调研文档

1 MongoDb

文档性 NoSql 数据库

1.1 Mongo 分片集群模式

部署文档.docx

部署文档 ：

生产环境部署Mo
ngoDB集群——性

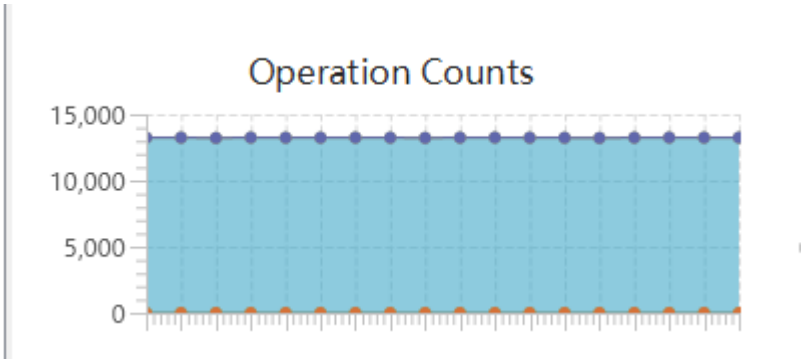
优化文档 ：

1.2 性能测试

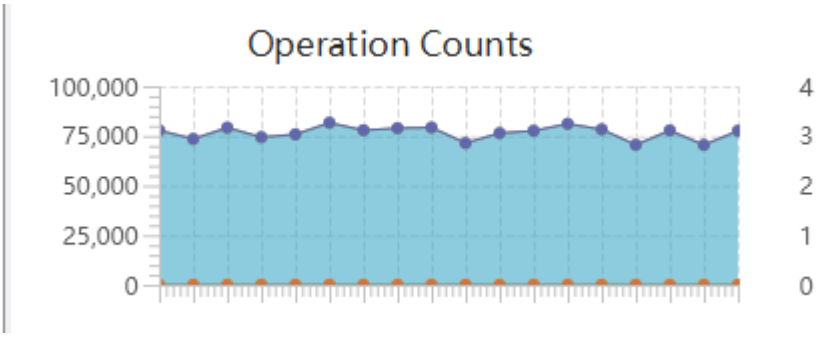
测试方式 1： YCSB 专门的 Nosql 压力测试
测试基准： 100Thread 2000000W doc 8 Column 200length/Column Value 1K Insert 100%
磁盘存储大小指标 CPU 内存指标

1.2.1 压力工具测试

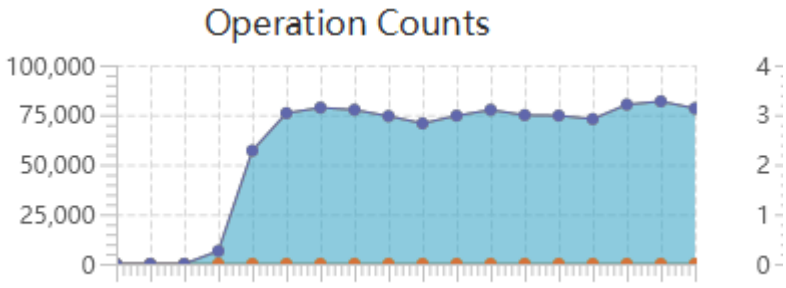
1.2.1.1 单实例



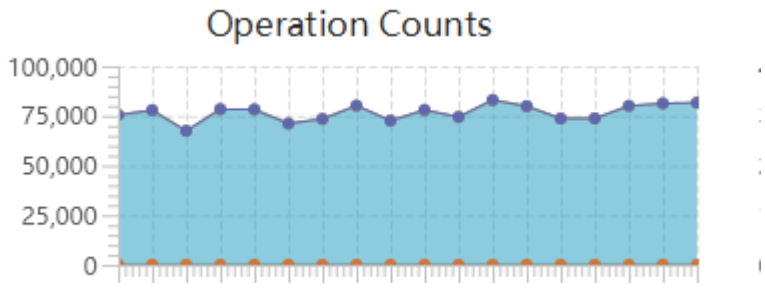
70% Insert 30% Read



50% Insert 20% Update 30% read



50% Insert 50% Read



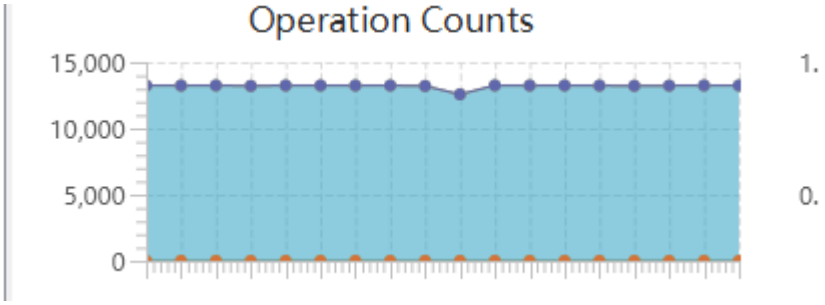
CPU　基本占用 1 到 2 个 Core
内存　占满
磁盘空间

```
> show dbs
admin   0.000GB
local   0.000GB
test    3.789GB
```

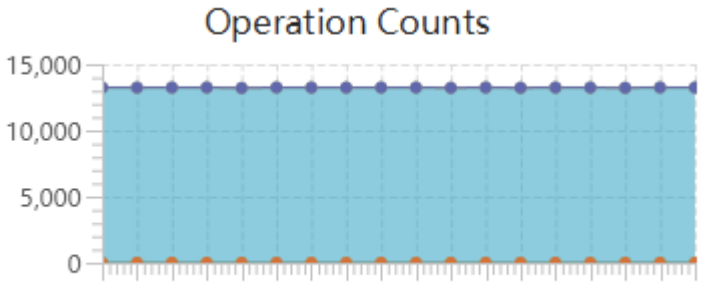
Swap　暂用没超过 内存的 1%-2%　合适

1.2.1.2 分片集群模式

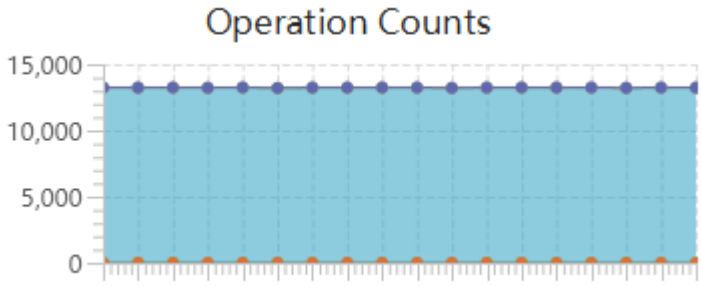
100% Insert



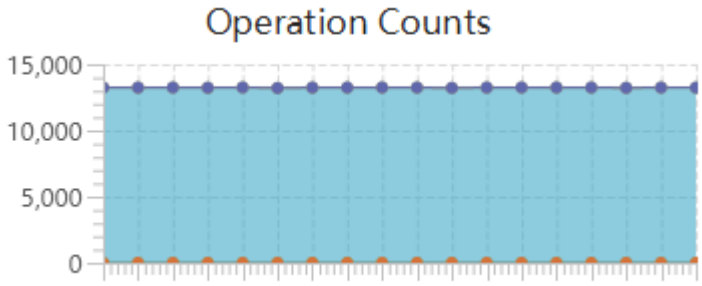
70% Insert 30 % Read



50%　Insert 30%Read 20% Update

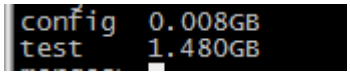


50 % Insert　50% Read



CPU 1 到两个 Core

内存占满

磁盘空间 

1.4 *2

测试结果 ：写 1.3W 左右/s (可以看作是 3 个实例中的一个) 估算：1.3*3

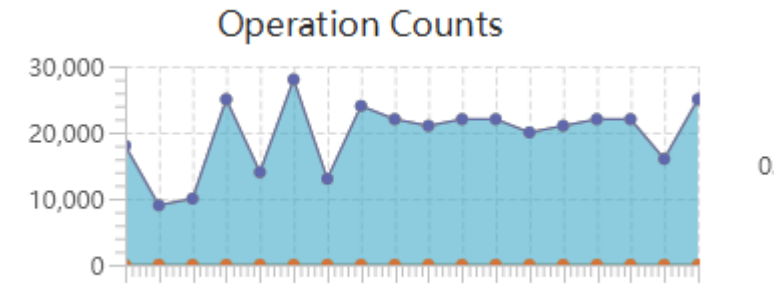
性能低原因 ：产生的 ID 没有进行 HASH 未同时向集群中的多个实例同时进行 Insert 操作

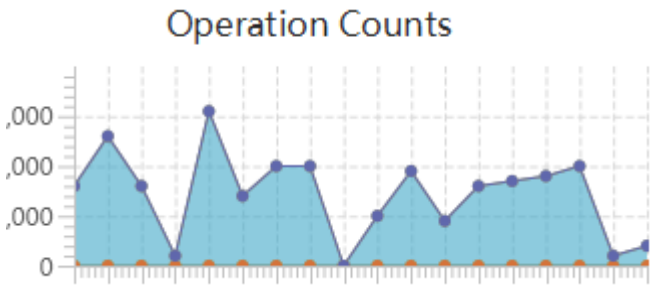
1.2.2 代码测试

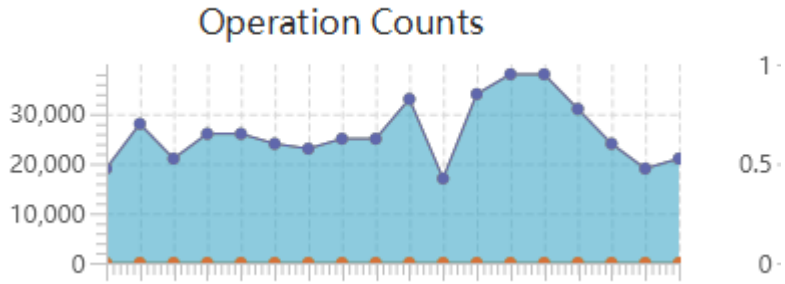
测试方式： 把 Mysql T_location20160216 5000W 数据 100 个线程 读 写

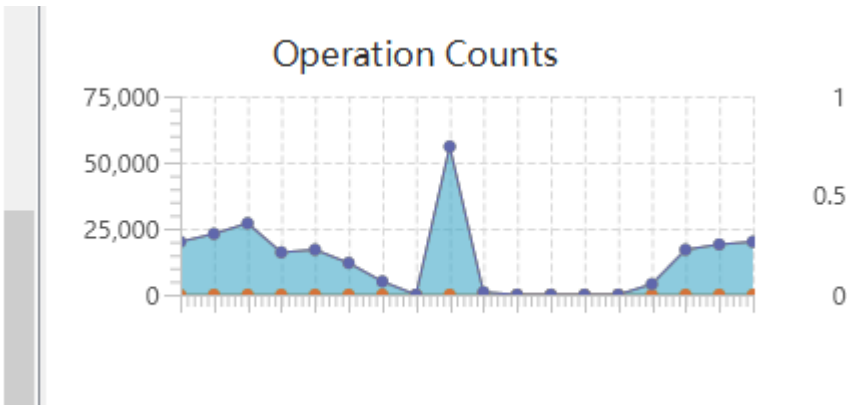
缺点 ： 不能保证同一时间多线程同时 Insert

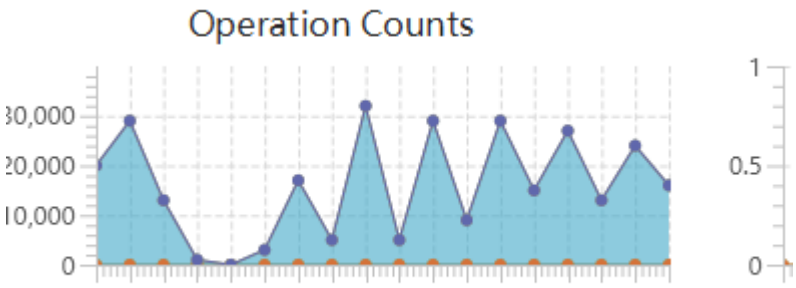
1.2.2.1 分片集群 Insert

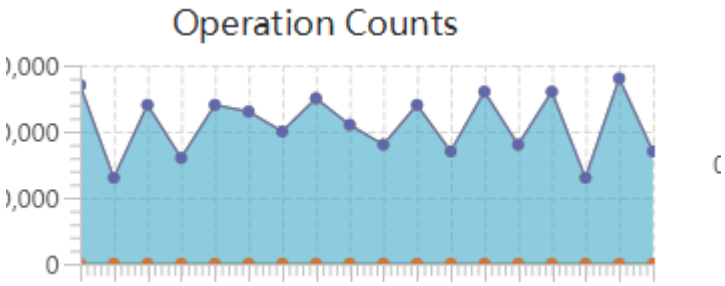


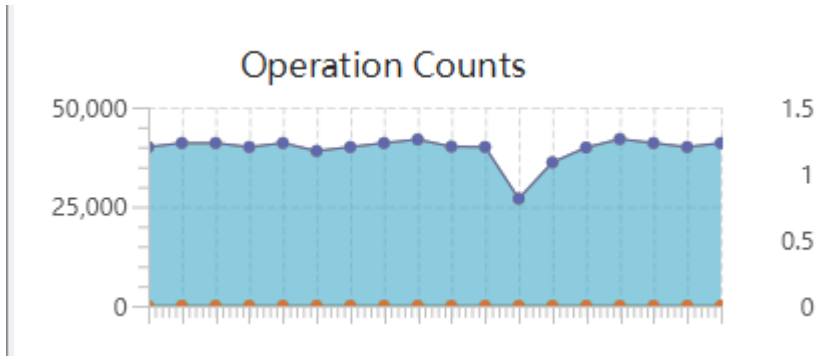






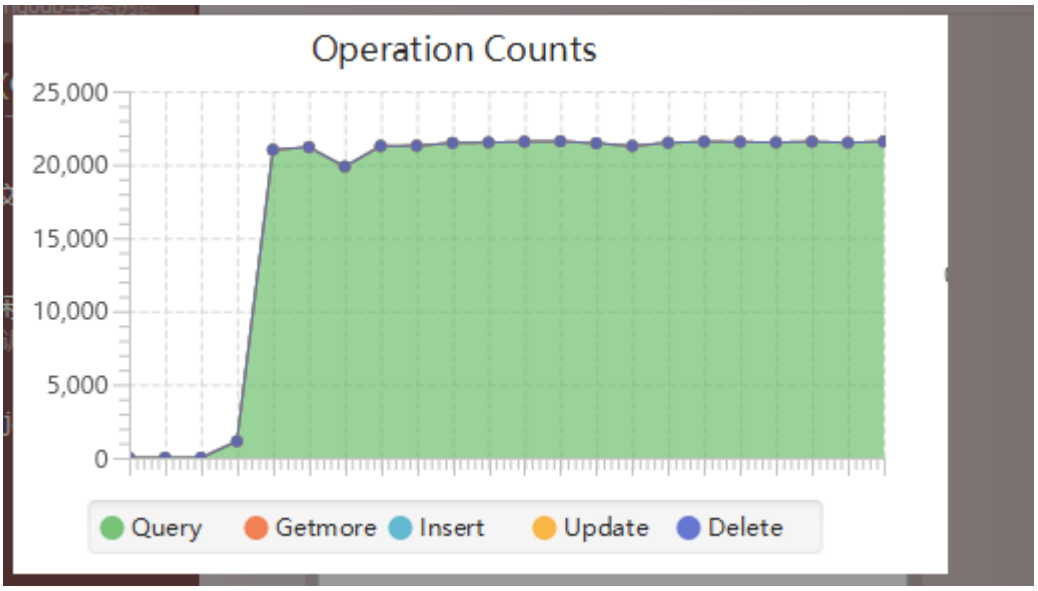


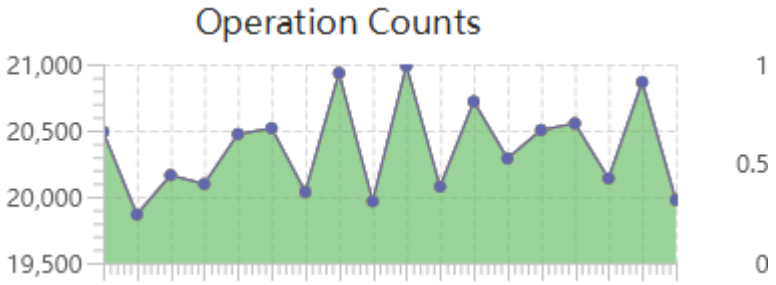


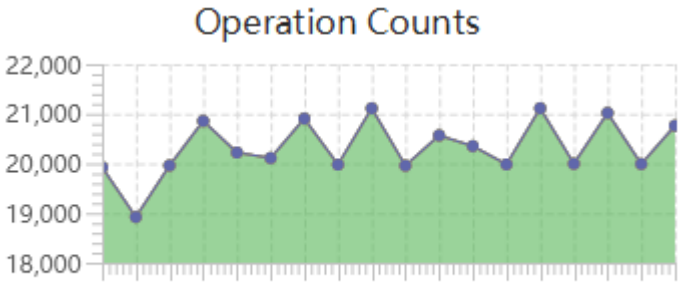


1.2.2.2 分片集群 Read

在写的同时读性能 差不多 2W







1.2.2.3 分片集群 测试机器状态

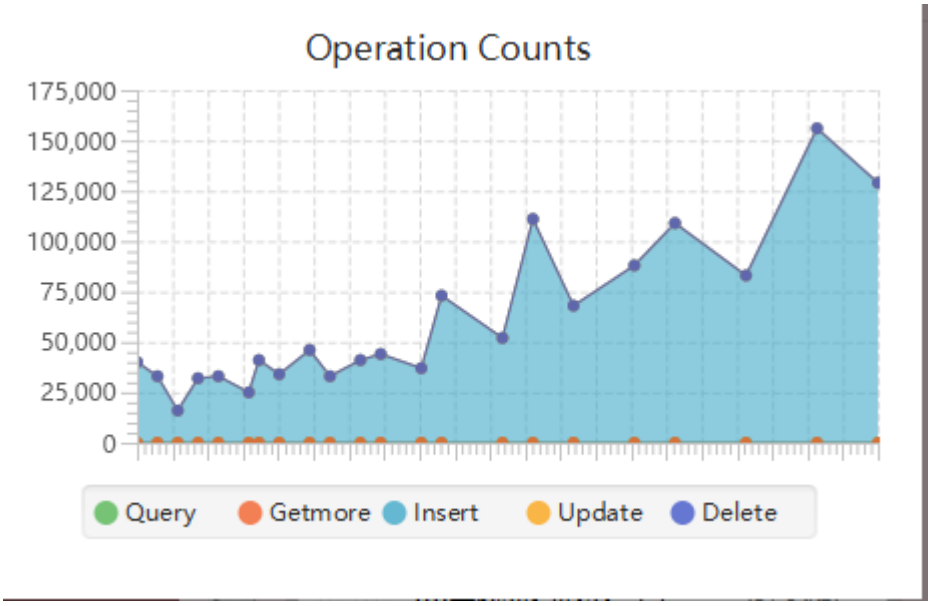
CPU
内存占满
空库写入性能可以达到 5W　数据量上来后 基本上稳定在 4W-5W 左右
磁盘存储量

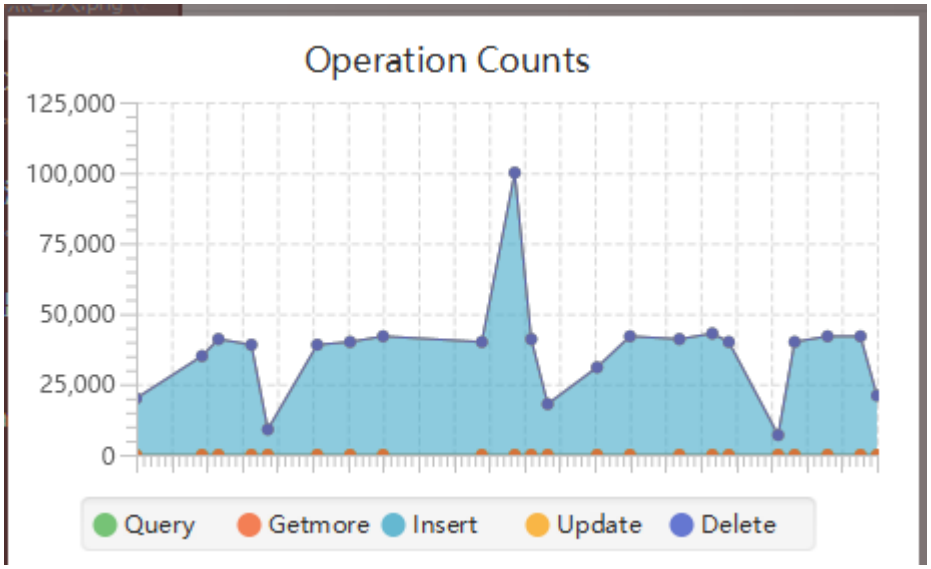
```
 data : 3.03GiB docs : 5649382 chunks : 98
 Shard shard1 contains 36.02% data, 36.02% docs in cluster,
 Shard shard2 contains 34.17% data, 34.17% docs in cluster,
 Shard shard3 contains 29.79% data, 29.79% docs in cluster,


mongos> show dbs
config  0.008GB
test    1.121GB
mongos> db.t_Location.count()
5649395
mongos>
```

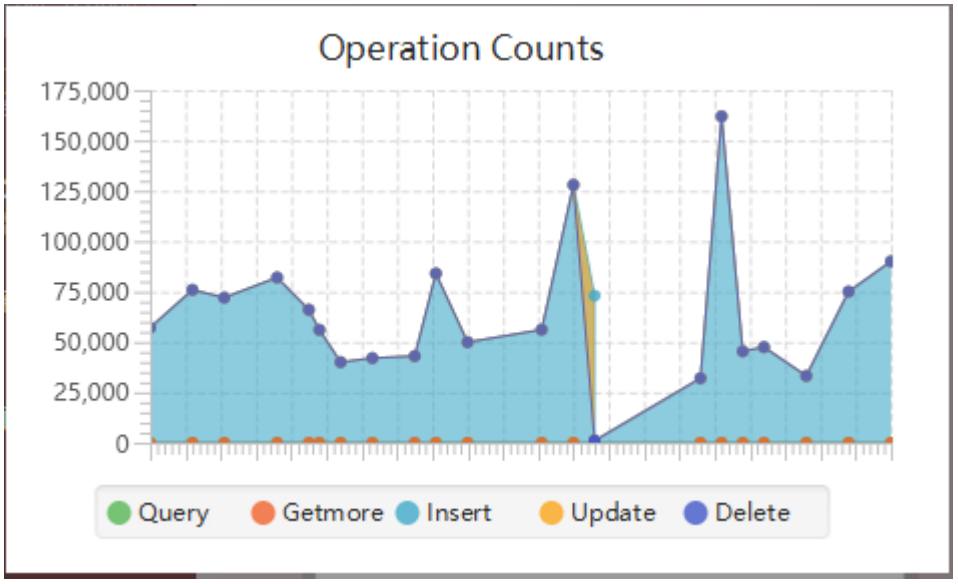
加上副本大约 2.4G

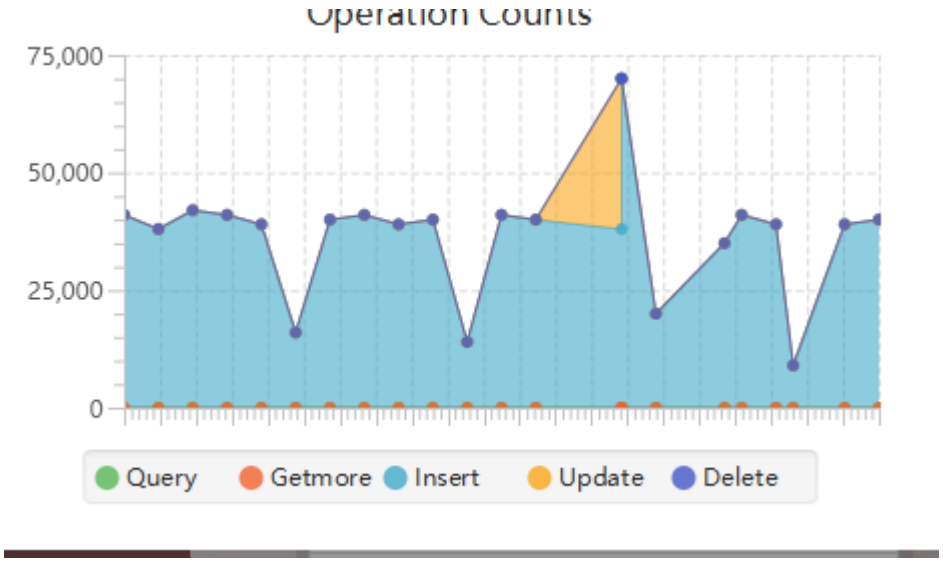
Swap 禁用 未超过正常值

1.2.3.1 单实例 Insert




Operation Counts
125,000
100,000
75,000
50,000
25,000
0
Query
Getmore
Insert
Update
Delete

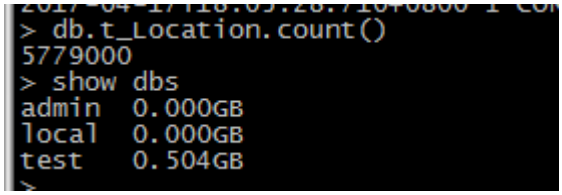

Operation Counts
175,000
150,000
125,000
100,000
75,000
50,000
25,000
0
Query
Getmore
Insert
Update
Delete


75,000
50,000
25,000
0
Query
Getmore
Insert
Update
Delete

1.2.3.2 单实例 READ

1.2.3.3 单实例机器状态

内存占满
CPU 1 到 2 Core
磁盘存储空间



Swap 未达到异常值

1.2.3 单实例 集群分片比较

存储量 ：相同的数据量 数据大小一样 集群有另外一个副本存在
CPU 都是占用 1 到 2 个 Core
内存都占满
Swap 优化后都处于正常值

集群分片性能没有单实例 Insert 性能高 原因：

1 ： 3 台机器每天机器分别存在 3 个不同的 mongod 实例
2 ：集群分片 3 台机器 有其他的 JAVA 进程 会存在部分影响，但是影响不大
3 ：从数据库中查询不同区间的数据 同时 Insert 的线程 无法确定

1 Redis

性能测试基准： 利用 Redis-benchmark
版本 ： redis 3.0.2

1.1 各指标

1.1.1Clster

```
[root@data3 cluster2]# redis-benchmark -h 10.10.11.16 -p 7000 -q -d 100
PING_INLINE: 164203.61 requests per second
PING_BULK: 165289.25 requests per second
SET: 164473.69 requests per second
GET: 167224.08 requests per second
INCR: 159235.66 requests per second
LPUSH: 165837.48 requests per second
LPOP: 166389.34 requests per second
SADD: 165289.25 requests per second
SPOP: 164203.61 requests per second
LPUSH (needed to benchmark LRANGE): 165562.92 requests per second
LRANGE_100 (first 100 elements): 167785.23 requests per second
LRANGE_300 (first 300 elements): 166389.34 requests per second
LRANGE_500 (first 450 elements): 167785.23 requests per second
LRANGE_600 (first 600 elements): 167504.19 requests per second
MSET (10 keys): 107296.14 requests per second
```

在并发大量往里面写入数据时 每个指标都下降 3W 图片丢了

1.1.2 Single

```
PING_INLINE: 54764.51 requests per second
PING_BULK: 53966.54 requests per second
SET: 59031.88 requests per second
GET: 59101.65 requests per second
INCR: 59880.24 requests per second
LPUSH: 59523.81 requests per second
LPOP: 59417.71 requests per second
SADD: 60240.96 requests per second
SPOP: 59772.86 requests per second
LPUSH (needed to benchmark LRANGE): 60132.29 requests per second
LRANGE_100 (first 100 elements): 60864.27 requests per second
LRANGE_300 (first 300 elements): 60753.34 requests per second
LRANGE_500 (first 450 elements): 60386.47 requests per second
LRANGE_600 (first 600 elements): 60679.61 requests per second
MSET (10 keys): 51599.59 requests per second
```

1.3 Master -Slave

```
PING_INLINE: 84602.37 requests per second
PING_BULK: 115606.94 requests per second
SET: 86505.19 requests per second
GET: 113895.21 requests per second
INCR: 112866.82 requests per second
LPUSH: 100806.45 requests per second
LPOP: 103412.62 requests per second
SADD: 93283.58 requests per second
SPOP: 108813.92 requests per second
LPUSH (needed to benchmark LRANGE): 99900.09 requests per second
LRANGE_100 (first 100 elements): 32626.43 requests per second
LRANGE_300 (first 300 elements): 12855.12 requests per second
LRANGE_500 (first 450 elements): 8809.02 requests per second
LRANGE_600 (first 600 elements): 6082.73 requests per second
MSET (10 keys): 44072.28 requests per second
```

在高并发写时测试

```
PING_INLINE: 69783.67 requests per second
PING_BULK: 80515.30 requests per second
SET: 35893.75 requests per second
GET: 81300.81 requests per second
INCR: 73046.02 requests per second
LPUSH: 35435.86 requests per second
LPOP: 73260.07 requests per second
SADD: 74962.52 requests per second
SPOP: 76687.12 requests per second
LPUSH (needed to benchmark LRANGE): 37707.39 requests per second
LRANGE_100 (first 100 elements): 27831.90 requests per second
LRANGE_300 (first 300 elements): 11957.43 requests per second
LRANGE_500 (first 450 elements): 7522.19 requests per second
LRANGE_600 (first 600 elements): 5295.21 requests per second
MSET (10 keys): 5552.16 requests per second
```

1.2 Master -Slave 集群同步

1.2.1 3.0 同步方法：

第一次 ： salve -> master master bgsave 然后传输 rdb 到 slave 节点
Slave 节点 load rdb Master Slave 分别记录当前节点 ip 和 offset
第 N 次 master 有执行 set Insert Del 等命令 同步上次 slave 节点的 offset 指令数据

1.2.1 存在问题

如果长期由于网络原因 断开连接 Master 的更改操作很多 超过了 redis 复制积压缓冲区 会发生 全备操作

1.2.2 解决方案

1 增大 复制积压缓冲区 大小

2 及时发现问题 修复问题进行同步

3 同一个 master 下不要存在多个 slave 减少 master 压力 slave 下可以存在 slave

4 使用一些中间件 Redis-port 目前处于编译 解决遇到问题 性能未知（类似于 Mysql 同步中间件 canal）