Elastic Stack@信息安全部

亢伟楠

2018.09.08

我

亢伟楠

喜欢研究Jvm方向的风控工程师

参与大量开源项目：
Elastic Stack
Apache Skywalking
JITWatch

Github:github.com/cyberdak

Wecaht:cyberdak

目录

我们用Elastic Stack 做什么

Nosql database

少量搜索场景

Saas下程序日志

业务日志

业务报警

数据分析

Near real time

99% term 查询

Userid:xxxx

Level:ERROR

集群规模

3 node (master + data)

@5.4.2

8 core E5V2 * 2 16C32T

128gb

6TB * 10 RAID5

3 master node(1 master + 2)

15 data node

8 core E5V2 * 2 16C32T

128gb 8TB SATA * 12 RAID10

@5.6.8

140tb

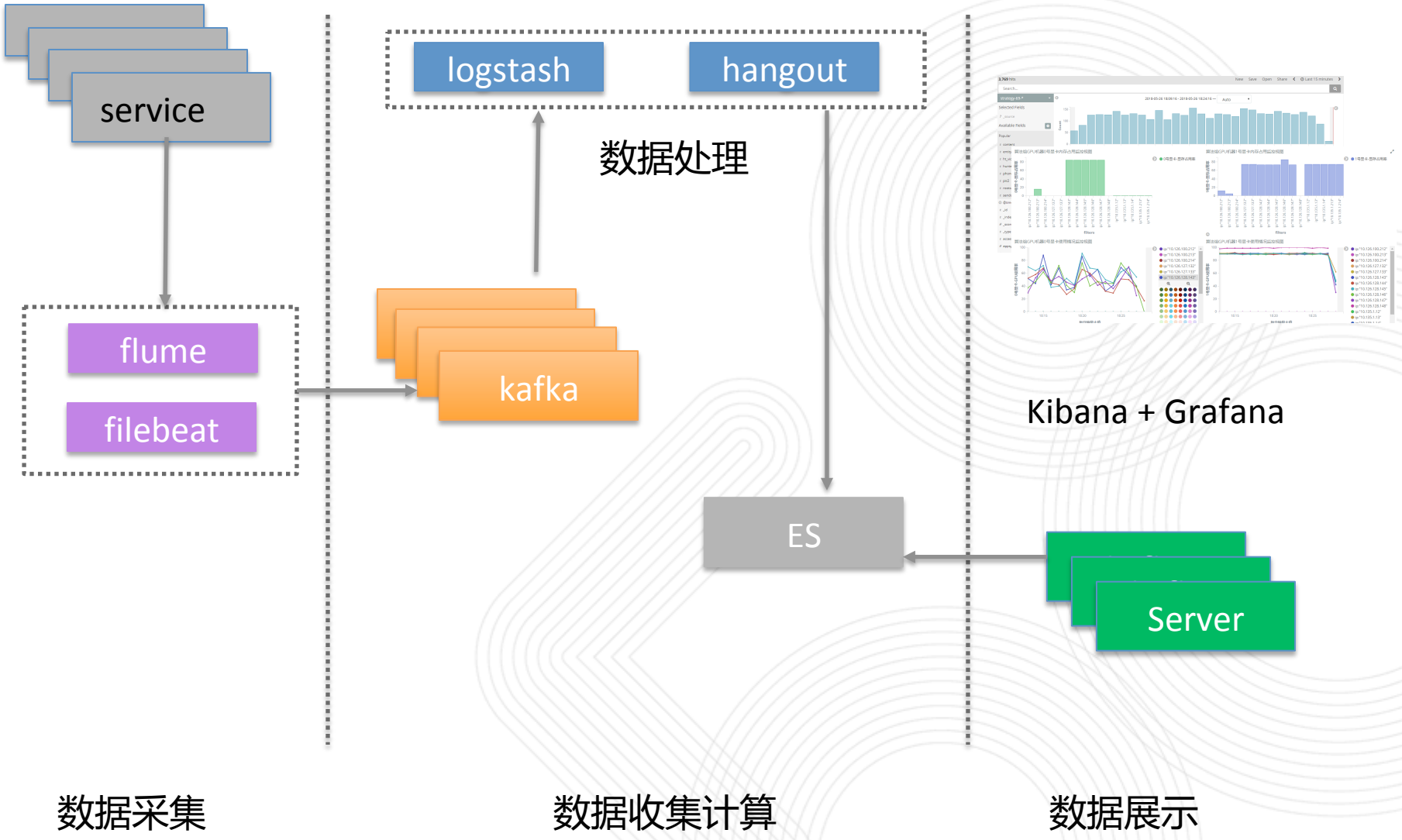
2700 indices

16000 shards

140亿 docs

5W write/s

架构


service
flume
filebeat
logstash
hangout
数据处理
kafka
ES
Server
Kibana + Grafana
数据采集
数据收集计算
数据展示

需求

数据保存90天，实时查询

不接受close
不接受snapshot

新老数据混合查询

90T

存储选择

HDD OR SSD ?

无脑 SSD ?
存储上线，数据本身要求保存时间长

SAS / SATA
分布式一致性算法的提升，降低了存储成本，SATA的可靠性也能接受

indexer

- Logstash
  - Jbuby实现，2.4.3是支持kafka 0.8的最新版本
  - 受限于 kafka 0.8 ，只能使用2.4.3版本，Jruby 中间层转换太多。
- Hangout
  - 纯java实现的，input_kafka 效率提升
  - 500% 效率提升
- Logstash 5.5+
  - 新的java实现，性能提升

数据展示

- Kibana
  - 产品，运营数据不求人
  - 节省RD编写sql时间
  - 降低使用成本：成本
    - 官方改进 https://github.com/elastic/kibana/issues/6515
    - 民间开源方案 http://t.cn/R8Opez4
- Grafana
  - 更偏向运维方向的数据图表

数据可靠性

数据可以离线，但是不能丢

至少1备份
至少RAID 1

1备份也会丢数据

1P + 1R
P掉线
提升R为P，开始复制R
P再次掉线
Shard 丢失

反思：HDFS 三副本

RAID

RAID	写惩罚	空间利用率	可靠性	写性能	读性能	特殊
0	1	100%	低	高	高	
1	2	50%	高	低	低	
10 ★	2	50%	高	中	中	
5 ★	4	(N-1)*单盘	中	低	低	大容量 SATA rebuild 容易挂
6 ★	6	(N-2)*单盘	高	低	低	

弹性？高可用？

1G 网卡的恢复峰值？

125MB/S * 60 * 60 = 543 GB

单机不宜存储过多数据

10G 网卡

网络不再是瓶颈

存储是

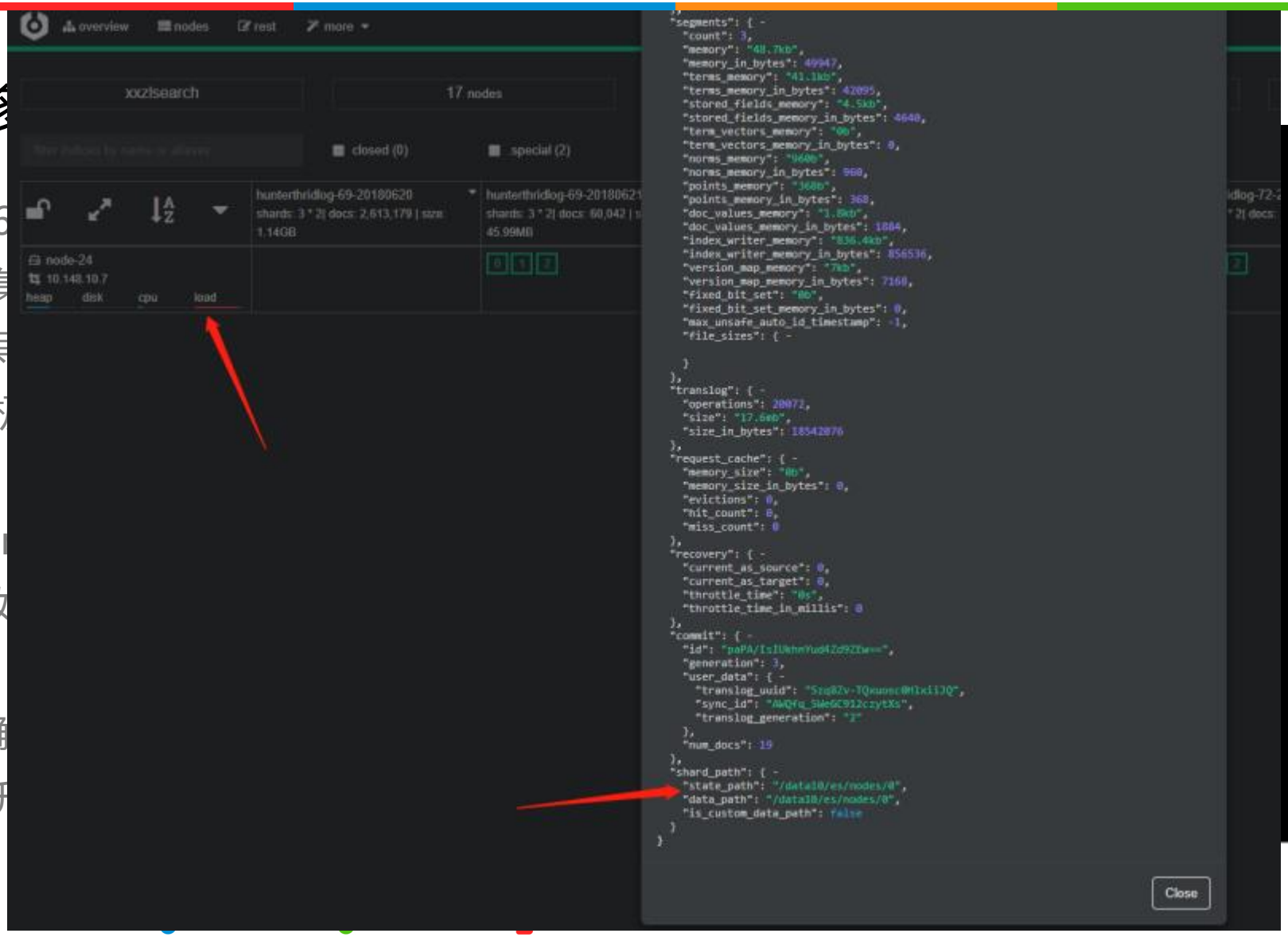
保证数据的快速恢复和重新索引

弹性？

弹性需要转移时间来保证

过长的恢复时间会让集群性能下降

一个服务器多个节点，可以增加资源使用率，但是增加恢复所需时间


overview
nodes
rest
more
xxzlsearch
17 nodes
closed (0)
special (2)
hunterthridlog-69-20180620
shards: 3 * 2| docs: 2,613,179 | size: 1.14GB
node-24
10.148.10.7
heap
disk
cpu
load
"segments": {
"count": 3,
"memory": "48.7kb",
"memory_in_bytes": 49947,
"terms_memory": "41.1kb",
"terms_memory_in_bytes": 42095,
"stored_fields_memory": "4.5kb",
"stored_fields_memory_in_bytes": 4640,
"term_vectors_memory": "0b",
"term_vectors_memory_in_bytes": 0,
"norms_memory": "960b",
"norms_memory_in_bytes": 960,
"points_memory": "368b",
"points_memory_in_bytes": 368,
"doc_values_memory": "1.8kb",
"doc_values_memory_in_bytes": 1884,
"index_writer_memory": "836.4kb",
"index_writer_memory_in_bytes": 856536,
"version_map_memory": "7kb",
"version_map_memory_in_bytes": 7168,
"fixed_bit_set": "0b",
"fixed_bit_set_memory_in_bytes": 0,
"max_unsafe_auto_id_timestamp": -1,
"file_sizes": {
"translog": {
"operations": 20072,
"size": "17.6mb",
"size_in_bytes": 18542076
},
"request_cache": {
"memory_size": "0b",
"memory_size_in_bytes": 0,
"evictions": 0,
"hit_count": 0,
"miss_count": 0
},
"recovery": {
"current_as_source": 0,
"current_as_target": 0,
"throttle_time": "0s",
"throttle_time_in_millis": 0
},
"commit": {
"id": "paPA/IslUkhnYud4Zd9ZIw==",
"generation": 3,
"user_data": {
"translog_uuid": "Srq07v-TQxuosc0tlxi1JQ",
"sync_id": "AWQYq_SWeGC912czytXs",
"translog_generation": "2"
},
"num_docs": 19
},
"shard_path": {
"state_path": "/data10/es/nodes/0",
"data_path": "/data10/es/nodes/0",
"is_custom_data_path": false
}
}
Close

内存多了也不行

Force merge

高CPU + IO 压力

Task Hung on 120 seconds timeout 大内存+慢速存储设备 常见原因

Sysctl.vm.dirty_ratio = 10

One box one app => 机器配置 128gb 可用内存 100 gb

Dirty_page = 10gb 过大

内存多了也不行

vm.dirty_ratio 可用内存的百分比，并非总内存的百分比

一旦到达该值，阻塞所有的写请求，讲内存中的数据刷入硬盘中，如果硬盘正好处在高IO下，应用性能疯狂下降

压力高过某个阈值的时候，会触发Linux hung task timeout .

系统假死

而此时es节点心跳正常，导致所有命令发送到该节点之后严重超时，恶劣情况将导致集群陆续crash

如何解决?

可用内存多的时候，降低vm.dirty_ratio以及vm.dirty_background_ratio

Sysctl 参数说明:https://www.kernel.org/doc/Documentation/sysctl/vm.txt

索引优化

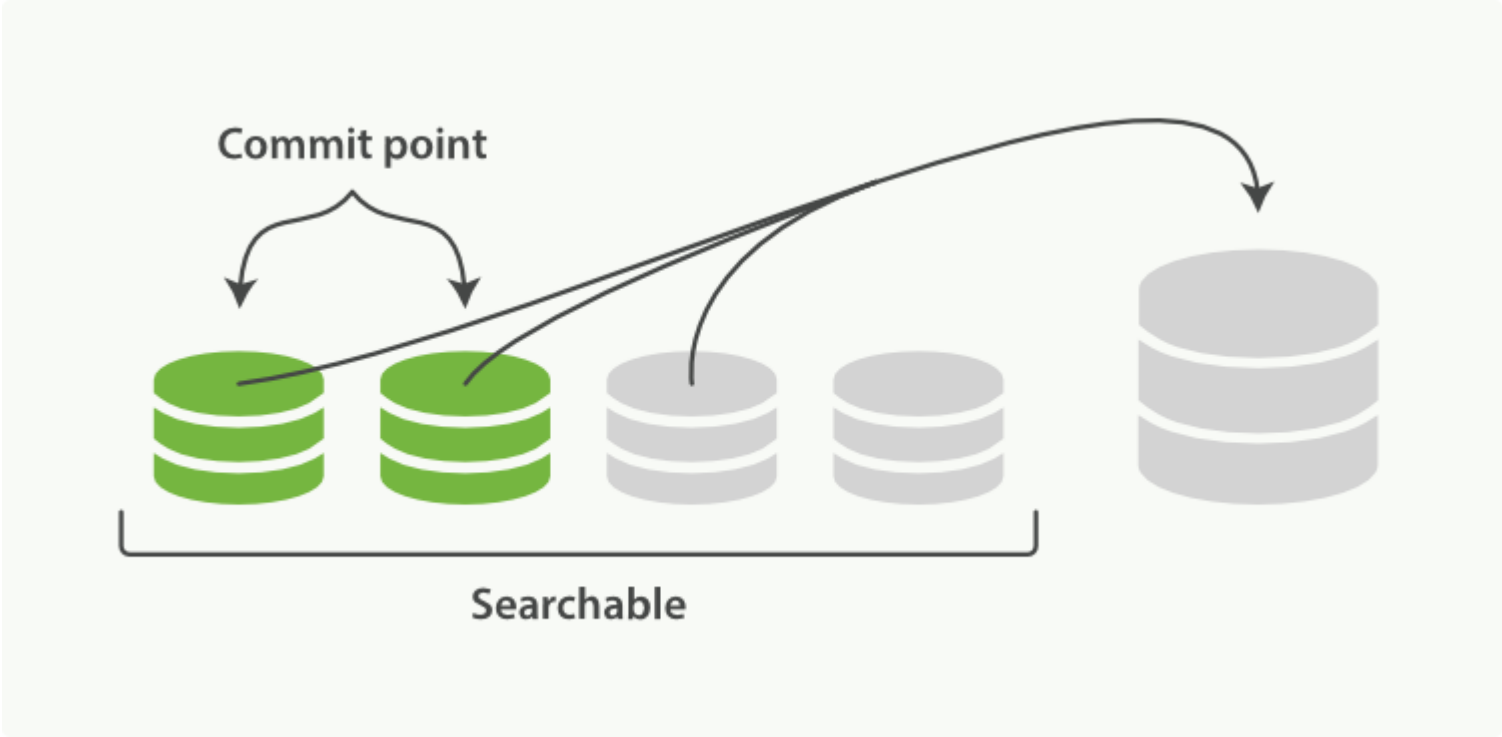
冷数据 refresh_interval : -1

- 需要规避 .kibana 等特殊索引，否则数据不会刷新，除非手动refresh
- 减少毫无必要的刷新

refresh_interval : 5s

- 避免刷新过长导致IO不均衡
- 过短导致集群刷新压力过大

Segment Merge



Merge thread

一秒一个segment

经过 backgroup 的merge thread 合并到一个大的segment

每一个segment都消耗文件句柄、内存和cpu运行周期，查询时轮流检查每个segment

合并之后，提升性能

Index.merge.scheduler.max_thred_count : default cpu /2 with min 4

SSD才能满足！！！

非SSD一定要设置1

优化内存和性能

- Force merge
  - Segment合并到1shard: 1segment
  - 占用cpu和内存
  - 低峰时期使用
- 使用正确的数据类型
  - 没必要分词的字段统统存储keyword
  - Ip真的用ip类型，不要用string
- 最佳实践 curator
  - 清理过期index
  - Index_settings
  - Force_merge
  - 根据计划自动运行
    - 机器比人靠谱

Curator 应用之一：自动 rebalacne

- 创建index时，选择最高可用空间磁盘。
- 定时任务删除index后，最高可用节点变更。
- 数据rebalance
  - 通过curator设置
  - 凌晨低峰时候开启rebalance
  - 调高rebalance限速
  - 凌晨结束时，关闭rebalance

集群健康状态

RED

有主节点丢失

YELLOW

主节点全部在线，有复制节点正在生成

GREEN

所有数据在线

集群到底在干什么

_nodes

```
::: {xnode-10}{FjeOB3b6SOm1YqpnKJXHkA}{g9I2teKjS1iXdqw9kNvZvw}[redacted]
Hot threads at 2018-09-06T14:28:19.345Z, interval=500ms, busiestThreads=10, ignoreIdleThreads=true:

29.3% (146.4ms out of 500ms) cpu usage by thread 'elasticsearch[xnode-10][bulk][T#24]'
  10/10 snapshots sharing following 2 elements
    java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
    java.lang.Thread.run(Thread.java:745)

23.4% (116.9ms out of 500ms) cpu usage by thread 'elasticsearch[xnode-10][[phoenix-2018.09.06][1]: Lucene Merge Thread #202]'
  9/10 snapshots sharing following 12 elements
    org.apache.lucene.codecs.blocktree.BlockTreeTermsWriter$TermsWriter.pushTerm(BlockTreeTermsWriter.java:907)
    org.apache.lucene.codecs.blocktree.BlockTreeTermsWriter$TermsWriter.write(BlockTreeTermsWriter.java:871)
    org.apache.lucene.codecs.blocktree.BlockTreeTermsWriter.write(BlockTreeTermsWriter.java:344)
    org.apache.lucene.codecs.FieldsConsumer.merge(FieldsConsumer.java:105)
    org.apache.lucene.codecs.perfield.PerFieldPostingsFormat$FieldsWriter.merge(PerFieldPostingsFormat.java:164)
    org.apache.lucene.index.SegmentMerger.mergeTerms(SegmentMerger.java:216)
    org.apache.lucene.index.SegmentMerger.merge(SegmentMerger.java:101)
    org.apache.lucene.index.IndexWriter.mergeMiddle(IndexWriter.java:4356)
    org.apache.lucene.index.IndexWriter.merge(IndexWriter.java:3931)
    org.apache.lucene.index.ConcurrentMergeScheduler.doMerge(ConcurrentMergeScheduler.java:624)
    org.elasticsearch.index.engine.ElasticsearchConcurrentMergeScheduler.doMerge(ElasticsearchConcurrentMergeScheduler.java:99)
    org.apache.lucene.index.ConcurrentMergeScheduler$MergeThread.run(ConcurrentMergeScheduler.java:661)
  unique snapshot
    org.apache.lucene.codecs.blocktree.SegmentTermsEnum.postings(SegmentTermsEnum.java:998)
    org.apache.lucene.index.MultiTermsEnum.postings(MultiTermsEnum.java:359)
    org.apache.lucene.index.MappedMultiFields$MappedMultiTermsEnum.postings(MappedMultiFields.java:127)
    org.apache.lucene.codecs.PushPostingsWriterBase.writeTerm(PushPostingsWriterBase.java:122)
    org.apache.lucene.codecs.blocktree.BlockTreeTermsWriter$TermsWriter.write(BlockTreeTermsWriter.java:866)
    org.apache.lucene.codecs.blocktree.BlockTreeTermsWriter.write(BlockTreeTermsWriter.java:344)
    org.apache.lucene.codecs.FieldsConsumer.merge(FieldsConsumer.java:105)
    org.apache.lucene.codecs.perfield.PerFieldPostingsFormat$FieldsWriter.merge(PerFieldPostingsFormat.java:164)
    org.apache.lucene.index.SegmentMerger.mergeTerms(SegmentMerger.java:216)
    org.apache.lucene.index.SegmentMerger.merge(SegmentMerger.java:101)
    org.apache.lucene.index.IndexWriter.mergeMiddle(IndexWriter.java:4356)
    org.apache.lucene.index.IndexWriter.merge(IndexWriter.java:3931)
    org.apache.lucene.index.ConcurrentMergeScheduler.doMerge(ConcurrentMergeScheduler.java:624)
    org.elasticsearch.index.engine.ElasticsearchConcurrentMergeScheduler.doMerge(ElasticsearchConcurrentMergeScheduler.java:99)
    org.apache.lucene.index.ConcurrentMergeScheduler$MergeThread.run(ConcurrentMergeScheduler.java:661)
```

安全时间

裸奔

Search guard 过重

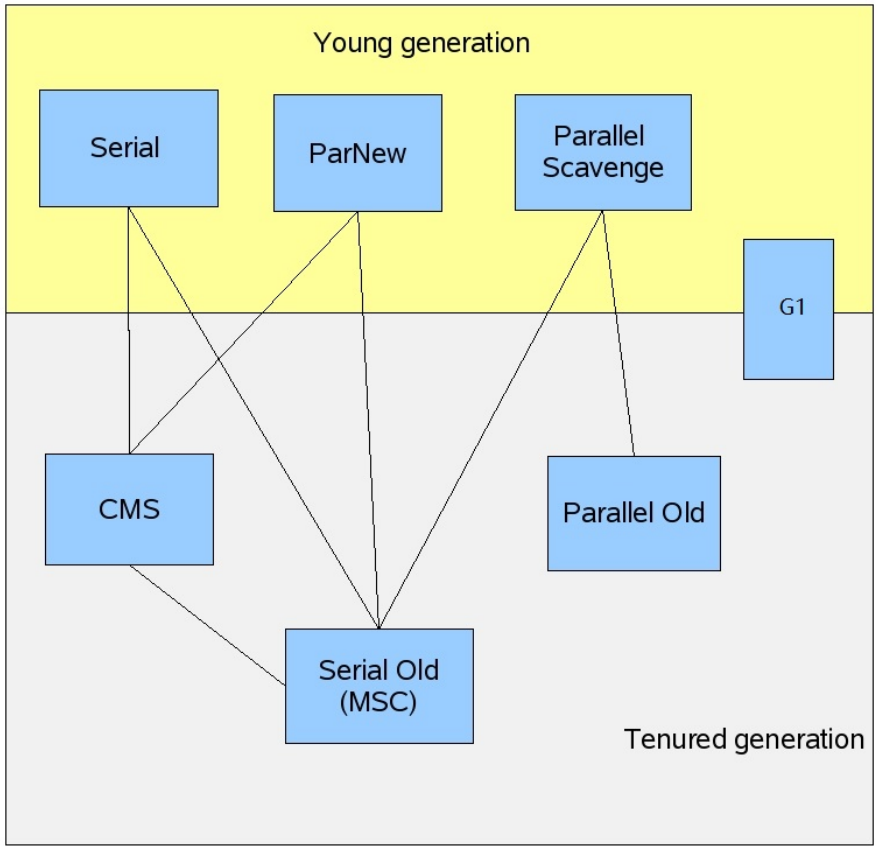
崩溃以后再无法恢复，无认证就无法连接master，没有master就没法恢复认证index

死循环

Restreadonly

Hotspot vm gc

- “Serial“ 单线程STW复制算法垃圾回收器。 别名 defNew , defualt new
- “ParNew” 多线程并行STW复制算法垃圾回收器。可以和cms搭配使用。Parallel new 的简称。
- “Parallel Scavenge” 多线程并行STW复制算法垃圾回收器 别名PSYoungGen
- "Serial Old" 单线程 STW标志-清除-整理算法回收器
- ⭐ “CMS” 大多数时间并发，低暂停回收器,主要阶段标记清除，辅助标记整理备份。
- “Parallel Old” 并行标记-整理算法回收器
- “G1” Garbage First： 并行并发



Hotspot vm gc

gc

Master node

分布式Shard status一致性算法

单master + 多eligible

集群性能受master节点限制

Master节点都干什么 – metainfo

- 分布式一致性协议数据
- 1 Cluster
  - n index
    - n*m*r_nums shard(primary + repica)
      - n*m*s field

gc

- Master node
  - 吞吐量优先
  - -XX:+UseG1GC
  - -XX:MaxGCPauseMillis=400
  - 如果因为mateinfo过多，导致master宕机，会引发集群雪崩。每个master node的处理能力对等，第一个撑不住的情况下，剩下的机器也撑不住。
  - 这种情况只能减少metainfo，拆分集群。

gc

- data node
  - Cache
  - Buffer
  - Thread , 很多线程池， bulk, index , search
  - 每个shard的merge thread
  - 分布式一致性协议中的选举线程

gc

- Indexr (logstash / hangout)
  - Object die young
  - 保证对象消逝在新生代，避免晋升，产生fullgc
  - -Xmx8g –Xms8g –Xms6g –XX:+UseParNew

选择正确的垃圾回收器

Hotspot VM 中的所有GC都会Stop the word . 程序暂停，集群阻塞

- 小堆
  - ParallelOldGC
  - 并行 = 高吞吐率，延迟在小堆能接受
- 4-6g
  - Cms
  - 大部分时间并发 = 并发标记清除+标记整理
- 6g+
  - G1GC（jdk9 默认回收器，7u40开始可用）
  - 多分区分代垃圾回收，99%情况没有full gc

选择正确的垃圾回收器

还是解决不了？

Azul Zing vm

C4

Continuously Concurrent Compacting Collector

完全并发压缩收集器

暂停时间= linux噪音级别

新的希望

JDK 11

ZGC 实验性低暂停算法

Azul zing vm c4 前身

Pauseless 算法

-XX：+UseZGC

(Lower is better)

ZGC

```
                avg: 1.091ms (+/-0.215ms)
    95th percentile: 1.380ms
    99th percentile: 1.512ms
  99.9th percentile: 1.663ms
 99.99th percentile: 1.681ms
                max: 1.681ms
```

G1

```
                avg: 156.806ms (+/-71.126ms)
    95th percentile: 316.672ms
    99th percentile: 428.095ms
  99.9th percentile: 543.846ms
 99.99th percentile: 543.846ms
                max: 543.846ms
```

Gc 指标分析

http://gceasy.io

指标

吞吐率 越接近 100%越好

暂停时间 越短越好

暂停次数 越少越好

如何获取gc日志

-XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:/dev/shm/gc.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=5M

免费的提升

总是使用最新的稳定版本，积极升级

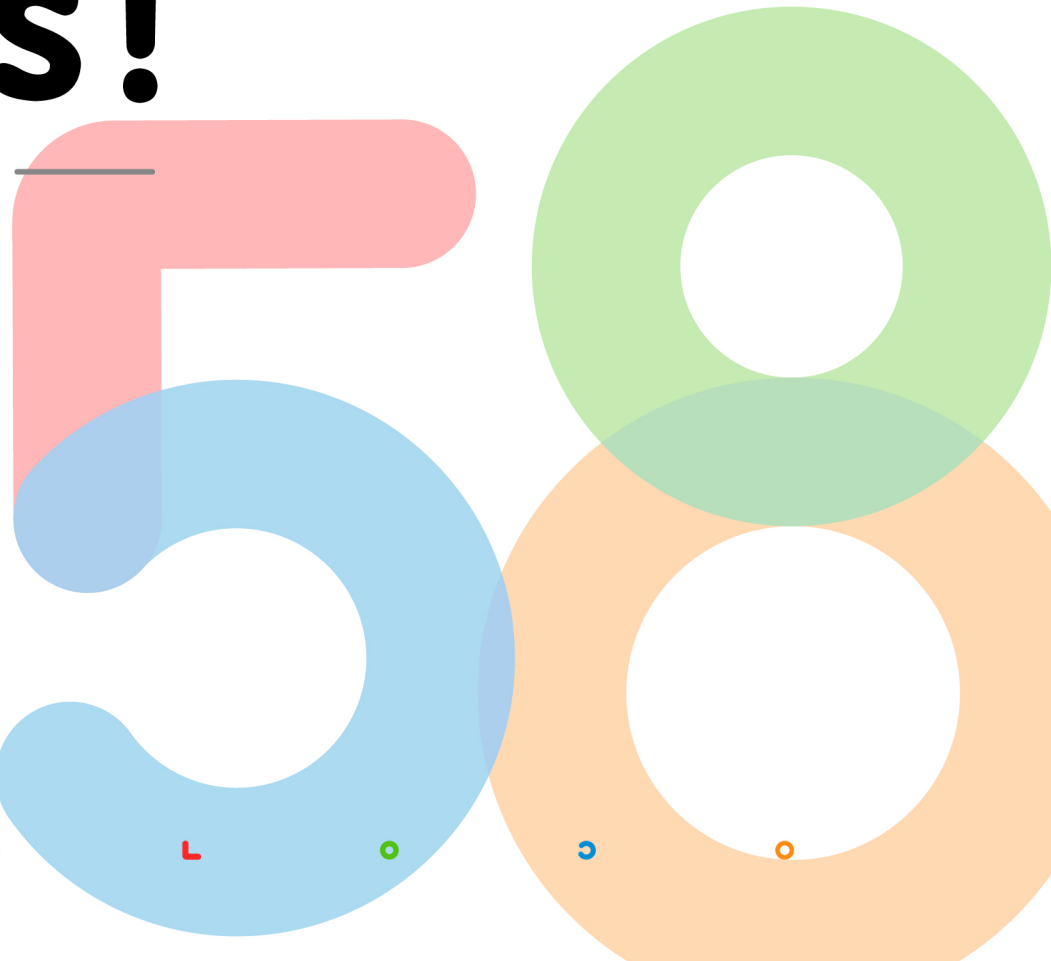
帮助开源社区发现问题

新版本总是有新的特性，更好的性能，更友善的操作

Jdk一样有免费的提升

Thanks!

让 生 活 更 简 单



elastic
中文社区

专业、垂直、纯粹的 Elastic 开源技术交流社区
https://elasticsearch.cn/