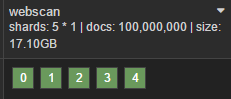
一亿条数据（大小17.10G）



内存占用情况：首先segment memory是必不可少，

<http://172.16.39.160:9305/_cat/nodes?v&h=name,port,sm>



之后使用filter查询，filter cache（又称query cache）用来缓存使用过的filter的结果集，需要注意的是这个缓存是常驻heap，无法gc的。（默认是百分之十的heap，这个不能确定），在heap没什么压力的情况下，才考虑加大这个设置。（通过indices.queries.cache.size 来设置，可以是10%，也可以是512mb这种）

然后进行的聚合查询，如果在上面的size设置为0的时候，结果集也会被缓存到heap中（又称request cache），这个缓存默认是关闭的，需要手动开启：

设置索引的\_settings ："index.requests.cache.enable": **true**

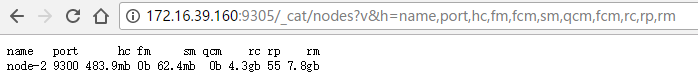
开启缓存后，需要在那些需要缓存的搜索请求上加上request\_cache=true这个参数才会使得查询请求被缓存。及index/\_search?request\_cache=true

且，查询语句中需要设置size为0，另：使用脚本执行查询的时候，必须制定request\_cache为false，因为脚本的执行结果是不确定的。默认情况下缓存的大小是jvm堆的1%大小，当然也可以在yml文件中通过indices.requests.cache.size: 1% 来设置。

设置fileddata占用内存大小：

indices.fielddata.cache.size: 40%

这是上述的描述，在做了几个filter之后：





这里的内容就不符合设想了，且clear cache后，heap大小还是没有减少，这个就是个问题了，通过重启可以解决问题，但是治标不治本。

后重启之后，通过es-sql一个关联查询：

select w.\*,i.id from webscan w join indextest2 i on w.id = i.id

heap使用增加300M，这里暂时没法释放，是否是因为这个机制因为通过插件实现，所以缓存无法通过常规的命令释放？（后证实这是jvm的heap问题，通过的是gc来释放）

## 聚合测试

{

"size": 0,

"aggs": {

"agg1": {

"terms": {

"field": "net\_recive"

},

"aggs": {

"agg2": {

"terms": {

"field": "net\_send"

},

"aggs": {

"agg3": {

"terms": {

"field": "cpu\_used"

},

"aggs": {

"agg4": {

"avg": {

"field": "mem\_used"

}

}

}

}

}

}

}

}

}

}

上述造成es OutOfMemory挂掉，后自己恢复重启（？？），heap是释放掉了，但是qcm内容还在（query\_cache\_memory）。

后来测试的时候直接报错，数据量超过堆的大小，所以结果

Jvm的gc原理，意思是heap使用是不断上涨的，当达到某个值的时候，开始gc，gc的效率非常之高，那如果oom了，heap是否会变化，是变成gc后的还是不变。

es自身有一个断路器的机制，包括fileddata的断路器，request的断路器，以及整体的断路器机制，查询的时候，es在进行一个预估，简单的统计出需要的大小，与断路器的大小进行对比，超过则报错，没超过则继续执行。

gc机制的话，heap大小达到设置的值之后，进行gc，gc速度非常快，所以

（160的配置是i7-3520，2核4线程）