**目录**

[1. 前言 1](#_Toc456010098)

[2. 状态监控 1](#_Toc456010099)

[2.1serverStatus() 1](#_Toc456010100)

[2.2mongostat 9](#_Toc456010101)

[2.3关于操作（查看、杀掉） 9](#_Toc456010102)

[3. 安全认证 10](#_Toc456010103)

[4. 备份和恢复 12](#_Toc456010104)

[4.1直接copy 12](#_Toc456010105)

[4.2mongodump和mongorestore 12](#_Toc456010106)

[4.3主从复制 12](#_Toc456010107)

[5. 参考资料 12](#_Toc456010108)

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

# 前言

一个MongoDB集群建立后，对集群状态的监控、维护、管理也是非常重要的，本篇粗略介绍MongoDB管理常用的方法和命令工具。

# 管理维护

## 2.1serverStatus()

查看当前全局锁、索引、用户操作等行为信息：

**> db.serverStatus();**

**{**

**"host" : "LIULEI26",**

**"version" : "2.6.11",**

**"process" : "mongod",**

**"pid" : NumberLong(11088),**

**"uptime" : 363,**

**"uptimeMillis" : NumberLong(362887),**

**"uptimeEstimate" : 361,**

**"localTime" : ISODate("2016-06-20T00:28:09.108Z"),**

**"asserts" : {**

**"regular" : 0,**

**"warning" : 0,**

**"msg" : 0,**

**"user" : 0,**

**"rollovers" : 0**

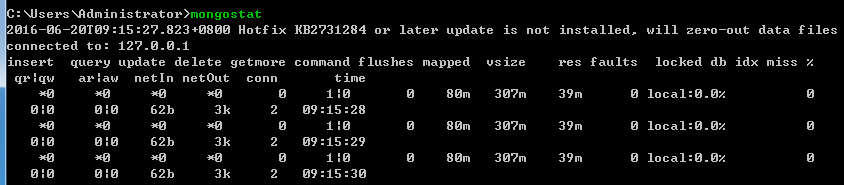
**},**

各个选项的含义如下：

|  |  |  |
| --- | --- | --- |
| Field | Example Value | Explanation |
| Host | te.rzw.com:27018 | 运行实例所监听的IP与端口 |
| version | 1.8.3 | 当前实例所使用的版本 |
| Process | mongod | 当前实例是mongod，还是mongos |
| Uptime | 231905 | 实例连续正常运行时间,单位秒 |
| uptimeEstimate | 223083 | 基于mongodb内部粗粒定时器的连续正常运行时间 |
| localTime | ISODate("2011-10-13T01:47:38.773Z") | 本机时间，以UTC时间为计算标准。 |
| globalLock.totalTime | 231905155987 | 自实例启动全局锁创建以来到现在多长时间，单位微秒. |
| globalLock.lockTime | 9053050 | 自全局锁创建以来锁定总时间，单位微秒 |
| globalLock.ratio | 0.0000390377262 | 锁定的时间所占的比例(lockTime/ totalTime) |
| globalLock.currentQueue.total | 0 | 当前等待全局锁的数量 |
| globalLock.currentQueue.readers | 0 | 当前等待读锁的数量 |
| globalLock.currentQueue.writers | 0 | 当前等待写锁的数量 |
| globalLock.activeClients.total | 1 | 连接到当前实例处于活动状态的客户端数量。 |
| globalLock.activeClients.readers | 1 | 处于活动状态的客户端中有多少是在执行read操作 |
| globalLock.activeClients.writers | 0 | 处于活动状态的客户端中有多少是在执行write操作 |
| Mem.resident | 6 | 到现在总共使用的物理内存,单位是MB |
| Mem.virtual | 17307 | 当前Mongodb实例使用的虚拟内存大小，单位MB，一般情况下比mem.map的值要超大一点，如果大很多很有可能发生内存泄露，如果使用journal,大约是2倍的map值 |
| Mem.maped | 8556 | Mongodb使所有数据都映射到内存中，所以这个值可以看似整个数据量的值。 |
| Mem.bits | 64 | 机器位数,32位或64位 |
| Mem.supported | true | 本机是否支持内存扩展 |
| Connections.current | 2 | 当前连接到本机处于活动状态的连接数 |
| Connections.available | 19998 | 剩余多少可供连接 |
| extra\_info.heap\_usage\_bytes | 521152 | 当前实例堆大小，单位bytes |
| extra\_info.page\_faults | 569 | 加载磁盘内容时发生页错误的次数 |
| indexCounters.Btree.accesses | 1 | 访问索引次数 |
| indexCounters.Btree.hits | 1 | 访问索引时，索引在内存中被命中的次数 |
| indexCounters.Btree.misses | 0 | 访问索引时，索引不是在内存中被命中的次数 |
| indexCounters.Btree.resets | 0 | 索引计数器被重置的次数 |
| indexCounters.Btree.missRatio | 0 | 索引非命中率 |
| backgroundFlushing.flushes | 3864 | 实例刷新数据到磁盘的数次 |
| backgroundFlushing.total\_ms | 15473 | 刷新到磁盘总共花费的时间,单位毫秒 |
| backgroundFlushing.average\_ms | 4.004399585921325 | 平均每次刷新执行时间 |
| backgroundFlushing.last\_ms | 3 | 最后一次刷新执行时间 |
| backgroundFlushing.last\_finished | ISODate("2011-10-13T01:46:51.184Z") | 最后一次刷新完成的时间点 |
| Cursors.totalOpen | 1 | 当前游标数量 |
| Cursors.timeOut | 2 | 从实例启动到现在游标超时的总数量 |
| Network.bytesIn | 4680778 | 发送到实例的字节大小 |
| Network.bytesOut | 4759827 | 发送出去的字节大小 |
| Repl.setName | myset | replSet结构定义的名称 |
| Repl.isMaster | True | 当前实例是否是replSet结构中的Master节点 |
| Repl.secondary | False | 当前实例是否是replSet结构中的secondary节点 |
| Repl.hosts | [ “te.rzw.com:27018”,”te.rzw.com:27019”] | replSet结构中每个节点情况 |
| Opcounters.insert | 4 | 自实例启动以来，执行insert次数 |
| Opcounters.query | 181 | 自实例启动以来，执行query次数 |
| Opcounters.update | 92084 | 自实例启动以来，执行update次数 |
| Opcounters.delete | 0 | 自实例启动以来，执行delete次数 |
| Opcounters.getmore | 92816 | 自实例启动以来，在游标执行getMore次数 |
| Opcounters.command | 316 | 自实例启动以来，执行其他操作的次数 |
| Asserts.regular | 0 | 自实例启动以来,断言正常的次数 |
| Asserts.warn | 0 | 自实例启动以来,断言警告的次数 |
| Asserts.msg | 0 | 自实例启动以来，断言内部错误的次数 |
| Asserts.user | 134 | 自实例启动以来,因用户使用造成的错误而被断言次数 |
| Asserts.rollovers | 0 | 断言被翻转的次数 |
| writeBacksQueued | False | 在mongos中，操作被重做的次数 |
| Dur.commits | 1 | 在最近间隔，向journal文件提交的次数 |
| Dur.journaledMB | 0.008192 | 在最近间隔,journalr文件写入数据量，单位MB |
| Dur.writeToDataFilesMB | 0 | 在最近间隔,从journal文件向数据文件写入的数据量，单位MB |
| Dur.commitsInWriteLock | 0 | 在最近间隔中，处于写锁状态下向journal提交的次数.这种提交方式是不被推荐的 |
| Dur.earlyCommits | 0 | 在预定时间之前请求提交的次数 |
| Dur.timeMs.dt | 3045 | Journal状态收集的频率,单位毫秒 |
| Dur.timeMs.prepLogBuffer | 0 | 向journal写数据前的准备所花费时间, 单位毫秒 |
| Dur.timeMs.writeToJournal | 2 | 向journal写数据所花费时间，单位毫秒 |
| Dur.timeMs.writeToDataFiles | 0 | 在写入journal文件后向数据文件写数据花费时间，单位毫秒 |
| Dur.timeMs.remapPrivateView | 0 | 重新在内存中映射数据花费的时间 |
| Ok | 1 | serverStatus操作是否被成功执行 |

## 2.2mongostat

serverStatus属于静态统计，mongostat可以动态的查看数据的变化，如下示例：



显示每秒，mongod的各项操作的统计：插入、查询、更新、删除、网络I/O等指标。

## 2.3Web

## 关于操作（查看、杀掉）

db.currentOp();可以当前库被操作的情况，如下示例：

**use vdc\_info;**

**db.currentOp(); # 查看当前操作**

**db.killOp(opid); # 强行关闭某个操作**

显示的列的含义如下：

{

"opid" : "mongourl-14:-1418896652", # 操作id

"active" : true, # 操作状态

"secs\_running" : 0, # 运行了多少秒

"op" : "getmore", # 操作属性

"ns" : "local.oplog.rs", # namespace的缩写而已

"query" : { # 查询操作的详细状况

},

"client\_s" : "10.58.127.55:20822", # 客户端信息

"desc" : "conn61", # 连接状态

"threadId" : "0x7ecbef780700",

"connectionId" : 61, # 连接id

"waitingForLock" : false, # 是否在等待锁的状态

"numYields" : 0, # 当前操作在读数据时，为使其它操作操作数据而放弃的次数，官方解释如下：

# numYields is a counter that reports the number of times the operation has yielded to allow other operations to complete.

"lockStats" : { # 锁的状态

"timeLockedMicros" : {

"r" : NumberLong(18),

"w" : NumberLong(0)

},

"timeAcquiringMicros" : {

}

}

}

更多示例如下：

|  |  |
| --- | --- |
| 命令 | 含义 |
| db.currentOp().inprog.forEach(function(item){if(item.secs\_running == 0) print(item.opid);}); | 查询并打印查询时间等于0秒的操作。 |
| db.currentOp().inprog.forEach(function(item){if(item.op!="query"){print(item.opid);}}) | 查看所有非查询的任务。 |
| db.currentOp().inprog.forEach(function(item){if(item.query.aggregate=="vdc\_info"){db.killOp(item.opid);}}) | 批量查询并杀死聚合为vdc\_info的操作。 |

## 2.4远程执行MongoDB命令

语法是：mongo …… --eval <javascript>

db.currentOp().inprog.forEach(function(item){if(item.query.virus\_name=="161e771b91618d603b9629aeb7260041:VeriSign Class 3 Code Signing 2010 CA:Beijing Rising Information Technology Corporation Limited"){db.killOp(item.opid);}})

161e771b91618d603b9629aeb7260041:VeriSign Class 3 Code Signing 2010 CA:Beijing Rising Information Technology Corporation Limited

示例如下：查看某个数据库的数量

待定！！！

# 状态监控

# 安全认证

我们可以为特定的数据库添加用户，来控制访问权限。普通数据库的用户只能针对本数据库进行操作，而admin数据库中的用户可以进行全库操作。

**db.addUser(username, password, onlyread);**

添加用户之后，我们如果需要在重启数据库的时候加上-**-auth选项**，来启用认证功能。如果不添加该选项，依然可以无密访问。

**mongod --dbpath="G:\MongoDB 2.6 Standard-mongod-1\data" --auth**

！注意：

1. admin数据库下添加的用户具有管理员权限，可以操作任何数据库，其他用户添加的数据库只能操作本数据库；
2. 在没有任何用户的情况下，以auth方式启动不会影响操作；
3. 如果你添加的第1个用户不是admin数据库的，然后以auth方式启动开启认证，那么后面是没有办法对其他数据库进行其他操作的。因为认证已经开启，而admin没有用户名和密码认证，这是MongoDB的逻辑。所以必须在非auth模式下添加的第1个用户是admin下面的，这样后面才能进行其他操作。

删除用户需要在admin库下面操纵，如下：

**use admin;**

**db.system.users.remove({user:”aaa”});**

我们也可以在进入MongoDB之后，再用db.auth(username, password)进行登录，比如：

**[bae@\*\*\* bin]$ ./mongo -u mongodb localhost:7222/admin**

**……**

**> show dbs;**

**Tue Jun 21 11:04:58 uncaught exception: listDatabases failed:{ "errmsg" : "need to login", "ok" : 0 }**

**> db.auth("mongodb", "zhangsan")**

**1**

**configsvr> show dbs;**

**admin 0.53515625GB**

**config 1.472412109375GB**

**local (empty)**

**configsvr>**

# 备份和恢复

## 4.1直接copy

暴力最直接，需要关掉服务器。

## 4.2mongodump和mongorestore

参见“02 MongoDB数据导入导出备份恢复篇.docx”。

## 4.3主从复制

参见“02 MongoDB数据导入导出备份恢复篇.docx”。

注意：4.1和4.2不能保证数据的实时性，有些数据可能在内存中。mongodb提供了fsync+lock的方式将数据强制写入到磁盘中。这个机制的工总方式如此：首先将缓冲区的数据写入到磁盘，然后给数据库一个写入锁，其他实例的写入曹组全部被阻塞，知道fsync+lock释放锁为止。

加锁：db.runCommand({“fsync”:1, “lock”:1});

释放锁：db.$cmd.unlock.findOne();

# 参考资料

1. http://www.cnblogs.com/huangxincheng/archive/2012/03/08/2384571.html
2. http://www.cnblogs.com/xuegang/archive/2011/10/13/2210339.html
3. https://docs.mongodb.com/v3.2/reference/method/db.currentOp/