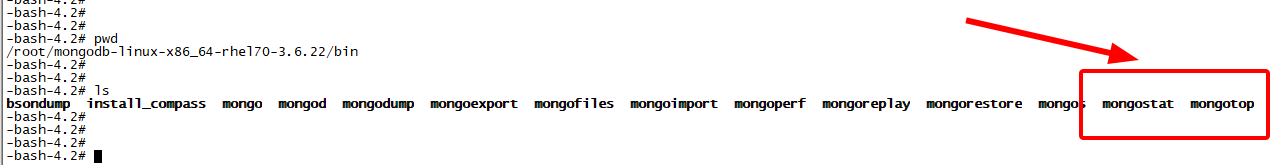
# 集群抖动异常分析方法二：mongodb官方自带性能分析工具

mongodb提供有完善的gotool工具，这些开源工具作用主要有：数据导出及恢复(mongodump、mongorestore、mongoexport、mongoimport)工具、客户端shell链接工具(mongo)、IO测试工具(mongoperf)、流量qps/时延等监控统计工具(mongostat、mongotop)。

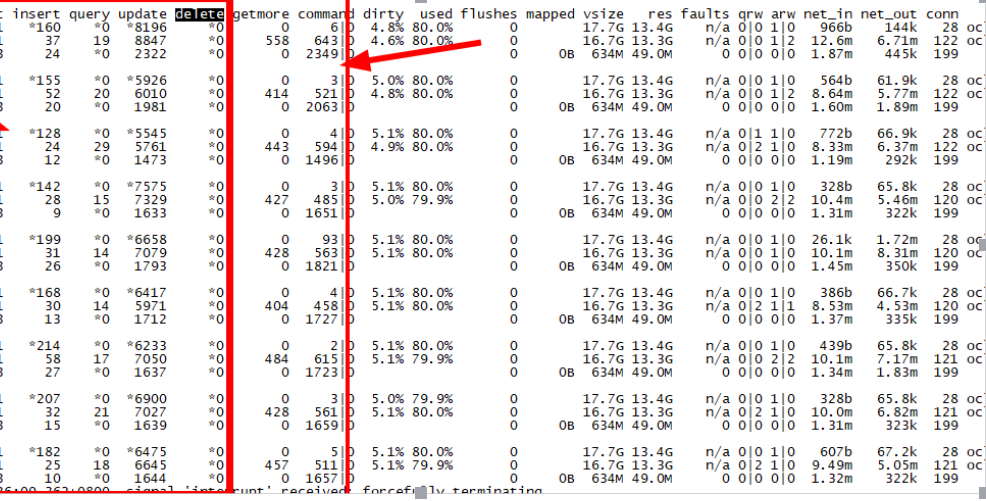
mongodb二进制安装包中，默认包含有如下二进制可执行文件：



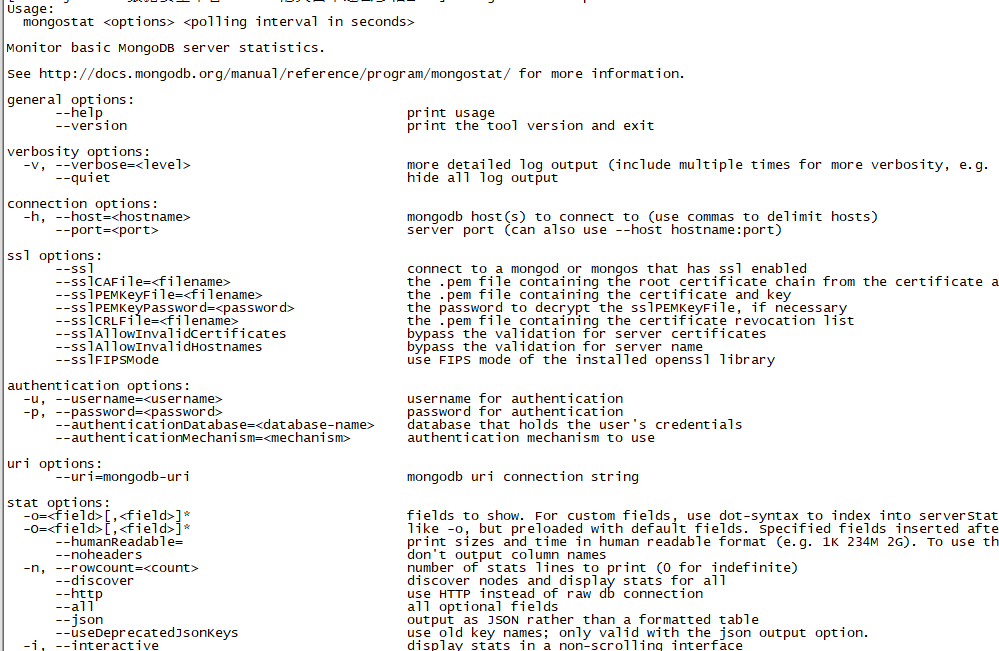
这些二进制文件中包括了所有和mongodb集群相关的实例、监控工具、数据备份恢复工具等，本文重点分析和mongodb集群抖动异常问题相关的几个工具，主要包括：mongostat、mongotop。

* **mongostat监控工具实时分析集群潜在问题**

mongodb提供了mongostat工具来监控当前集群的各种操作统计。Mongostat监控统计如下图所示：



其中，insert、delete、update、query、getMore、command这几项统计比较好理解，分别对应增、删、改、查，getMore及command六项QPS统计。mongostat help参数功能详细说明如下：

 该命令help帮助信息中各个参数选型功能说明如下：

|  |  |
| --- | --- |
| 参数项 | 功能说明 |
| general options: | 获取版本信息、help帮助信息 |
| verbosity options: | 是否打印日志信息，-v表示打印日志，v个数越多日志打印越多  quiet默认不答应日志 |
| connection options: | 链接的mongo实例ip:port地址 |
| ssl options: | SSL认证相关配置 |
| authentication options: | 鉴权认证的用户名和密码 |
| uri options: | uri链接认证方式，类似mongodb://username1:password1@ip:port |
| stat options: | 统计选项设置：  --discover：如果链接的是复制集节点，则输出整个复制集所有节点监控信息；如果 链接的是代理mongos节点，则输出整个分片集群节点监控信息。  -n：一共输出多少行即停止监控输出，默认没限制  --json：指定输出个数为json格式  -i: 直接同一个屏幕显示统计信息，屏幕刷新周期就是-i指定的时间  --humanReadable：是否进行字节到M或者K等的转换，默认true |

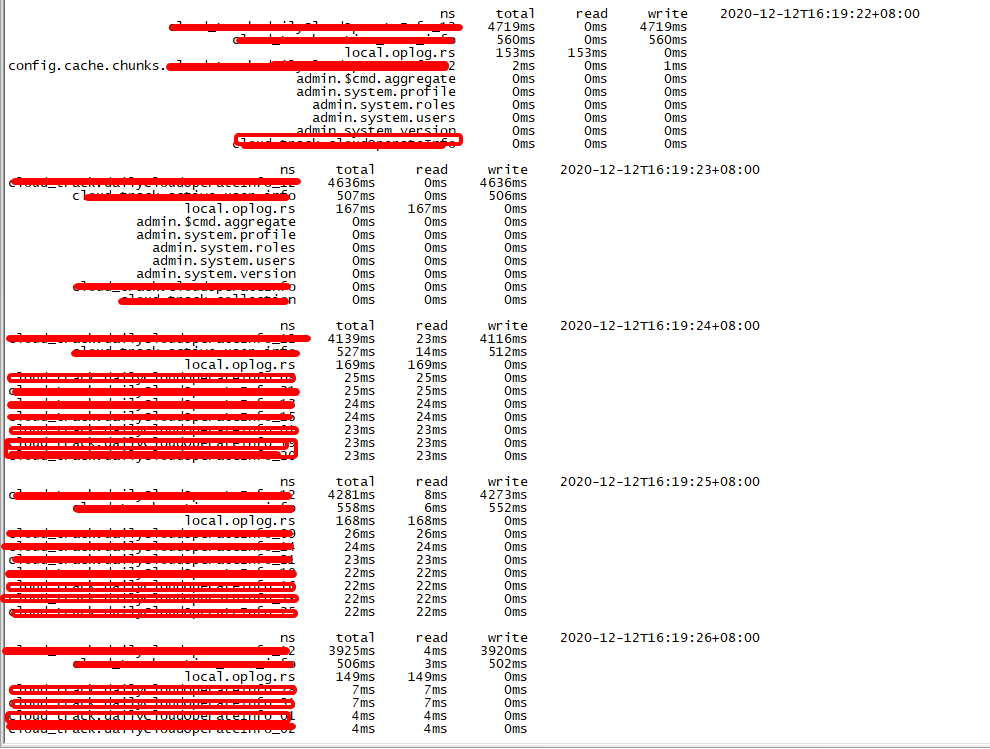
一般通过如下命令来链接mongos代理地址，从而可以获取相关节点监控信息，使用方法如下:

mongostat -h ip:port -u用户名 -p密码 --authenticationDatabase=admin --discover

mongostat工具带上--discover，可以把所有分片节点信息一起打印出来，直观查看整个集群所有节点实例级监控信息。mongostat统计信息中最核心的几个影响性能的统计项：

* **Dirty(重要，影响性能的核心监控指标)：**存储引擎脏数据比例，默认该值为5%的时候，wiredtiger存储引擎自带的evict现成开始选择脏数据page淘汰到磁盘；如果该值达到20%，客户端请求对应mongodb处理现成将会选择脏数据page淘汰到磁盘，等page淘汰腾出内存空间后，才会处理客户端请求的DB访问，所以如果阀值达到20%客户端访问将会变慢。
* **used(重要，影响性能的核心监控指标)：**存储引擎cacheSize配置占用百分比，如果配置cacheSize=10G，存储引擎实际使用了7G，则used赞比为70%。当该统计值达到80%，evict线程将会触发选择涨数据淘汰，如果这个占比提高到95%，用户请求线程将会触发淘汰，客户端请求将会变慢。
* **qrw arw(重要，影响性能的核心监控指标)：**等待队列数，如果该值越大，说明会引起客户端请求排队处理。一般该值会再dirty占比超过20%，used占比过高超过95%，或者磁盘IO慢会出现。
* **vsize res(重要，影响性能的核心监控指标)：**虚拟内存和物理内存真实占用，如果vsize过高，远远超过res，或者res过高，远远超过cachesize配置，则说明内存碎片，pageheap等问题，这时候可以通过加速tcmalloc内存释放速率来解决问题。
* **mongotop实时表级流量监控**

mongotop实现对所有表的读写时延消耗统计，并按照总耗时排序直观输出，对应统计打印信息如下图所示：



mongotop监控输出项各字段说明如下：

* ns: 表名
* read：1秒钟内客户端对该表读操作消耗的总时间
* write：1秒钟内客户端对该表写操作消耗的总时间
* total：1秒钟内客户端对该表读写消耗的总时间

mongotop工具help参数信息和mongostat中的相关字段内容作用几乎一样，请参考上面的mongotop --help参数详解部分。

通过mongotop监控，我们就可以实时的获取所有表的读写时延统计，这样就可以瞬间快速定位那个表的流量最高，是读流量还是写流量。有了该监控，我们就可以确定可能是那个表流量突发异常引起的问题。

**注意事项：该工具只能在mongod实例运行，只能监控单个节点流量。**

# 慢日志分析

mongodb默认会把慢日志记录到慢日志表和日志文件中，日志表对应库下面的表system.profile，对应日志文件也就是配置文件中systemLog.path指定的文件信息。慢日志阀值默认为100ms，可以实时动态调整配置。

* **慢日志记录阀值调整方法**

慢日志记录阀值可以通过以下配置文件中如下几个参数进行调整：

1. operationProfiling.mode，日志模式设置，不同取值及其含义如下：

0：不记录日志

1：只记录执行时间超过slowOpThresholdMs阀值的慢日志

2： 记录所有SQL日志，不管时延高还是低

1. operationProfiling.slowOpThresholdMs，慢日志记录阀值，含义如下：

**注意：**该配置只有在operationProfiling.mode为1的时候才有效。

如上，operationProfiling.mode和operationProfiling.slowOpThresholdMs分别对应配置文件中的慢日志模式和日志阀值设置。此外，我们还可以动态实时对这两个参数进行调整，对应命令如下：

//设置mode，取值0或者1

db.setProfilingLevel(0)

//获取mode取值

db.getProfilingLevel()

//设置慢日志记录阀值，默认ms

db.setProfilingLevel(1,100)

//获取慢日志阀值，模式ms

db.getProfilingStatus().slowms

* **如何从日志文件分析慢日志**

mongodb包含各种不同类型，通常我们可以通过以下脚本来过滤慢日志中的核心信息：

* **找出文件末尾1000000行中存在扫表的操作，不包含oplog，getMore**

tail mongod.log -n 1000000 | grep ms |grep COLLSCAN |grep -v "getMore" | grep -v "oplog.rs"

* **找出文件末尾1000000行中所有的慢日志，不包含oplog，getMore**

tail mongodb.log -n 1000000 |grep ms | grep op\_msg | grep find | grep -v "oplog.rs" |grep -v "getMore"

* **找出文件末尾1000000行中执行时间1-10s的请求，不包含oplog，getMore**

tail mongodb.log -n 1000000 |grep ms | grep op\_msg | grep find | grep -v "oplog.rs" |grep -v "getMore" | egrep [1-9][0-9][0-9][0-9]ms

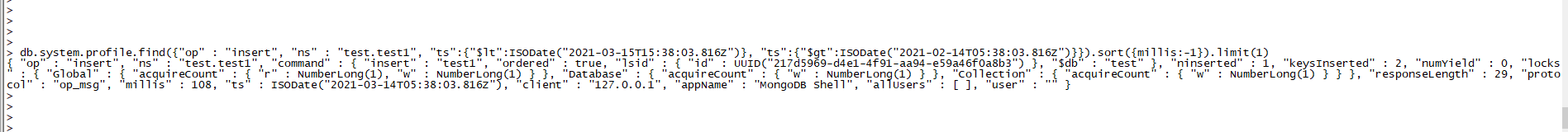
* **找出某一时间范围内(例如**2020-01-23T10:00-2020-01-23T10:10**)的慢日志，不包含oplog，getMore**

tail mongodb.log -n 1000000 |grep “2020-01-23T10:0”|grep ms | grep op\_msg | grep find | grep -v "oplog.rs" |grep -v "getMore"

上面列举的操作只是几个例子，如果想过滤其他关键字，则可以替换相关grep信息来实现，具体不在详述。

* **通过system.profile表获取库对应慢日志信息**

除了通过日志文件获取指定的慢查询外，我们还可以通过system.profile表获取慢日志，如下图所示：



应用举例，查找一段时间内，某表中insert最慢的操作，命令如下：

1. use test
2. system.profile.find({"op" : "insert", "ns" : "test.test1", "ts":{"$lt":ISODate("2021-03-15T15:38:03.816Z")}, "ts":{"$gt":ISODate("2021-02-14T05:38:03.816Z")}}).sort({millis:-1}).limit(1)

**【注意事项】：**

system.profile慢日志表需要管理员权限才可以访问，不同读写账号没权限，因此需要提前授权。如下：

1. //注意这里是test库
2. use test
3. //给test库创建一个只读账号test\_r
4. db.createUser({user: "test\_r",pwd: "xxxx",roles: [ {role: 'read', db: 'test'} ]})
5. //给test库创建一个访问system.profile慢日志表的可读角色
6. db.runCommand({ createRole: "slowlog\_role", privileges: [{ resource: { db: 'test', collection: 'system.profile'}, actions: ['find']}, ], roles: [{role: 'read', db: 'test'}]})
7. //切换到admin库
8. use admin
9. //给test\_r账号增加slowlog\_role角色，这样就可以访问system.profile表了
10. db.system.users.update({"\_id" : "test\_r"}, {$addToSet:{roles:{"role" : "slowlog\_role", "db" : "test" }}});

* **获取慢日志注意事项**

获取慢日志一定要注意以下几个要点：

1. 如果从日志文件过滤，如果日志文件中的日志打印非常大，切记不要分析整个日志文件，因为非常慢并会消耗系统CPU和内存资源。由于一般我们线上集群抖动都是查看最近的慢日志信息，因此可以通过tail来规避大文件的日志分析。
2. 通过system.profile慢日志表获取慢日志，切记提前count该表数据大小，如果数据量很大，切记查询的时候带上ts字段，查询某一时间范围的慢日志，同时带上limit条件，这样查询会更快，否则会扫描整个system.oplog慢日志表，这样会对业务有影响。
3. 慢日志默认时延为100ms，如果发现CPU高，但是又没用慢日志，则可以调下慢日志阀值来记录请求日志，从而判断是否有没走索引的请求。

# 5. 操作系统性能瓶颈分析

系统层面性能分析工具主要有：top、iostat、pstak、ptress、perf、iotop、isof等，具体请参考对应工具说明。

**【注意事项】：**

以下系统指标过载，需要特别注意：

1. 服务器CPU使用率超过80%，需要判断是否流量过大，如果流量不高，则需要分析是否索引不是最优引起CPU消耗过高。
2. 服务器磁盘IO util超过60%，需要关注是否有大量持续性写入，磁盘IO过高，会有抖动。如果是写流量过高引起，则可以考虑扩容分片了。
3. 服务器mem内存超过服务器85%，则需要关注mongo实例内存消耗是否异常，是否没用做cacheSize限制、是否又不合理的排序请求、是否同时有多个索引添加等。

## 6. 开源mongodb详细监控套记

#### 开源方案可以参考以下组件，相关安装使用说明请参考对应官方文档，这里只是给出一个简单的开源方案：

#### Grafana+Prometheus+**node\_exporter+mongodb\_exporter**

* **服务端组件：**  
  Prometheus #服务端  
  Grafana #前端展示
* **客户端组件：**  
  node\_exporter  
  mongodb\_exporter