



GOPS 2020  
Shanghai

# GOPS

2020 全球运维大会  
- AIOps 风向标



指导单位：



主办单位：



大会时间：2020年11月27日-28日

大会地点：上海中庚聚龙酒店



# PB级大规模Elasticsearch集群运维与调优实践

高斌龙 腾讯云大数据工程师



# 高斌龙

腾讯云大数据工程师

高斌龙(bellen), 腾讯云大数据开发工程师, 目前专注于Elasticsearch云产品的研发工作。

CONTENTS

# 目录

- ① 背景介绍
- ② Elasticsearch运维场景与优化经验
- ③ Elasticsearch稳定性与成本优化
- ④ Elasticsearch自动化运维



# 背景介绍



# 手游业务

- 基于ELK构建日志系统
- 单日产生10TB数据
- 日志保留一年，不能删除





# Elasticsearch运维场景与优化经验

## 2.1 集群规模评估

### ■ 评估什么

- 节点CPU核数和内存

16核32G, 32核64G

- 节点磁盘类型

SSD、HDD？本地盘、云盘？

- 节点数量

集群整体规模

### ■ 评估依据

- 应用场景

日志场景、监控场景、搜索场景

- 查询或者写入量

写多读少？读多写少？

- 数据总量

根据原始数据量确定出ES集群需要多大的存储容量



## 2.1 集群规模评估

### ■ 评估准则

- 2核8G的节点可以承担5000写入qps
- 写入量大时优先选择64G内存节点规格
- ES实际存储空间为原始数据量2.8倍(1副本)
- 搜索场景优先选择大内存节点规格

## 2.2 索引配置评估

### ■ 评估什么

- 怎么划分索引

按天创建、按月创建？

- 索引的分片数量设置为多少

合理的分片数量可以提高写入性能和稳定性

- 索引字段mapping设置

text类型？keyword类型？

### ■ 评估依据

- 应用场景

日志场景、监控场景、搜索场景

- 每天产生的数据量多大

每天10TB？

- 查询类型

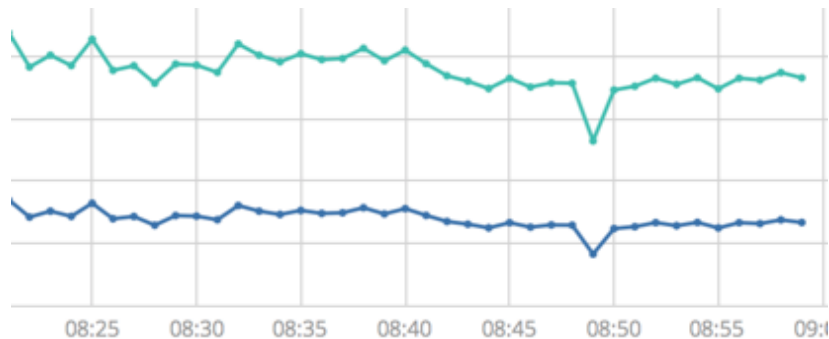
模糊搜索？精确匹配？

## 2.2 索引配置评估

### ■ 评估准则

- 单个分片保持在30-50GB
- 集群总体的分片数量不要超过3w
- 写入量大的场景，索引的分片数量保持和节点数量一致
- 合理利用ILM管理索引
- 合理利用SLM降低存储成本

# 实际案例




## 业务场景

- 游戏业务日志
- 写入峰值200w qps
- 单日10TB， 数据保存一年


## 集群规模与索引配置

- 60\*32核128G SDD + 50\*32核128G HDD
- 按小时创建索引->每两个小时创建索引
- 开启ILM， 自动shrink旧索引
- 实现SLM， 存量索引降低副本

 数据节点 x 60

标准型, 32核128G  
100GB SSD云硬盘 x 1

已配置

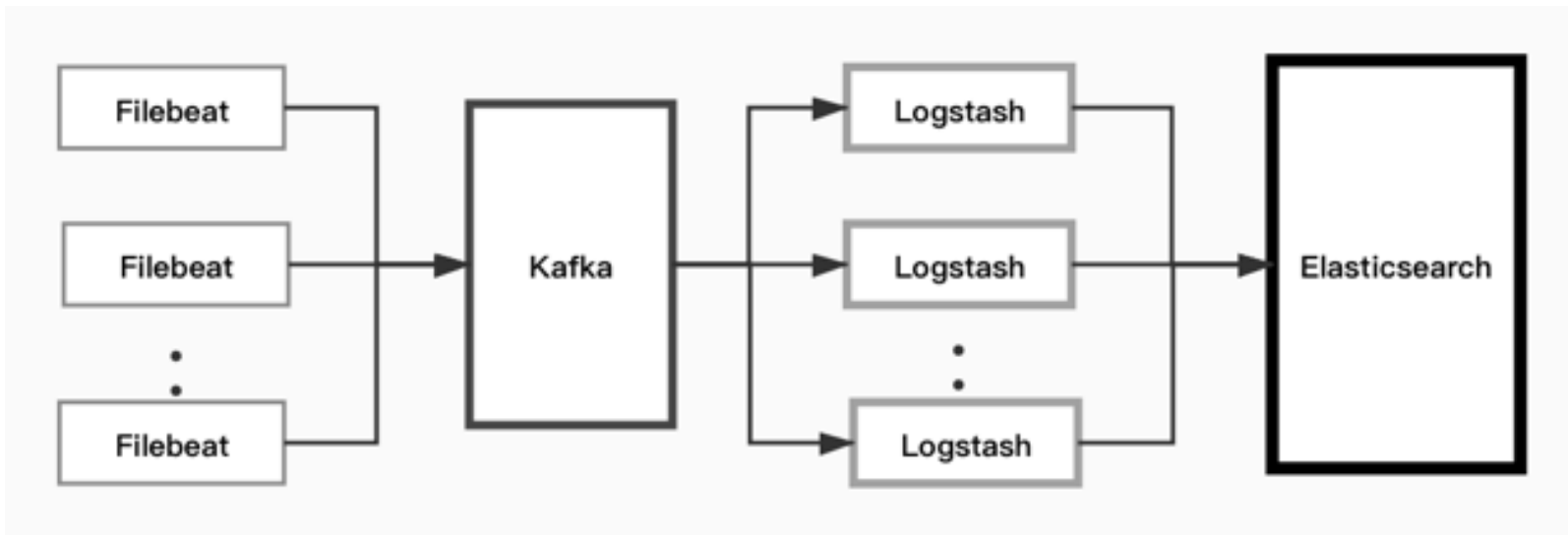
 冷数据节点 x 50

标准型, 32核128G  
100GB 高性能云硬盘 x 1

已配置

## 2.3 Logstash消费kafka性能调优

### 存在问题



增加多台logstash, 消费性能没有提升

不同topic消费速度不均匀

同一个topic下不同partition消费速度不均匀

## 2.3 Logstash消费kafka性能调优

### ■ 原因分析

- topic的partition数量少
- 所有logstash的配置文件都相同，使用同一个group消费所有的topic

### ■ 解决办法

- 提高topic的分区数量
- 对logstash进行分组，数据量较大的topic设置单独的一组logstash和消费组
- 每组logstash中总的消费线程数量和总的partition数量保持一致



## 2.4 集群扩容

### ■ 客户需求

780TB



4PB

云盘



本地盘

## 2.4 集群扩容

### ■ 出现问题



## 2.4 集群扩容

### 问题定位1

```
[c_log@VM_69_124-centos /data1/containers/1588756763000214332/er/health?pretty
{
  "cluster_name" : "es-10jh2nby",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 262,
  "number_of_data_nodes" : 259,
  "active_primary_shards" : 31455,
  "active_shards" : 62867,
  "relocating_shards" : 2648,
  "initializing_shards" : 26,
  "unassigned_shards" : 17,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 9,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 4397,
  "active_shards_percent_as_number" : 99.931648386584
}
```

**GET \_cluster/health**

## 2.4 集群扩容

### 问题定位2

Console Search Profiler Grok Debugger

```

22 GET _cat/thread_pool
23 GET _cat/tasks
24 GET _cat/pending_tasks
25 POST _tasks/jWWV8IjzQBGNpWTwha7Z5Q:2338227745/_cancel
26 GET _tasks/jWWV8IjzQBGNpWTwha7Z5Q:2338227745
27 POST _tasks/_cancel?nodes=TuPSc-ylTgSfgS-rjLS3PA&actions=*start_recovery
28 GET _cluster/health
29 PUT _cluster/settings
30 {
31   "transient": {
32     "cluster.routing.allocation.disk.watermark.low": "100mb"
33   }
34 }

```

```

1 12368430 1.2m URGENT shard-started StartedShardEntry{shardId
  [[bglog-gzlj-2020.06.15-01][35]], allocationId
  [YXQkiyjLSVuWvTRcMT0r4Q], primary term [1], message [after
  peer recovery]]}
2 12368431 1.2m URGENT shard-started StartedShardEntry{shardId
  [[bglog-gzlj-2020.06.15-01][35]], allocationId
  [235WST7VRGu7C3FCusmM3A], primary term [1], message [after
  peer recovery]]}
3 12368432 1.1m URGENT shard-started StartedShardEntry{shardId
  [[shrink-bglog-gzlj-2020.04.28-10][4]], allocationId
  [W6Lja9RaRAqOFRDPKikHaA], primary term [4], message [after
  peer recovery]]}
4 12368440 44.9s URGENT create-index [bglog-gzlj-2020.06.18-05],
  cause [auto(bulk api)]
5 12368433 59.5s URGENT create-index [bglog-gzlj-2020.06.18-05],
  cause [auto(bulk api)]
6 12368434 59.5s URGENT create-index [bglog-gzlj-2020.06.18-05],
  cause [auto(bulk api)]
7 12368456 44.6s URGENT create-index [bglog-gzlj-2020.06.18-05],
  cause [auto(bulk api)]
8 12368441 44.9s URGENT create-index [baloo-azli-2020.06.18-05].

```

**任务优先级：IMMEDIATE>URGENT>HIGH>NORMAL**



## 2.4 集群扩容

### 问题定位4

平均每秒写入次数（单位：次/秒）



整点时索引创建很慢，写入掉0



## 2.4 集群扩容

### ■ 经验总结

#### 慎重调整并发恢复分片数

分片数过多而需要进行数据搬迁时，并发恢复分片数不能设置的过高

#### 提前创建好索引

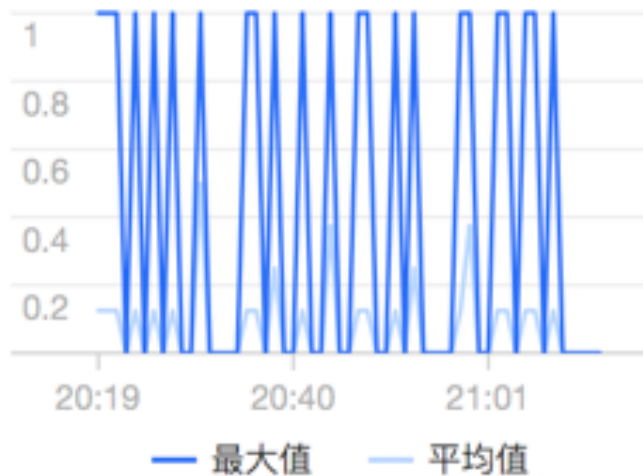
必须要进行长时间的数据迁移时，可以提前创建好索引以及索引的mapping避免写入失败

## 2.5 10w个分片

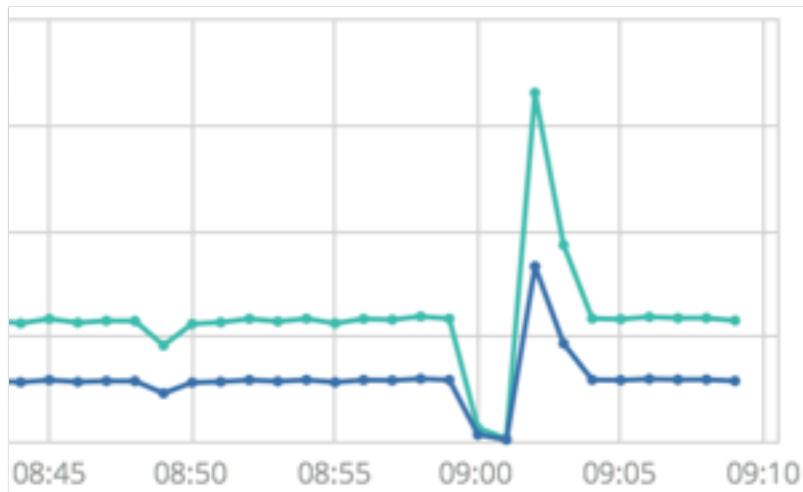
### 出现问题

节点内存使用率过高，频繁OOM

Old GC 次数 (次) ⓘ



影响索引创建



## 2.5 10w个分片

### 解决办法

#### 减小索引粒度

每个小时创建索引变更为每两个小时创建索引

#### 使用shrink

ILM中开启shrink, 老的索引由60分片shrink到10分片

#### 数据冷备份

老的索引先备份到COS, 然后副本调低为0

#### 关闭/冻结索引

定期关闭或者冻结更老的索引

## 2.5 10w个分片

### shrink实战

#### 1.修改索引属性

修改索引settings把索引old-index的分片全部移动到选择的节点上

#### 4.添加别名

删除old-index, 添加别名old-index, 指向shrink-index

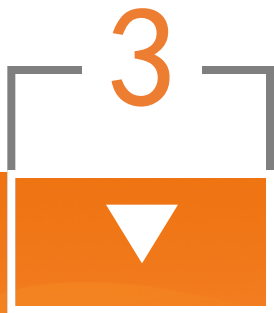


#### 2.确定目标分片数量

分片移动完毕后, 根据old-index索引的数据量确定目标分片数量, 按单分片50GB确定, 向上取整, 并且需要为old-index索引分片数量的因子

#### 3.执行shrink

等待新索引shrink-index状态为green



# Elasticsearch稳定性与成本优化

## 3.1 灵魂发问

**索引不断新建，如何保证一年内，集群总的分片数量保持在较稳定的水平？**

**数据量太大，如何降低成本？**



## 3.1 稳定性与成本优化

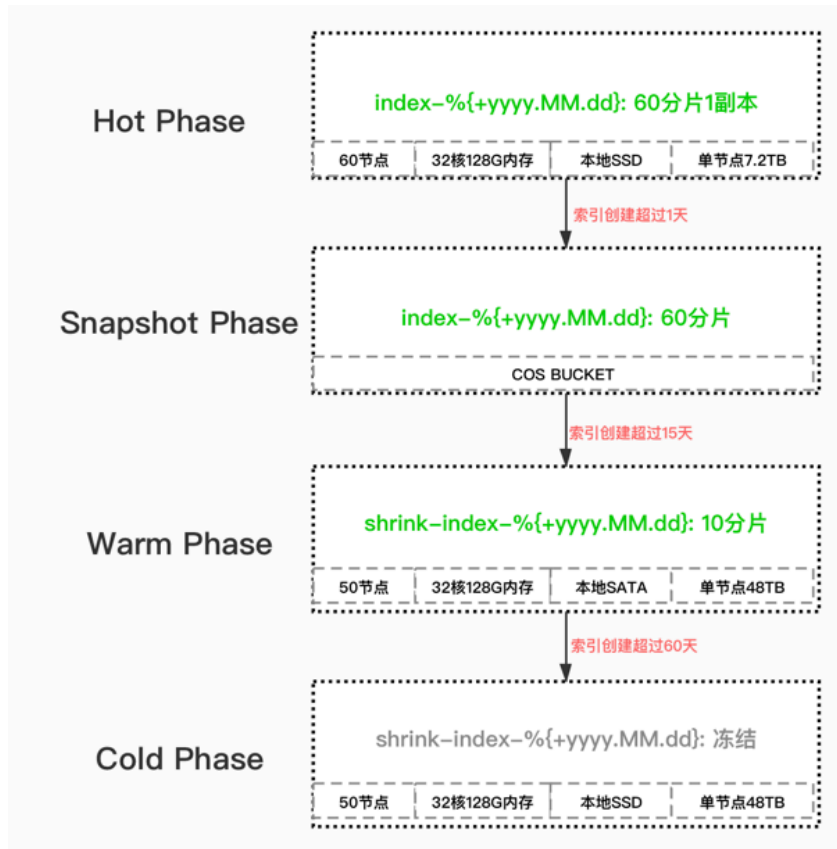
### ■ 优化办法

- 数据冷备到COS对象存储
- 老索引去副本
- 定期冻结旧索引

### ■ 实现效果

- 集群整体分片数量降低一半、总的存储量降低一半
- 老的索引仍然可查
- 极端情况下，磁盘故障引起部分索引无法本地恢复时，可从COS恢复

# 3.1 稳定性与成本优化



## 3.1 Searchable Snapshots





# Elasticsearch自动化运维

## 智能诊断

- 索引是否过多
- 分片大小是否合理
- 分片数量是否过多
- 是否出现热点

⋮

## 诊断报告

- 集群健康情况
- 节点健康情况
- 配置使用合理性
- 索引和分片合理性

⋮

## 改进措施

- 参数调优
- 横向、纵向扩容
- 重新规划索引
- 开启ILM降低成本

⋮



# Thanks

高效运维社区  
开放运维联盟

荣誉出品