Elasticsearch在字节跳动的内核优化实践

字节跳动资深技术专家 / 张超

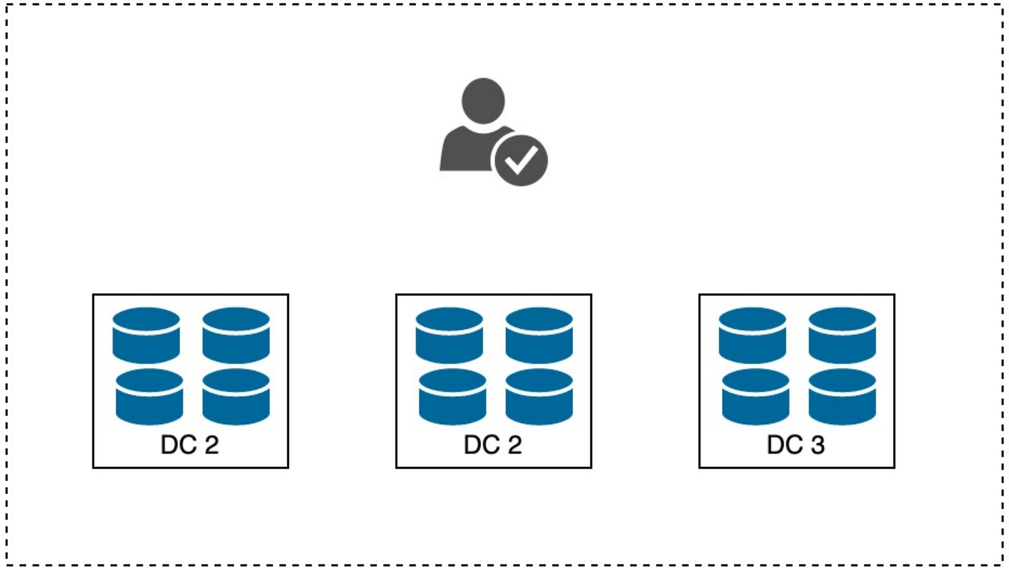
ByteDance字节跳动

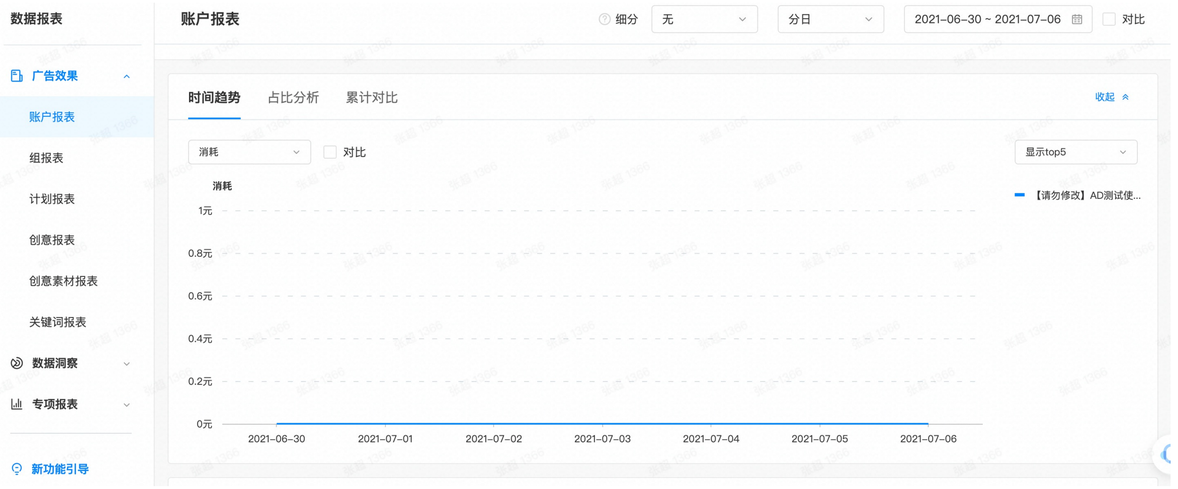
目录

- 广告业务特点
- 遇到的主要挑战
- 对内核的持续改进
- 未来展望

广告业务的特点

- 极高的可用性
- 高QPS，低延迟





广告业务的特点

- 查询模式比较固定
- terms in 数量很多
- 大多没有分词
- 没有嵌套聚合

```
"query": {
    "bool": {
      "filter": [
        { "terms": { "dim_id": [...] }},
        { "range": { "time": {"lt": "","gte":""}}} ]
  }},
  "aggs": {
    "time": {
      "date_histogram": { "field": "time"},
      "aggs": {
        "dim_id": {
          "terms": { "field": "dim_id","size": 100000 },
          “aggs”:  { "f1": { "sum": { "field": "f1"}},
            ...
```

主要问题-稳定性

问题1: OOM

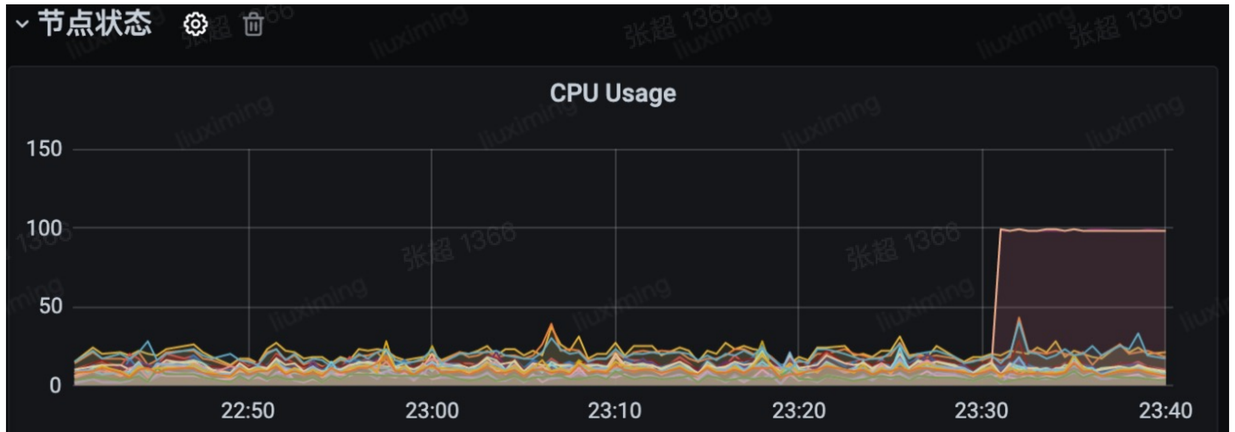
被写入或者查询打挂，大部分是查询，对于查询的熔断，7.12 的效果已经比较理想，或者配置 100g 内存

问题2: breaker导致CPU跑满

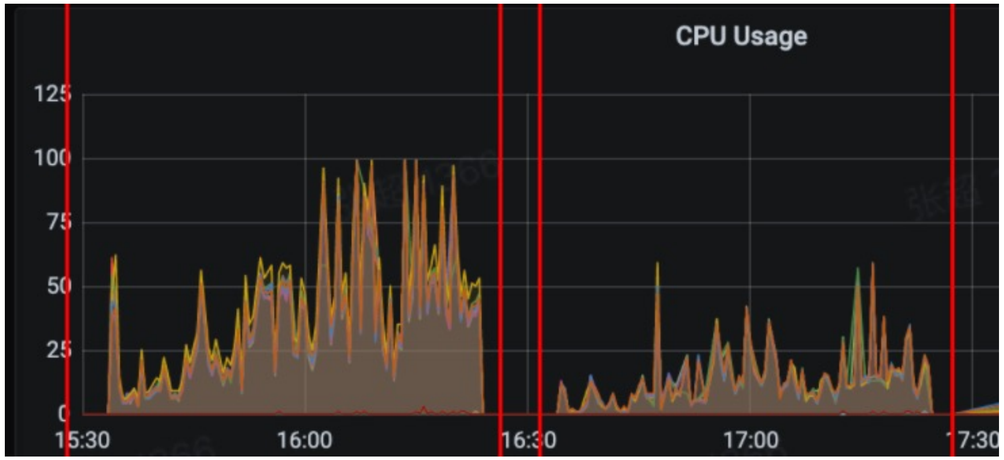
对于每个bucket都检查breaker，高并发下CAS占用CPU严重。

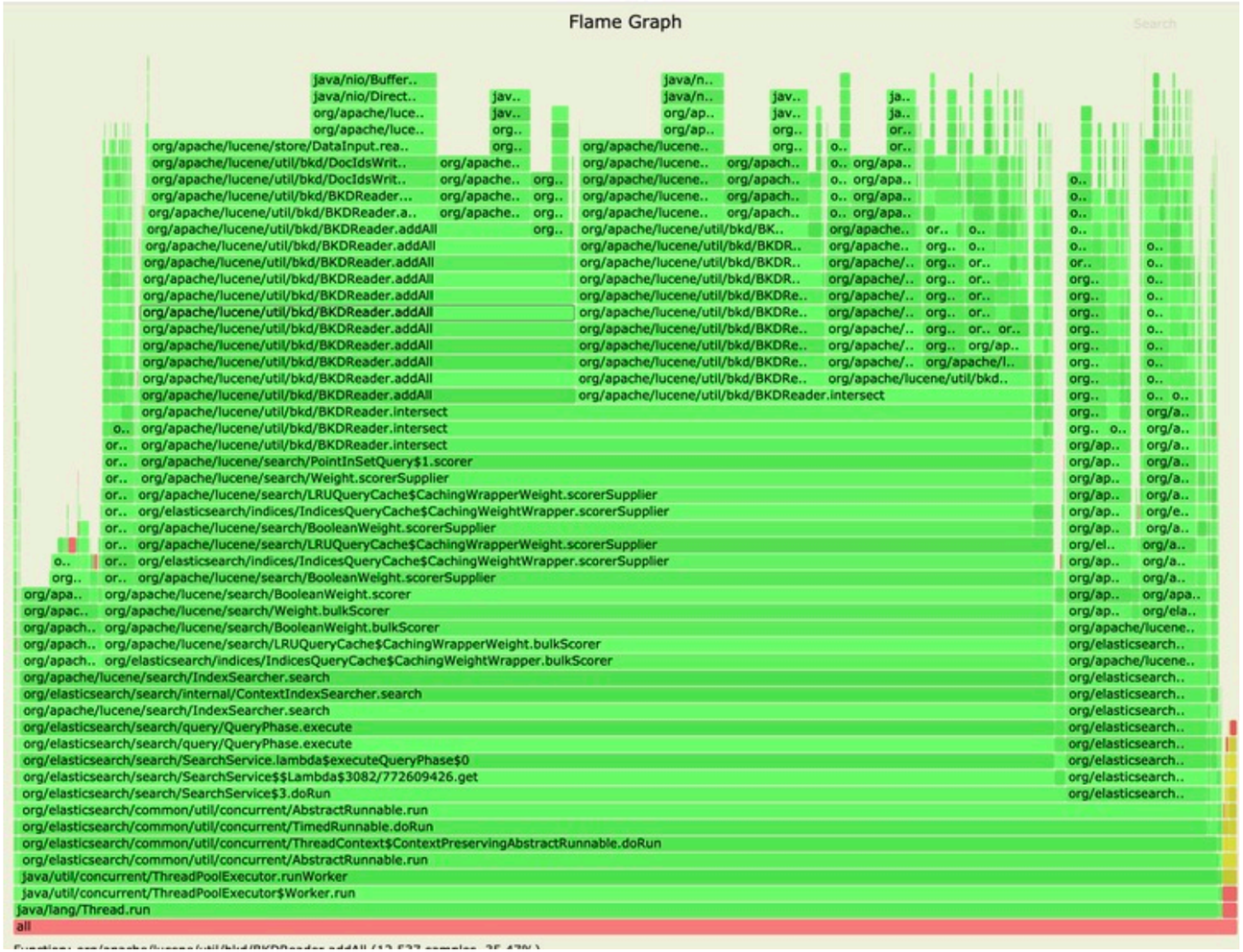
PreallocatebreakerService的方式代替现有的addEstimateBytesAndMaybeBreak方法

Issue:58647



主要问题-keyword/long数据类型不合理





在long类型的低基数字段上执行term/term查询

- 需要构建巨大的bitset
- 计算cost代价大

引擎优化-新的数据类型： longkey

像id、tags这种字段需要是快速的点查，高效的聚合

Keyword：

- 【优】索引结构为FST+跳表，点查速度很快。 集合运算效率高
- 【劣】聚合需要构建全局序数

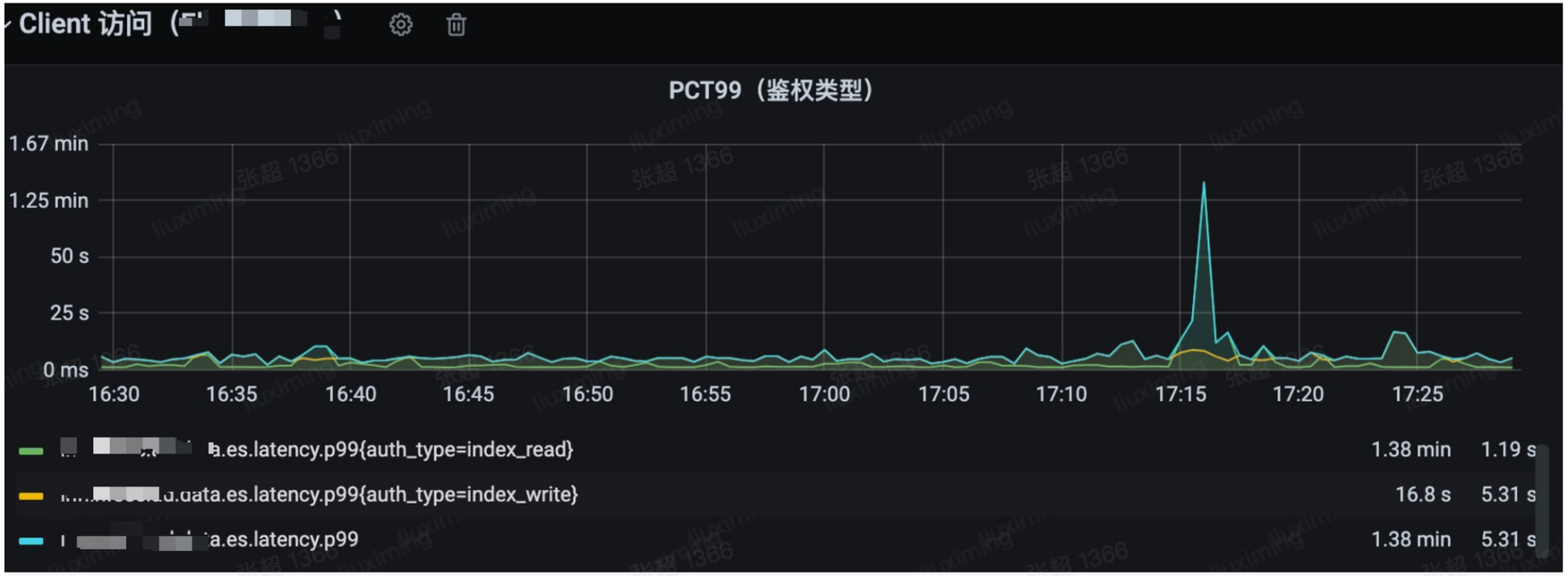
Long：

- 【优】聚合性能高，无需构建全局序数
- 【劣】集合运算需要构建bitset。

解决方案：定义新的数据类型 Identifier。 Identifier 的索引使用 FST+跳表，列式存储使用 SortedNumericDocvalues。

- 高效的点查索引 + 高效的列式存储，强强联合。

主要问题-慢查询



当系统的 CPU 跑满，或者有大查询导致节点 GC 时，集群整体表现为 pct99 升高。这种情况下，普通的小查询因为集群资源不足也会 pct99 升高，因此很难从 access log 或 slow log层面找到是哪个或者哪些查询导致了集群负载过高， 从 log 中单独拿出查询延迟很长的语句去跑，都能很快返回

引擎优化-慢查询定位

集群负载包括 cpu 和 jvm 层面，实际上也包括 io 层，但是我们基本都是 ssd 的盘，io 层问题不明显，因此我们希望让引擎多输出一些信息，把每个query 占用的资源打出来，帮助我们找到最耗费资源的 query，以便后续的优化工作。

getThreadAllocatedBytes 返回一个递增的近似值，这样我们可以聚合调用前后打点来捕获这个过程中分配了多少内存

```
/Documents/DATA/LINUX/zhangchao-so/byte_work ❯ curl ":9200/_nodes/hot_threads?type=mem&interval=1s"
:: {node-1}{piPTqSeQQOiTYEi6BLCUKw}{1Cu0YiYTSOeXuxuFpC_5MA}{127.0.0.1}{127.0.0.1:9300}{dilm}{ml.machine_memory=34359738368, xpa
   Hot threads at 2021-03-10T10:21:53.958Z, interval=1s, busiestThreads=3, ignoreIdleThreads=true:

    0.8% (41720848 out of 5368709120) mem usage by thread 'elasticsearch[node-1][search][T#25]'
     10/10 snapshots sharing following 18 elements
       java.base@13.0.2/java.util.ArrayList$ListItr.add(ArrayList.java:1064)
       app//org.elasticsearch.search.aggregations.bucket.histogram.InternalDateHistogram.addEmptyBuckets(InternalDateHistogram.j
       app//org.elasticsearch.search.aggregations.bucket.histogram.InternalDateHistogram.reduce(InternalDateHistogram.java:455)
       app//org.elasticsearch.search.aggregations.InternalAggregations.reduce(InternalAggregations.java:168)
       app//org.elasticsearch.search.aggregations.InternalAggregations.topLevelReduce(InternalAggregations.java:116)
       app//org.elasticsearch.action.search.SearchPhaseController.reducedQueryPhase(SearchPhaseController.java:490)
       app//org.elasticsearch.action.search.SearchPhaseController.reducedQueryPhase(SearchPhaseController.java:404)
       app//org.elasticsearch.action.search.SearchPhaseController$1.reduce(SearchPhaseController.java:725)
```

引擎优化-慢查询定位

问题现场：对实时请求观测：

tasks?pretty&actions=indices:data/read/search*

&detailed&group_by=parents”

事后分析：留存慢日志追踪

PR https://github.com/elastic/elasticsearch/pull/72850

```
y allocated in query phase,alloc_bytes:4830416,[ad-reindex-split][2],source[{"size":1
y allocated in query phase,alloc_bytes:4965056,[ad-reindex-split][0],source[{"size":1
y allocated in query phase,alloc_bytes:4941416,[ad-reindex-split][1],source[{"size":1
y allocated in query phase,alloc_bytes:5585888,[ad-reindex-split][4],source[{"size":1
y allocated in query phase,alloc_bytes:4927408,[ad-reindex-split][3],source[{"size":1
y allocated in query phase,alloc_bytes:4533088,[ad-reindex-split][5],source[{"size":1
y allocated in query phase,alloc_bytes:4518216,[ad-reindex-split][6],source[{"size":1
y allocated in query phase,alloc_bytes:4537512,[ad-reindex-split][7],source[{"size":1
y allocated in coordinator,alloc_bytes:522117176,reduce[361823088],consumeResult[1602
y allocated in fetch phase,alloc_bytes:724160,[ad-reindex-split][0],source[{"size":1,
```

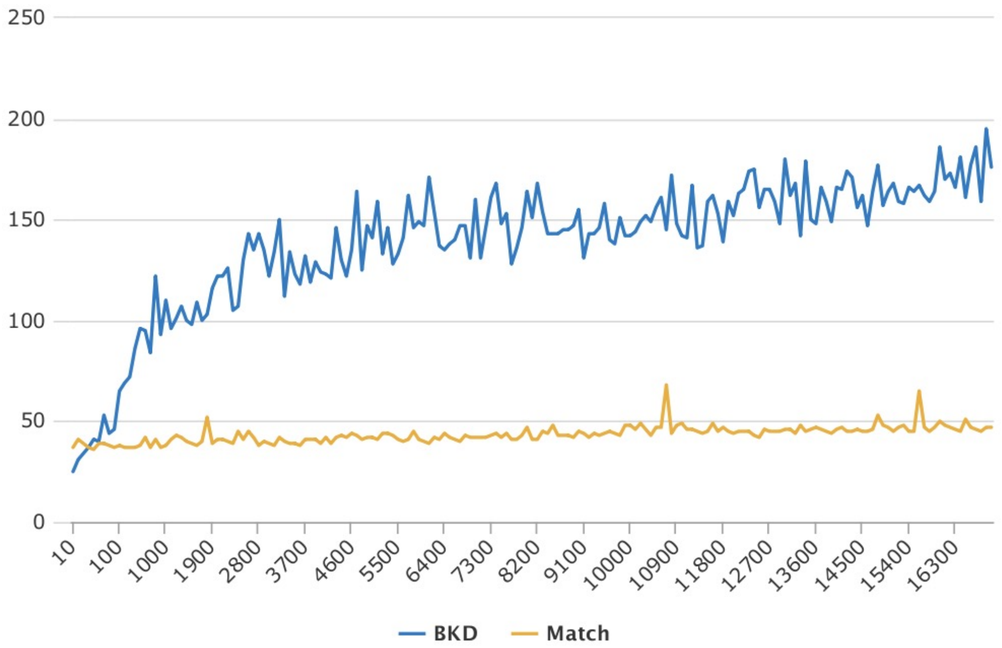
查询优化-TermsQuery

target	baseline	candidate
1 index (8 shards)	83ms	21ms
30 indices (240 shards)	2935ms	231ms

优化1：避免冗余序列化。将TermsQuery的序列化操作前置到构造函数中，外部调用序列化时则只需要做简单的byte copy。

优化2：实现match的查询方式并基于leadcost和terms数量实现查询路径自动优化。

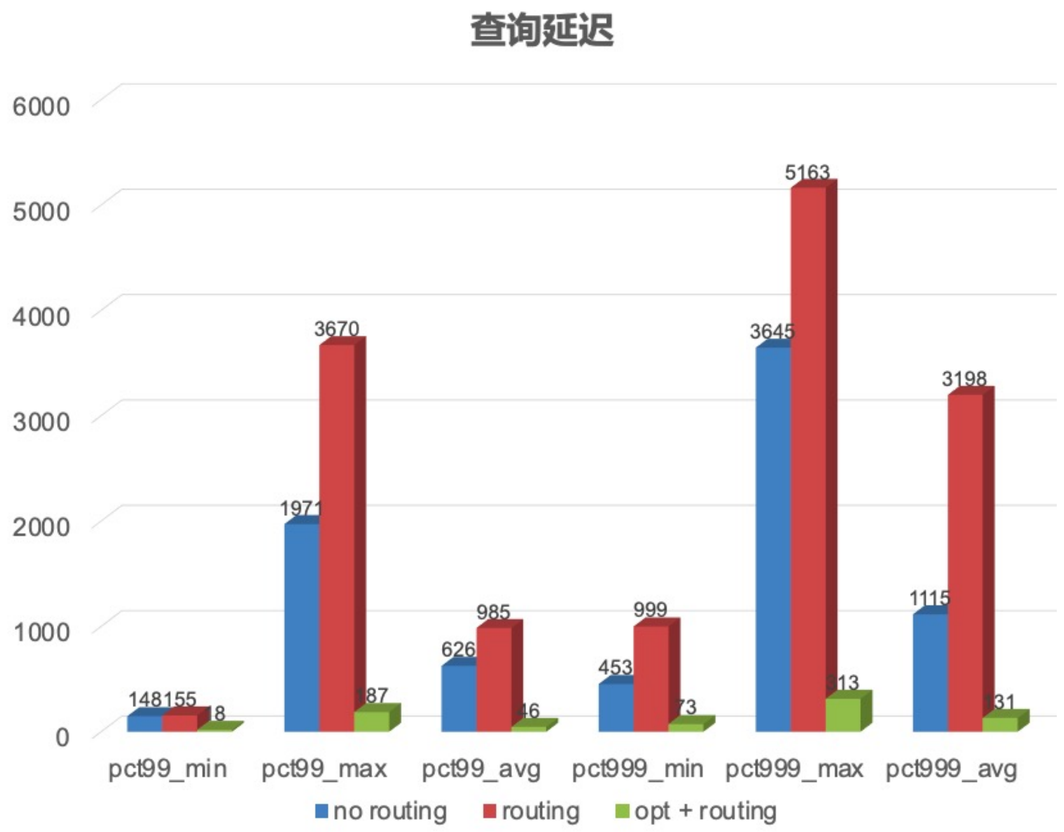
```
[
    {
    "a_id": ['123'],
    "b_id": [1,2,3 ....... 10000001]可能有10w个id
    }
]
```



PR：https://github.com/elastic/elasticsearch/pull/67223

查询优化-GroupbyAgg

```
"query": {
    "bool": {
      "filter": [
        { "terms": { "dim_id": [...] }},
        { "range": { "time": {"lt": "","gte":""}}} ]
  }},
  "aggs": {
    "time": {
      "date_histogram": { "field": "time"},
      "aggs": {
        "dim_id": {
          "terms": { "field": "dim_id","size": 100000 },
          “aggs”:   { "f1": { "sum": { "field": "f1"}},
```



- 千万分桶5s内返回
- 单集群12W QPS（40node,hdd）

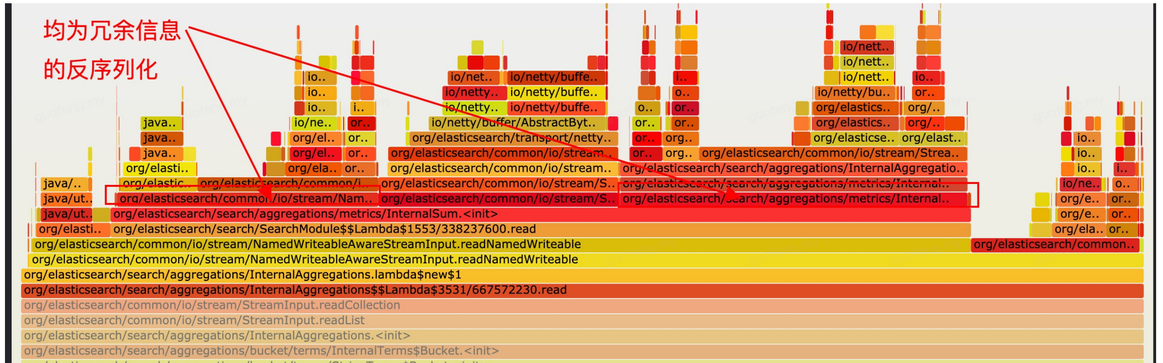
查询优化-GroupbyAgg

```
"buckets": [
  {
    "key": "electronic",
    "doc_count": 6
  },
  {
    "key": "rock",
    "doc_count": 3
  },
  {
    "key": "jazz",
    "doc_count": 2
  }
```

优化1：削减冗余报文

以百万分桶查询为例，序列化json加网络的时间达到近 10s。Groupby Agg 将维度和指标完全铺平，以降低报文序列化和反序列化的开销.平铺结构会损失一定的灵活性，例如不支持同一层包含多个Bucket Aggregation。但平铺结构也会带来一些优势：如指标排序等。

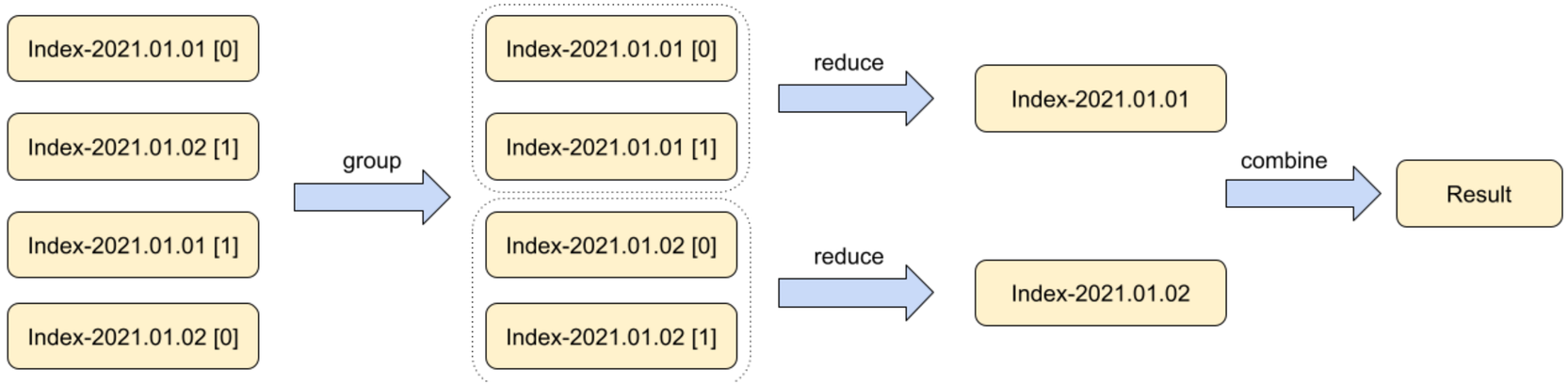
	ES took	E2E（took + to json + network）
Baseline	32350 ms	41933 ms
Candidate	22315 ms	25732 ms



查询优化-GroupbyAgg

优化2：基于时序索引特征分区

- 业务数据以时序性天级索引存储，单索引内只包含一天数据。而聚合的第一个维度是时间维度，即可以确保不同索引不会产生相同维度组合。
- 结合Index Routing，保证单条查询在单索引内只会查询一个分片，结合此优化，协调节点甚至不再需要做任何的reduce操作，直接将数据节点返回的Bucket Combine即可。



查询优化

优化3：实现指标聚合模块

- Long替换double，解决精度丢失，运算慢，序列化反序列化慢
- Count 优化

特定场景优化

- 查询超时自动取消 PR #71714
- 部分场景使用倒排聚合，性能提升数量级

写入优化/预处理

- Esloader外部构建离线索引，主要用于reindex的场景
- AdaptiveForceMerge，并发merge，根据负载动态限速
- 写入过程定向路由，写入速度提升30%
- 预处理生成HLL用于基数计算

在做的事情+未来规划

- 计算过程使用arrow替代json，千万分桶秒内返回
- 支持Join，支持衍生字段排序
- 单集群支持更大规模（moveshard定期执行，reroute提前计算等）

我们是谁？

字节跳动商业化数据平台。商业化数据平台作为商业化数据中心，负责所有广告业务在线和离线数据处理、数据服务及数据可视化分析及异常归因服务，国际化数据集市等。数据平台致力于提供高质量的数据，并基于海量数据提供平台化的数据服务于数据应用。

我们是做什么的？

负责商业化系统**OLAP**引擎的研发，包括不限于**Elasticsearch**、**Druid**、**Clickhouse**、**Doris**等。我们源于开源，但不局限于开源，结合商业化业务场景，突破现有引擎瓶颈，探索最佳**OLAP**解决方案，解决系统性能和稳定性问题，在**Elasticsearch**，**Apache Lucene**，**Apache Druid**等知名开源项目贡献了 **10+** 高价值**PR**。

加入我们

字节跳动技术公众号

THANKS.

ByteDance字节跳动