58到家搜索服务化实践和演进

邢天宇@58到家

58到家
为一切美好

目 录

CONTENTS

背景

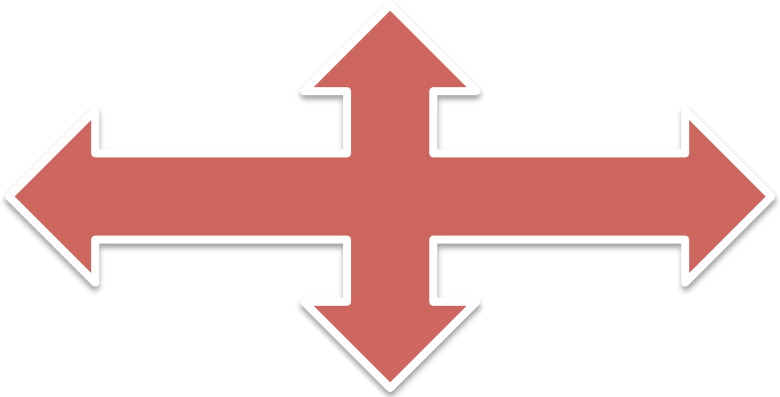
58到家
为一切美好

- 司机基于地理位置信息推单等业务
- 搜索推荐等业务
- 传统db分库分表查寻效率低
- 策略匹配类业务

必要性

降低学习成本

统一运维管理



必要的约束和定制

数据集中维护，成本核算

目标


提供基本增删改查，聚合排序功能。前缀，多值，分词匹配等
使用dsf对外提供接口的方式，与业务不相关
阈值控制，返回值，请求超时时间等
路由，searchType，preference等优化选项

目标

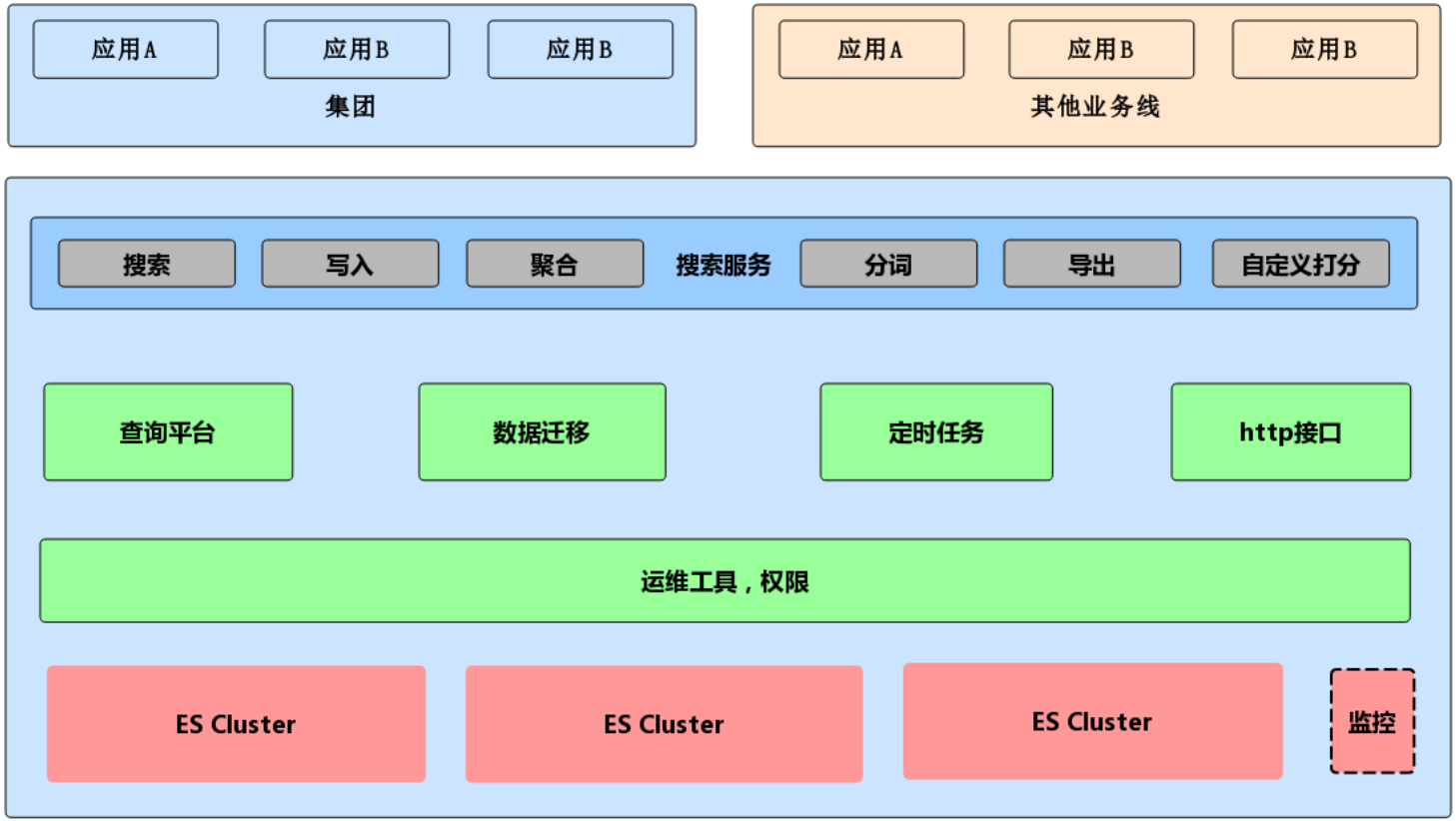
58到家
为一切美好

```
NQuery nQuery = new NQuery();
nQuery.index("alias_system")
        .types(new String[]{"ty_system"})
        .filter(new TermParam<>("cityId",city))
        .must(new MatchParam("name",keyword))
        .size(20);
SearchResults<AppIndex> results = SearchCloudProvider.search(nQuery);
```

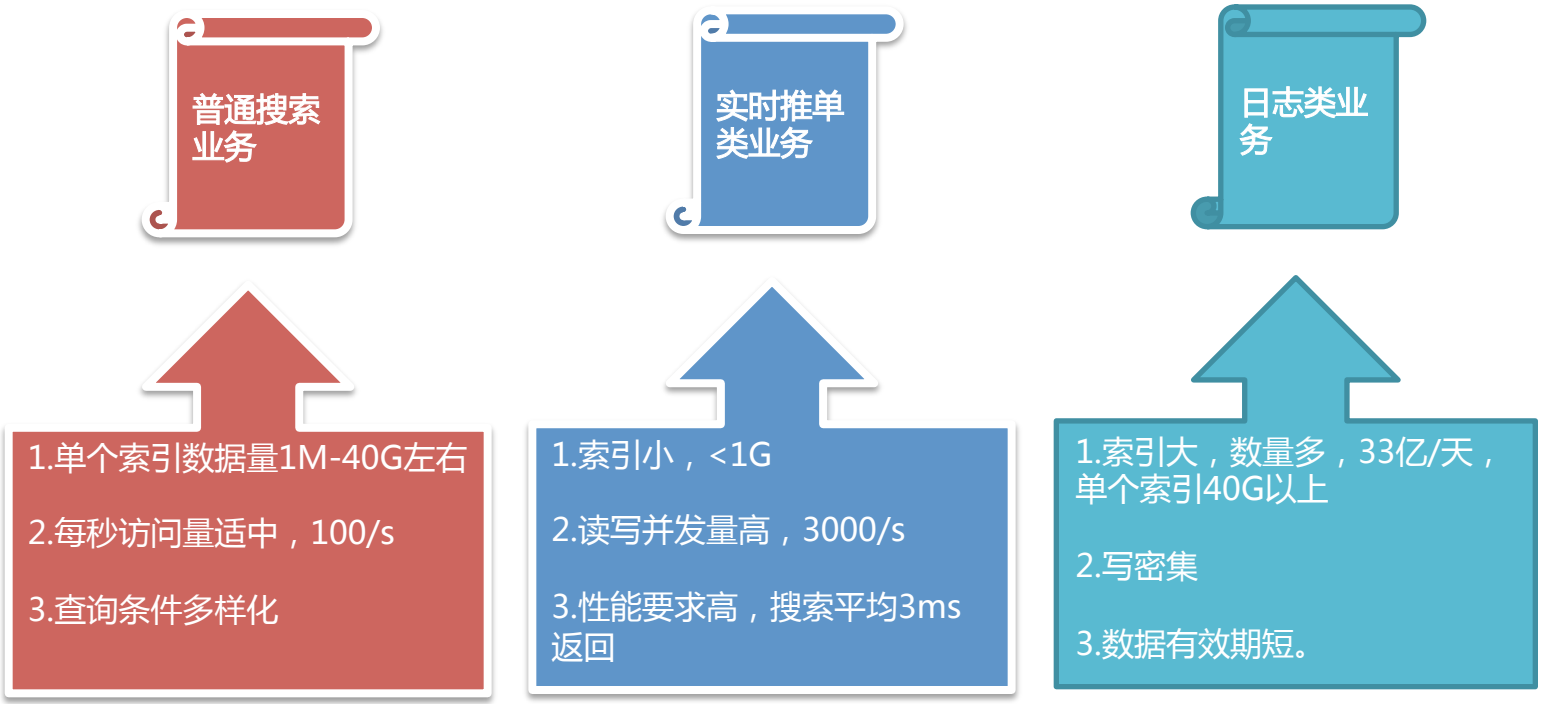
规范和约束

- 只存储查询条件，不保存长文本，不需要检索的字段不索引
- 业务索引尽量自定义id，数据敏感业务自备插入修改时间
- 取消dynamic mapping和auto index
- 一个索引一个type
- 控制单次搜索结果条数，总条数由es限制。控制请求超时时间
- 不提供脚本api，不提供update by query/delete by query，统一走id操作

整体架构


应用A
应用B
应用B
集团
应用A
应用B
应用B
其他业务线
搜索
写入
聚合
搜索服务
分词
导出
自定义打分
查询平台
数据迁移
定时任务
http接口
运维工具，权限
ES Cluster
ES Cluster
ES Cluster
监控

集群部署


普通搜索业务
1.单个索引数据量1M-40G左右
2.每秒访问量适中，100/s
3.查询条件多样化
实时推单类业务
1.索引小，<1G
2.读写并发量高，3000/s
3.性能要求高，搜索平均3ms返回
日志类业务
1.索引大，数量多，33亿/天，单个索引40G以上
2.写密集
3.数据有效期短。

集群运维工作

58到家
为一切美好

- 使用kopf进行集群管理，扩容下线等维护
- 低峰期定期对索引进行merge操作
- 提供业务线别名操作，数据清理直接重建新索引，切换别名，旧索引保留一天业务稳定后删除
- 使用模板记录业务索引字段
- nginx http authentication认证

58到家
为一切美好

目 录

CONTENTS

App搜索

58到家
为一切美好

App搜索

58到家
为一切美好

需求

- 根据服务标题进行分词检索
- 城市维度过滤
- 自营优先
- 提示词检索
- 评价，位置距离等排序

方案

•索引使用IK，搜索条件IK_SMART。防止匹配到不相关的名词，同时防止短标题的商家被漏掉。

•索引期间按城市做路由，提升搜索效率

•自营枚举值作为should过滤条件，自营boost高于其他

•使用pinyin4j将提示词词库的词构建成内存中的前缀树，用于提示词检索

•使用盒模型过滤商家，再使用评价和位置字段进行排序

App搜索

58到家
为一切美好

需求改版

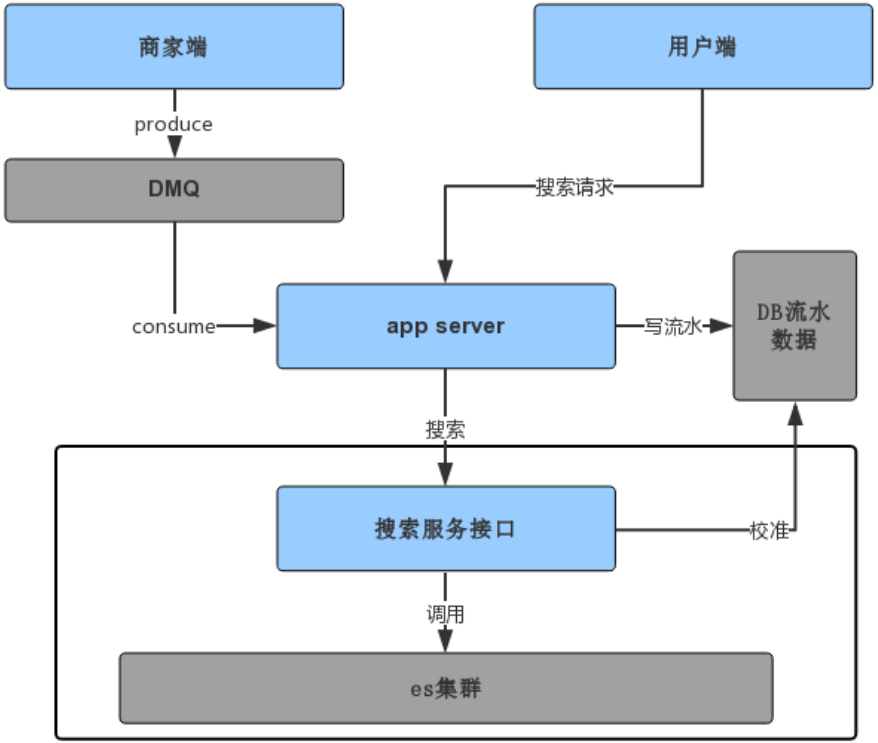
- app侧需要根据多种维度进行混排，不再按es搜索结果的相关度排序，但需要借助搜索结果的得分。同时希望命中相同关键字的文档得分相同

方案

•禁用标题字段的tf（"index_options": "docs"）和norms，忽略同一个关键字出现多次的情况。

•设置 dfs_query_and_fetch,保证多个分片上，存在相同关键字的文档计算得分相同。

App搜索


商家端
用户端
produce
DMQ
搜索请求
consume
app server
写流水
DB流水
数据
搜索
搜索服务接口
校准
调用
es集群

司机推单

58到家
为一切美好

需求

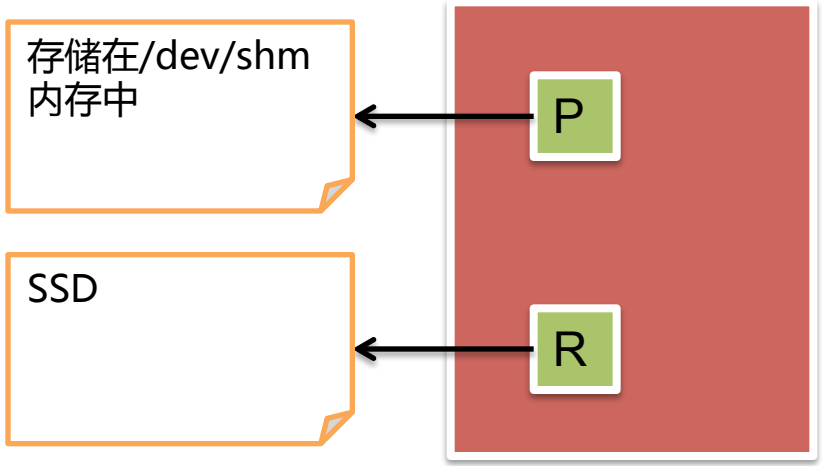
- 司机位置状态实时更新记录
- 用户下单，推单给附近的司机，司机捞取匹配策略
- 用户获取附近司机，查看司机信息

方案

- 司机端server使用mq推送司机实时状态信息，后端消费司机信息，批量写入
- 使用盒模型过滤司机，每次20个，返回结果不足20个则扩大搜索半径
- 同上，盒模型过滤，使用id查看司机信息

司机推单

- 一主一副，主分片使用内存，副本放在固态硬盘上
- 搜索优先使用主分片
- 盒模型过滤司机



调用链日志

traceId : 95955b22-307b-4ac3-92da-f60051906607 查询 返回

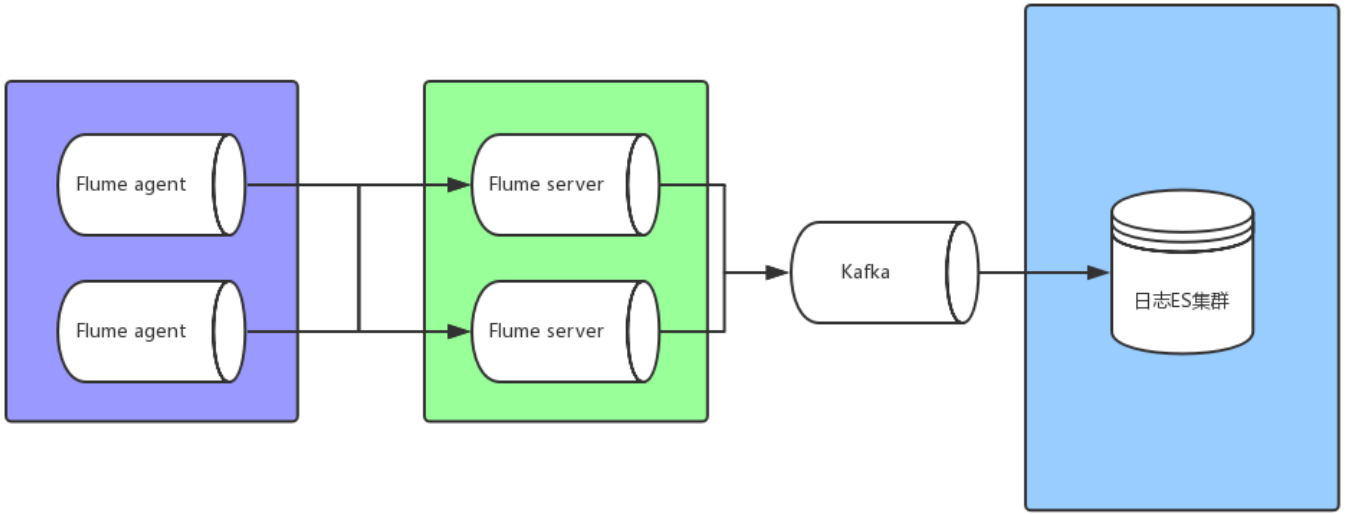
开始执行时间	耗时	起始IP	请求url
2018-08-30 19:04:17.691	4000ms	[redacted]	lbs.MileageServiceImpl.uploadLocatio.

集群名称	类型	状态	IP	接口	参数	业务Id
⊟ lbs	dsf(c)	ERROR				

共 1 条数据

```
com.daojia.spat.dsf.protocol.exception.TimeoutException: ServiceName:[lbs],ServiceIP:[redacted],Receive data timeout or error!timeout:4000ms
  at com.daojia.spat.dsf.client.communication.socket.CSocket.receiveWithinMillSeconds(CSocket.java:193)
  at com.daojia.spat.dsf.client.communication.socket.CSocket.receive(CSocket.java:183)
  at com.daojia.spat.dsf.client.loadbalance.Server.request(Server.java:316)
  at com.daojia.spat.dsf.client.proxy.ServiceProxy.invoke(ServiceProxy.java:217)
  at com.daojia.spat.dsf.client.proxy.builder.MethodCaller.doMethodCall(MethodCaller.java:68)
  at com.daojia.spat.dsf.client.proxy.builder.ProxyStandard.invoke(ProxyStandard.java:107)
  at com.sun.proxy.$Proxy46.uploadLocations(Unknown Source)
  at com.bj58.daojia.suyun.gps.store.GpsPointStore$GpsPointStoreRun.run(GpsPointStore.java:58)
  at java.lang.Thread.run(Thread.java:745)
```

调用链日志


Flume agent
Flume agent
Flume server
Flume server
Kafka
日志ES集群

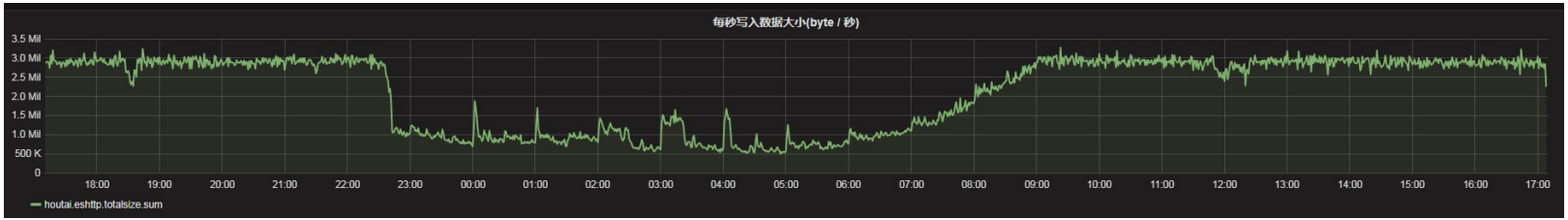
调用链日志

58到家
为一切美好

- 一个节点一个主分片，0副本
- 设置refresh_interval = -1
- 批量写入，控制单批写入字节数。
- 固态硬盘存储
- 定期归档，删除过期索引，merge上一个时间点的索引

调用链日志

- 高峰期每秒5w条日志
- 每天平均产出33亿条，数据大小720G
- 日志消费侧打印埋点，写入opentsdb，通过grafana实时监控日志写入情况



58到家
为一切美好

目 录

CONTENTS

调用链日志写入慢

58到家
为一切美好

描述和排查：

调用链日志集群自交接后半年内，数据量自然增长至原来的3倍，集群出现明显的负载过高，导致日志写入性能急剧下降，kafka出现大量堆积，日志滞留时间过长。

经日志排查，发现日志写入每条高于1.5ms时kafka出现堆积现象。经jstat 观察gc状况正常（老年代gc平均每1小时一次，内存晋升率较低），排除gc影响；dstat 观cpu load在堆积时间段较高，cpu的us占比较高，而iowait较低，集群使用固态硬盘，因此排除磁盘瓶颈；通过jstack和es 热点线程api发现，cpu占用较高的线程集中在bulk和merge上，因此将问题集中在es的批量插入和merge上

解决方案：

索引写入时会伴随着id校验，请求体解析，分词等操作，都会带来一定的cpu开销。原先的索引结构中存在部分多余字段，无需进行分词，取消后可以减轻cpu压力。

取消索引中的_all字段减小索引，并减少分词带来的开销；使用es自动生成id，省去id检查步骤。调整translog合并时间，半小时一次，防止过多merge任务导致cpu开销过大。

解决后日志集群吞吐量趋于稳定，系统cpu load 大幅度下降，目前能够承载优化前至少3倍的日志写入量。

调用链日志写入慢


```
"number_of_replicas": "0",
"translog": { -
  "flush_threshold_ops": "10000000",
  "flush_threshold_size": "5120m",
  "flush_threshold_period": "30m",
  "sync_interval": "30s",
  "durability": "async"
},
"search": { -
  "slowlog": { -
    "threshold": { -
      "fetch": { -
        "warn": "1s"
      },
      "query": { -
        "warn": "3s"
      }
    }
  }
},
"indexing": { -
  "slowlog": { -
    "threshold": { -
      "index": { -
        "warn": "500ms"
      }
    }
  }
},
"merge": { -
  "policy": { -
    "max_merged_segment": "1g",
    "segments_per_tier": "20"
  }
},
"codec": "best_compression",
"uuid": "nyFHmFc3RViw9L-sUES5AQ",
"number_of_shards": "4",
"refresh_interval": "-1",
```

司机推单集群cpu过高

描述：

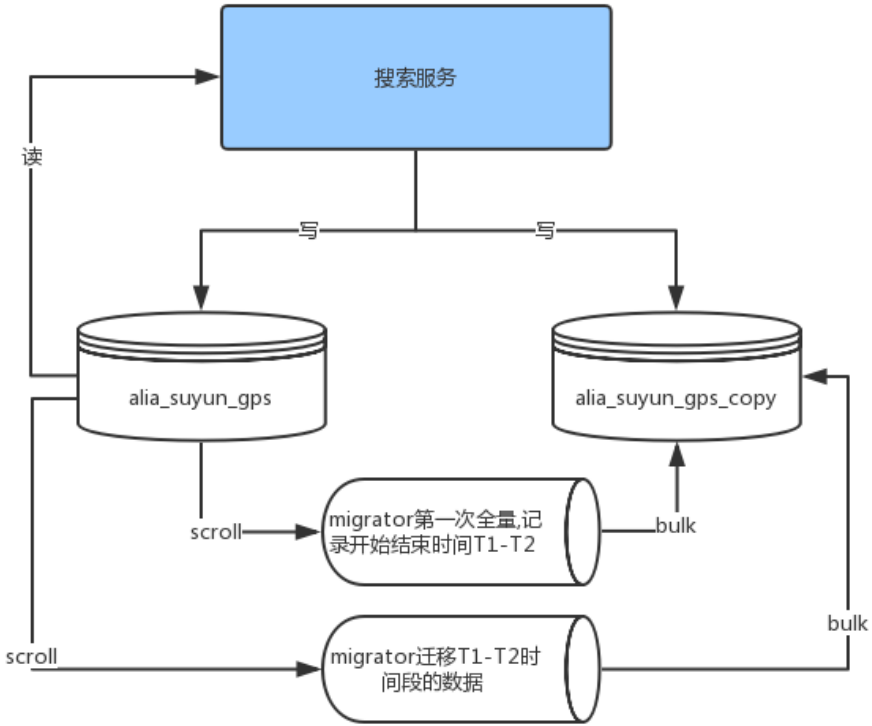
速运gps业务索引随着场景变化，写入量逐渐增多，集群cpu load变高，原来单个主分片写入出现瓶颈（耗时变长）。

方案：

重建索引，主分片改为2个，分别分布在两台机器，达到负载均衡效果，数据需要迁移，并对上层业务无感知

成果：

集群节点load从8.2下降到5.6，低于核数的0.7，趋于平稳。服务平均延时从10ms降低到7ms，查询耗时稳定在2-5ms



业务集群二次划分

58到家
为一切美好

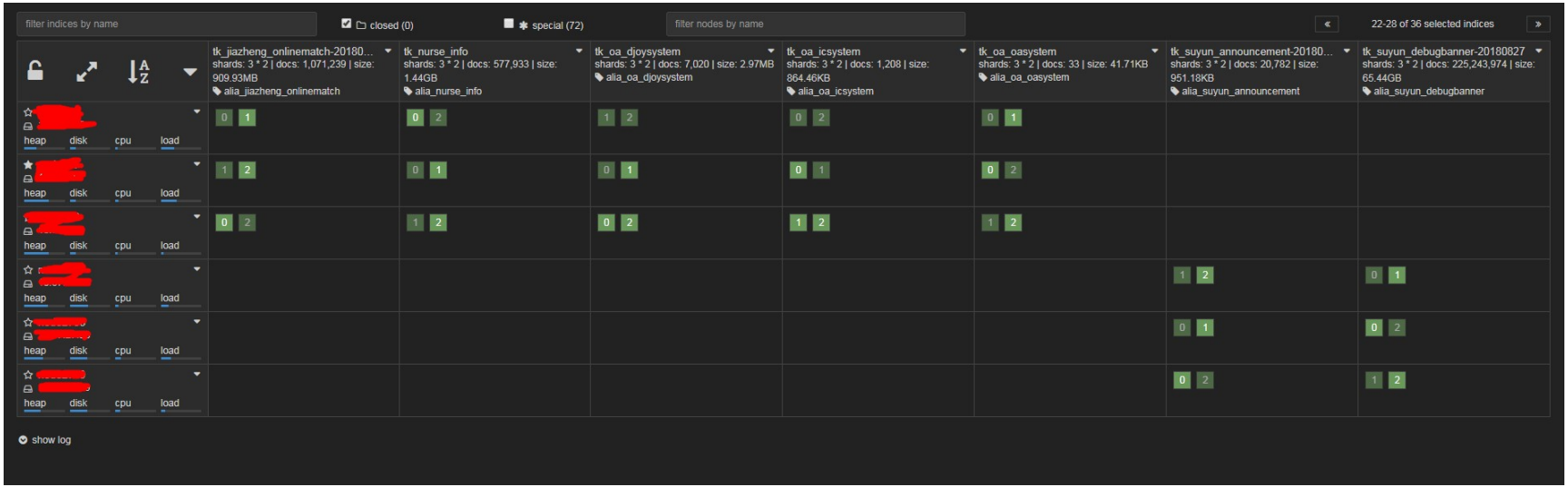
描述：

早期搜索业务线都在同一个业务集群内，有的业务不恰当使用搜索服务导致其他业务的索引受牵连（index/search队列在同一个集群上堆积等等）

方案：

按业务线做二次规划，索引设置index.routing.allocation.zone将分片分布到指定的机器，做到统一集群内请求互不干扰。同时服务层也做隔离，确保请求完全分隔开

业务集群二次划分


filter indices by name
closed (0)
special (72)
filter nodes by name
22-28 of 36 selected indices
tk_jiazheng_onlinematch-20180... shards: 3 * 2 | docs: 1,071,239 | size: 909.93MB alia_jiazheng_onlinematch
tk_nurse_info shards: 3 * 2 | docs: 577,933 | size: 1.44GB alia_nurse_info
tk_oa_djoysystem shards: 3 * 2 | docs: 7,020 | size: 2.97MB alia_oa_djoysystem
tk_oa_icsystem shards: 3 * 2 | docs: 1,208 | size: 864.46KB alia_oa_icsystem
tk_oa_oasystem shards: 3 * 2 | docs: 33 | size: 41.71KB alia_oa_oasystem
tk_suyun_announcement-20180... shards: 3 * 2 | docs: 20,782 | size: 951.18KB alia_suyun_announcement
tk_suyun_debugbanner-20180827 shards: 3 * 2 | docs: 225,243,974 | size: 65.44GB alia_suyun_debugbanner
heap disk cpu load
show log

总结

58到家
为一切美好

服务化

对外提供rpc接口和orm like sdk。接口适当约束

集群部署

根据业务场景进行集群规划，相同业务分布到统一集群，易于管理，问题统一解决。

未来规划

实现集群监控，保存cpu load,内存,jvm gc等使用情况，以及es 各种请求队列，各个索引段，查询缓存等监控和阈值报警

58到家
为一切美好

架构师之路

Q&A

谢谢 THANKS

elastic
中文社区

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