutdownServer()来关闭数据库），数据库服务会清空journal目录下的所有文件
- MongoDB数据库每隔100ms或30ms向journal文件中flush一次数据

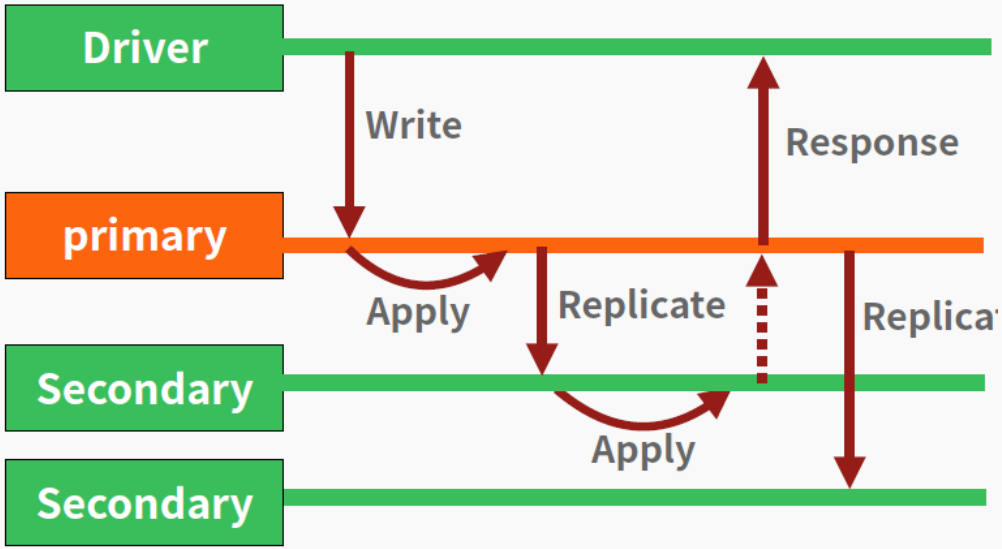
MongoDB的写安全机制-Journaled日志写入

写操作要等到操作记录存储到Journal日志后才返回结果，这种写入方式是可以容忍服务器突然宕机，有效的保障数据的可靠性。



MongoDB的写安全机制-复制集确认式写入

写操作不仅要得到主节点的写入确认，还需要得到从节点写入确认，这里可以设置写入节点的个数。



MongoDB的写安全机制-写安全级别小结

MongoDB提供的四种写安全级别，开发者可以根据自己具体的业务需要，灵活选择合适的写安全级别。

设置写安全级别，其实就是在**写操作的性能和写操作的可靠性之间取一个权衡**。使用的写安全级别越高，写操作等待时间越长，数据的可靠性也就越高。

MongoDB的写安全机制-写安全级别的使用

从MongoDB2.6开始，使用writeConcern函数来设置写安全级别。

writeConcern作为更新操作函数的一个参数，被整合到了更新操作中，所以使用起来非常方便。

例如：db.student.insert({name:”joe”},{writeConcern:{j:true}}

MongoDB的写安全机制-写安全级别的使用

writeConcern函数主要有以下参数：

- W选项 可以取0,1,2等整数值以及”majority”

  例如：{writeConcern:{w:2}}

w:0 使用非确认式安全级别

w:1 如果不使用复制集，采用确认式写入；如果使用复制集，表示主节点采用确认式写入安全级别（大于1，只能用于复制集）

w:2 在复制集中，更新操作，数据至少写到一个从节点才能返回

w:majority 在复制集中，更新操作应用到大多数的复制集成员中

MongoDB的写安全机制-写安全级别的使用

writeConcern函数主要有以下参数：

- j选项 将j选项设置为true来使用Journaled日志安全级别

  例如：{writeConcern:{j:true}}

- wtimeout选项　用于设置超时，单位为毫秒。如果在wtimeout指定的时间内写操作未能完成，将会返回一个错误。

  例如：writeConcern:{wtimeout:5000}})