elastic 中国开发者大会 2023

Elasticsearch在vivo搜索中台的实践

王文谦

vivo 互联网搜索团队

分享嘉宾

王文谦

vivo互联网技术经理
搜索中台负责人

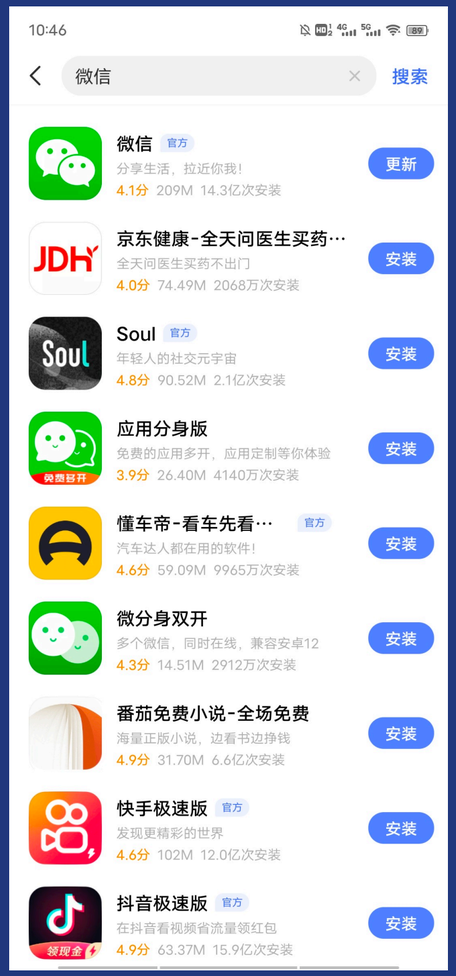
一、vivo搜索架构演进

二、ElasticSearch在vivo搜索中台的实践

三、搜索中台业务应用总结

一、vivo搜索架构演进

vivo 搜索业务介绍

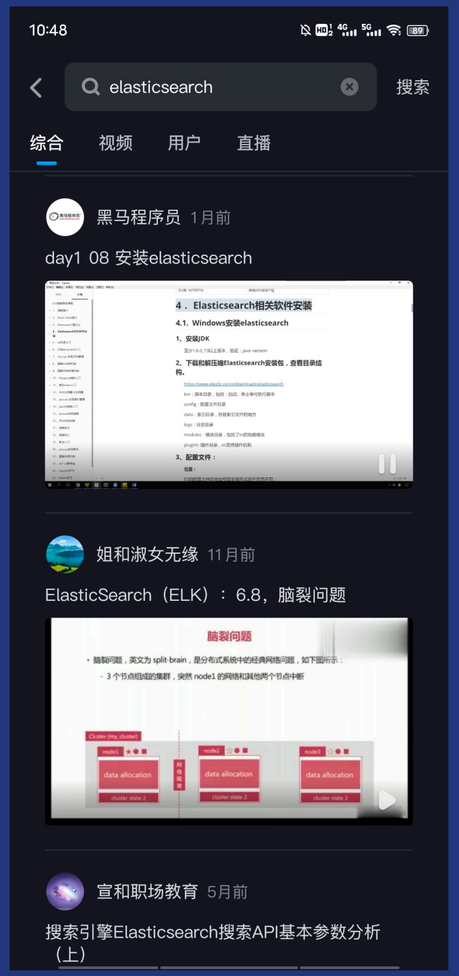


应用搜索

亿级PV

游戏搜索

千万级PV



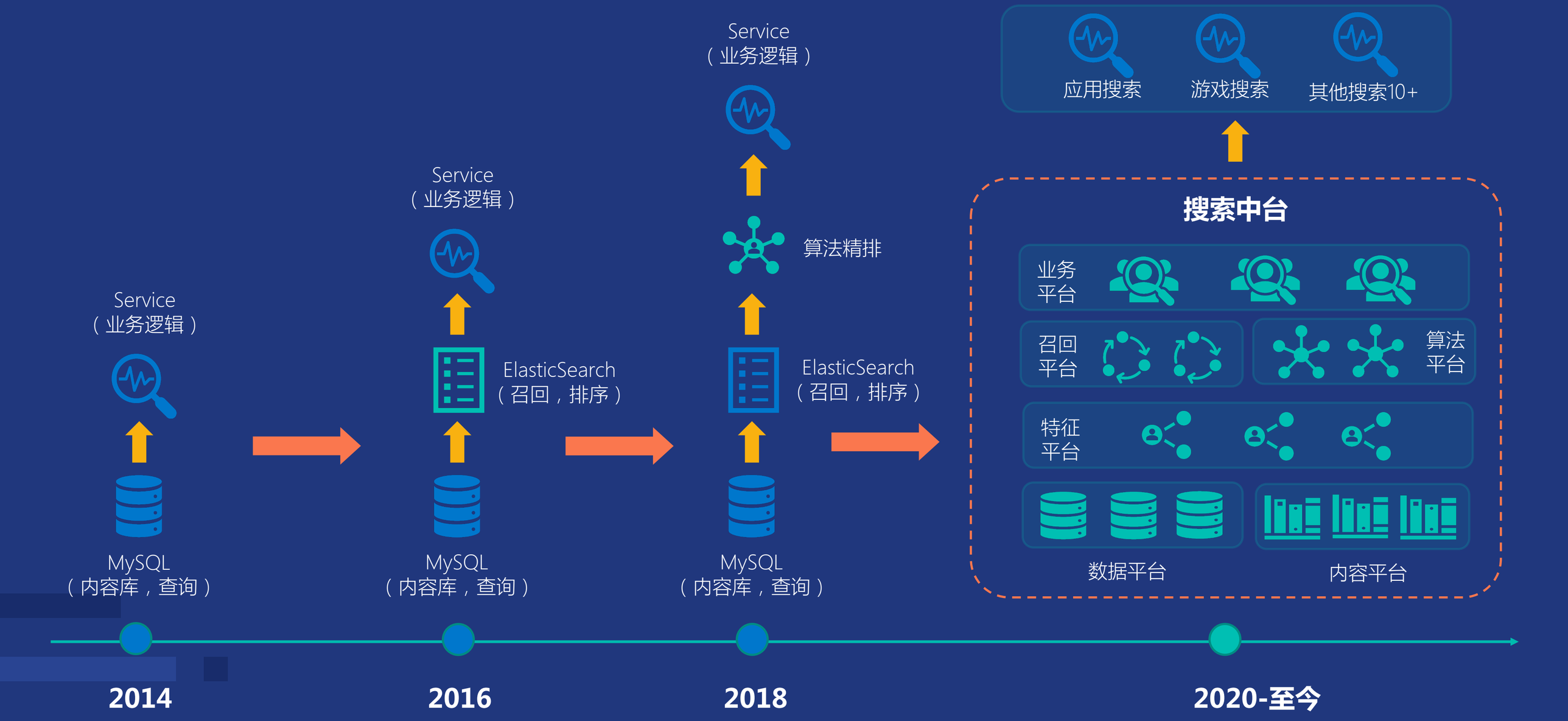
视频搜索

百万级PV

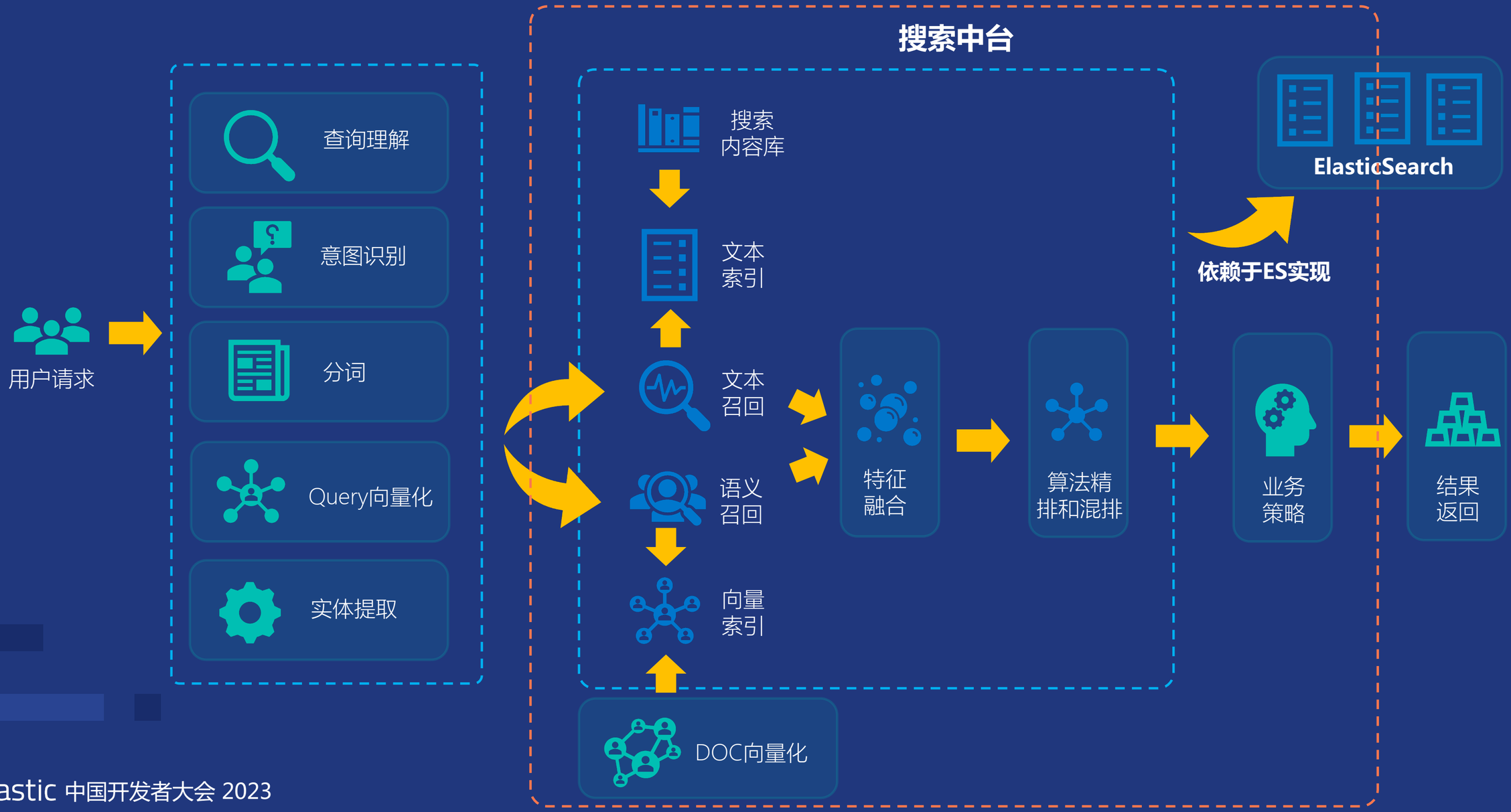
官网搜索

百万级PV

vivo 搜索架构演进


Service
（业务逻辑）
MySQL
（内容库，查询）
Service
（业务逻辑）
ElasticSearch
（召回，排序）
MySQL
（内容库，查询）
Service
（业务逻辑）
算法精排
ElasticSearch
（召回，排序）
MySQL
（内容库，查询）
应用搜索
游戏搜索
其他搜索10+
搜索中台
业务平台
召回平台
算法平台
特征平台
数据平台
内容平台
2014
2016
2018
2020-至今

vivo 搜索中台业务流程

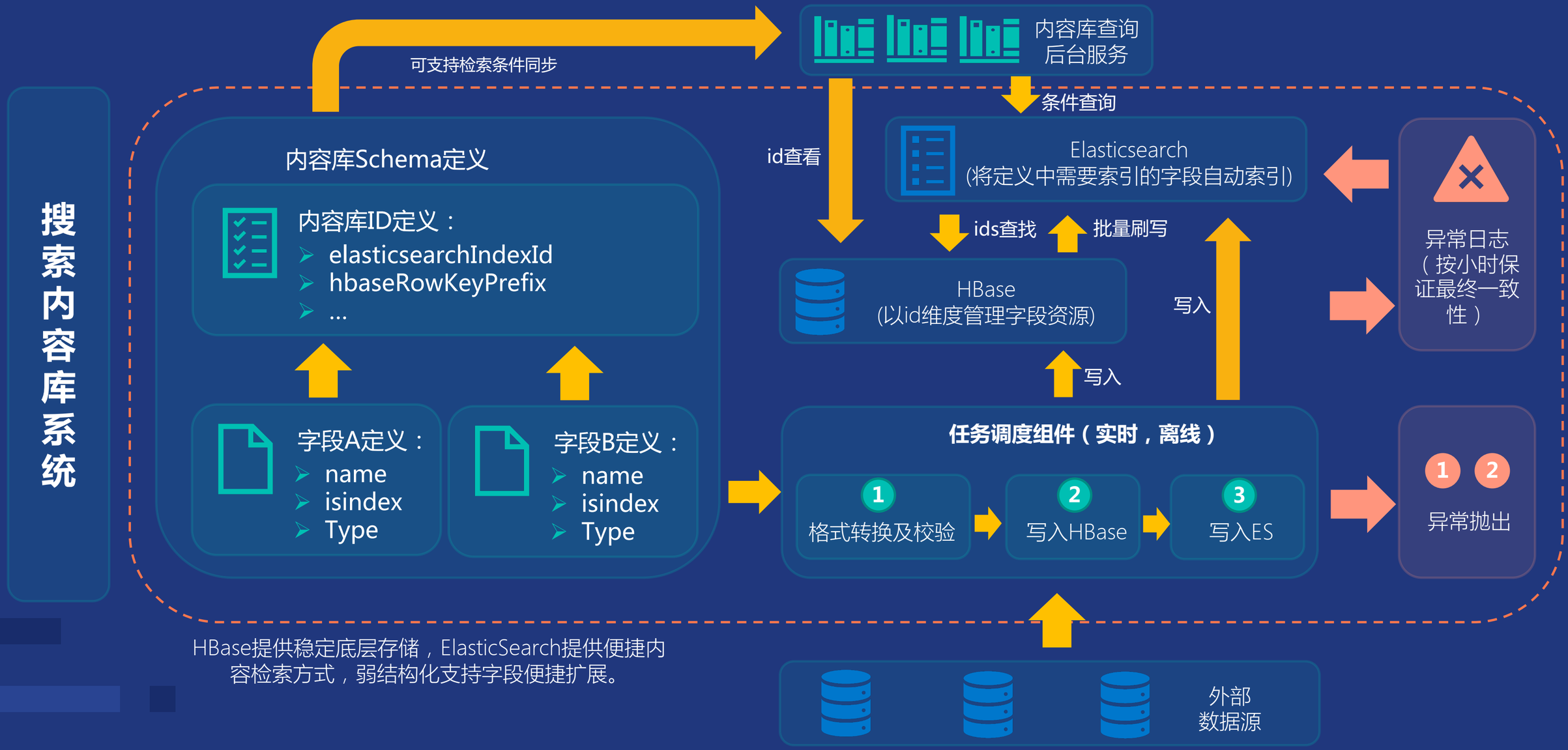


二、Elasticsearch在vivo搜索中台的实践

- Elasticsearch 在搜索中台内容库管理的实践
- Elasticsearch 在搜索中台文本召回的实践
- Elasticsearch 在搜索中台向量召回的实践
- Elasticsearch 在搜索中台精排的实践

Elasticsearch在搜索中台内容库管理的实践

搜索内容库架构



HBase提供稳定底层存储，ElasticSearch提供便捷内容检索方式，弱结构化支持字段便捷扩展。

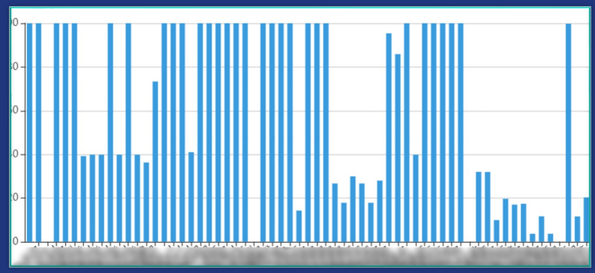
利用Elasticsearch提供内容库索引的优势

资源整体定义

搜索内容库索引

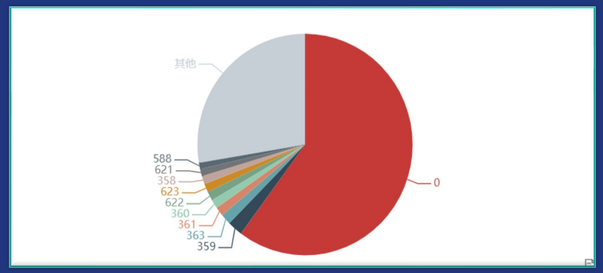
查询条件组合

字段存在性统计



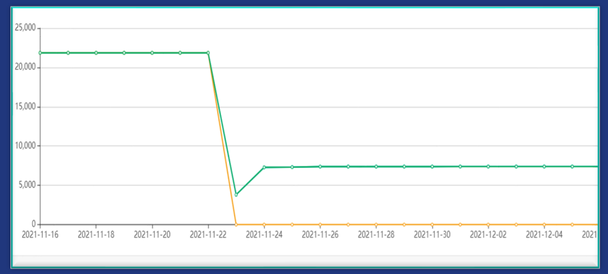
字段具体定义

字段top值分布



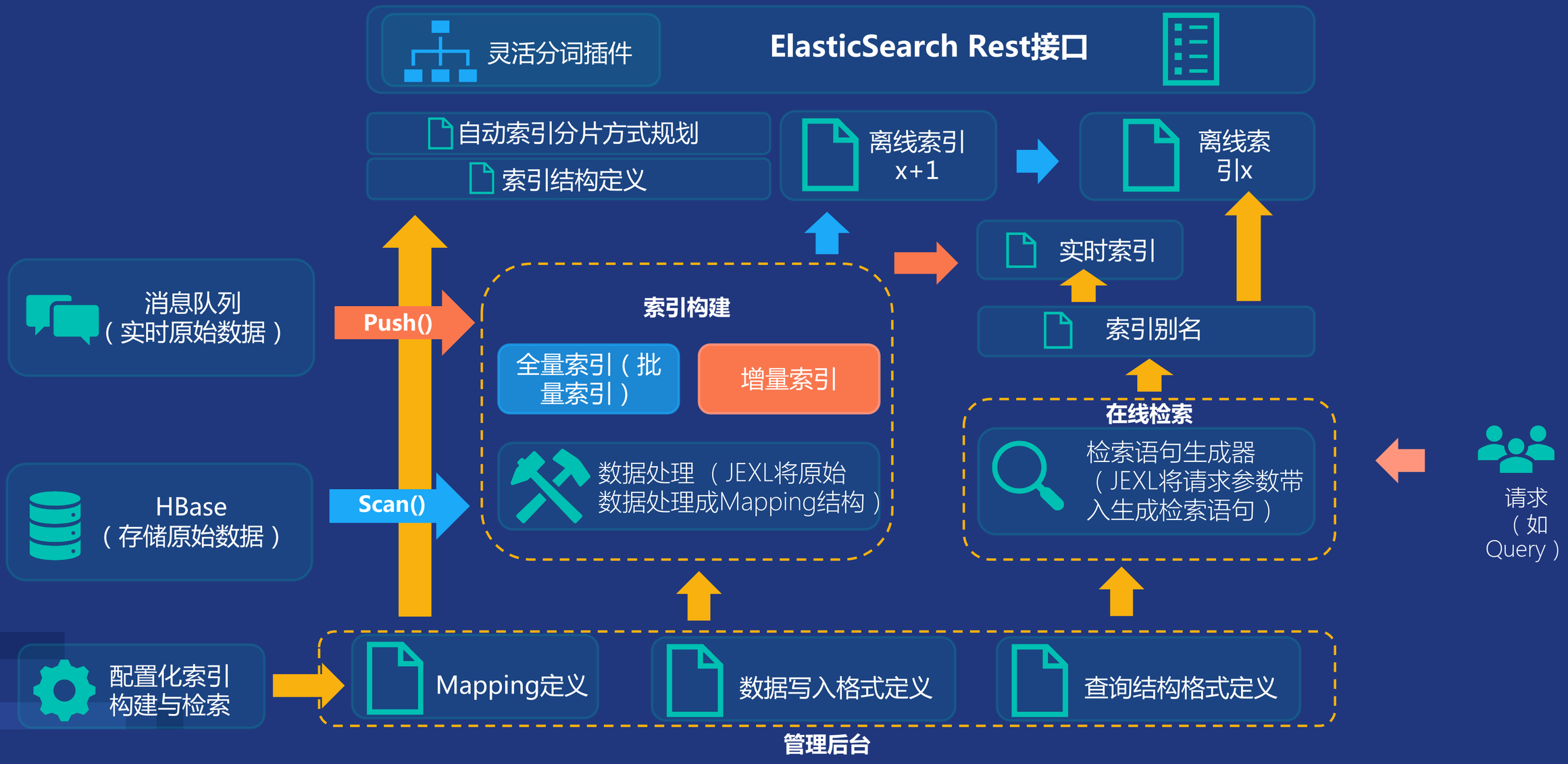
数据条目查看和管理

数据量变化 趋势监控



Elasticsearch在搜索中台文本召回的实践

索引管理架构


灵活分词插件
ElasticSearch Rest接口
自动索引分片方式规划
索引结构定义
离线索引x+1
离线索引x
实时索引
索引别名
消息队列（实时原始数据）
Push()
索引构建
全量索引（批量索引）
增量索引
数据处理（JEXL将原始数据处理成Mapping结构）
在线检索
检索语句生成器（JEXL将请求参数带入生成检索语句）
请求（如Query）
HBase（存储原始数据）
Scan()
配置化索引构建与检索
Mapping定义
数据写入格式定义
查询结构格式定义
管理后台

配置化的索引构建与检索

请求：
Query=“测试测试”

在线请求解析器（JEXL）

```
{
  "query": {
    "bool": {
      "must": [
        {"match": {"name": "${obj.getQuery()}"}},
        {"term":{"type":"1"}}
      ],
      "should": []
    }
  },
  "from": 0,
  "size": 100,
  "aggs": {}
}
```

[{"match":{"name":"测试测试"}}
{"term":{"type":"1"}}]

图谱（JanusGraph）

HBase Table1

Row1: id，name，type，number1,number2,outV

Row1: id，name，type，number1,number2,outV

HBase Table2

Row1:id,inV

Row2:id,inV

Scan 批量变更 **单条变更** **Hook变更**

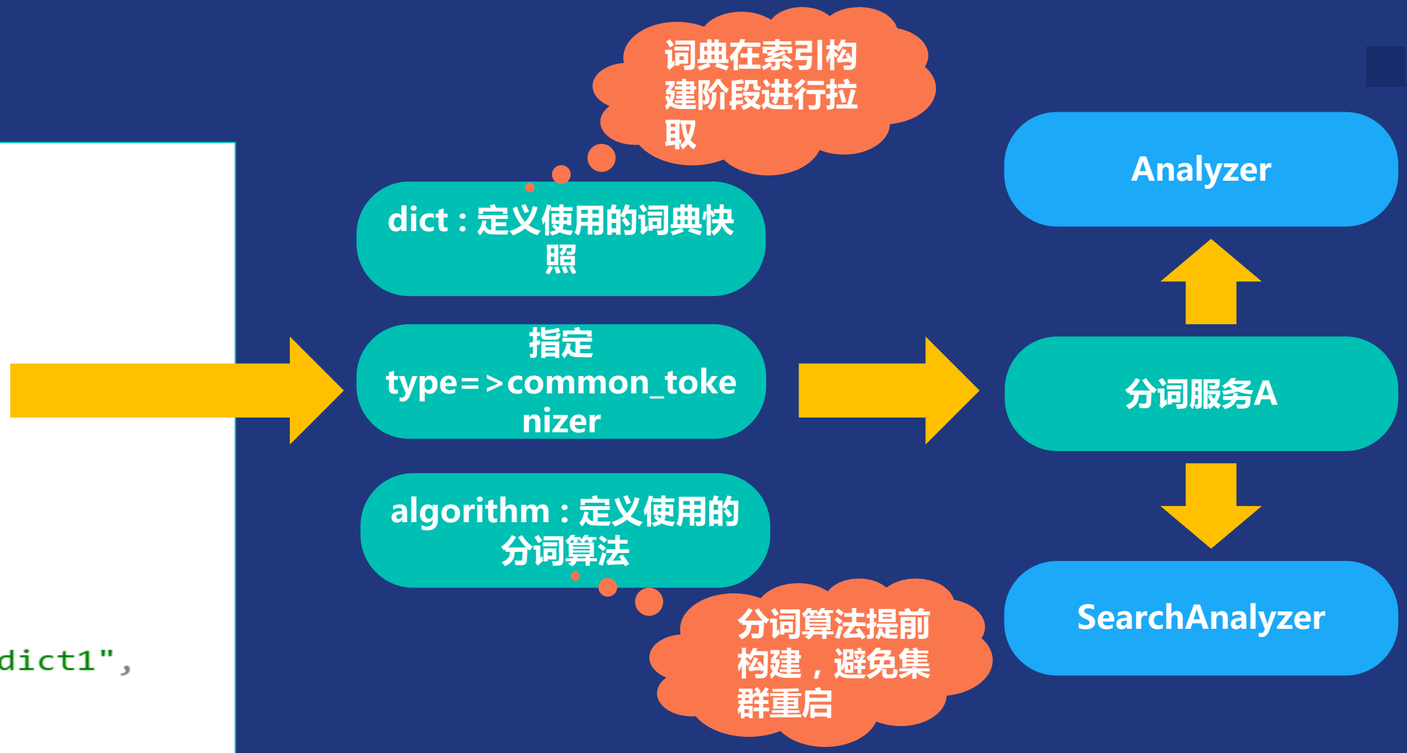
离线入库数据解析器（JEXL）

```
{
  "id": "${obj.targetSource().property('id')}",
  "name": "${obj.targetSource().property('name')}",
  "type": "${obj.targetSource().property('type')}",
  "totalNum": "${obj.targetSource().property('number1') + obj
    .targetSource().property('number2')}"
}
```

{"id":"xxx","name":"xxx","type":"xxx","totalNum":"xxx"}

灵活的分词

```
"settings": {
  "analysis": {
    "tokenizer": {
      "tokenizer_ik_smart_dict1": {
        "type": "common_tokenizer",
        "dict": "test_dict1",
        "algorithm": "ik_smart"
      }
    },
    "analyzer": {
      "analyzer_ik_smart_dict1": {
        "type": "custom",
        "char_filter": [],
        "tokenizer": "tokenizer_ik_smart_dict1",
        "filter": []
      }
    }
  }
}
```



问题1：分词后的term无法灵活更改（比如干预分词term）。
问题2：query的分词term难以附加词权重
问题3：match操作隐藏具体匹配细节，问题不易定位

场景1：需要term的位置信息时

提前分词，利用空格分开，再利用空格分词器进行分词

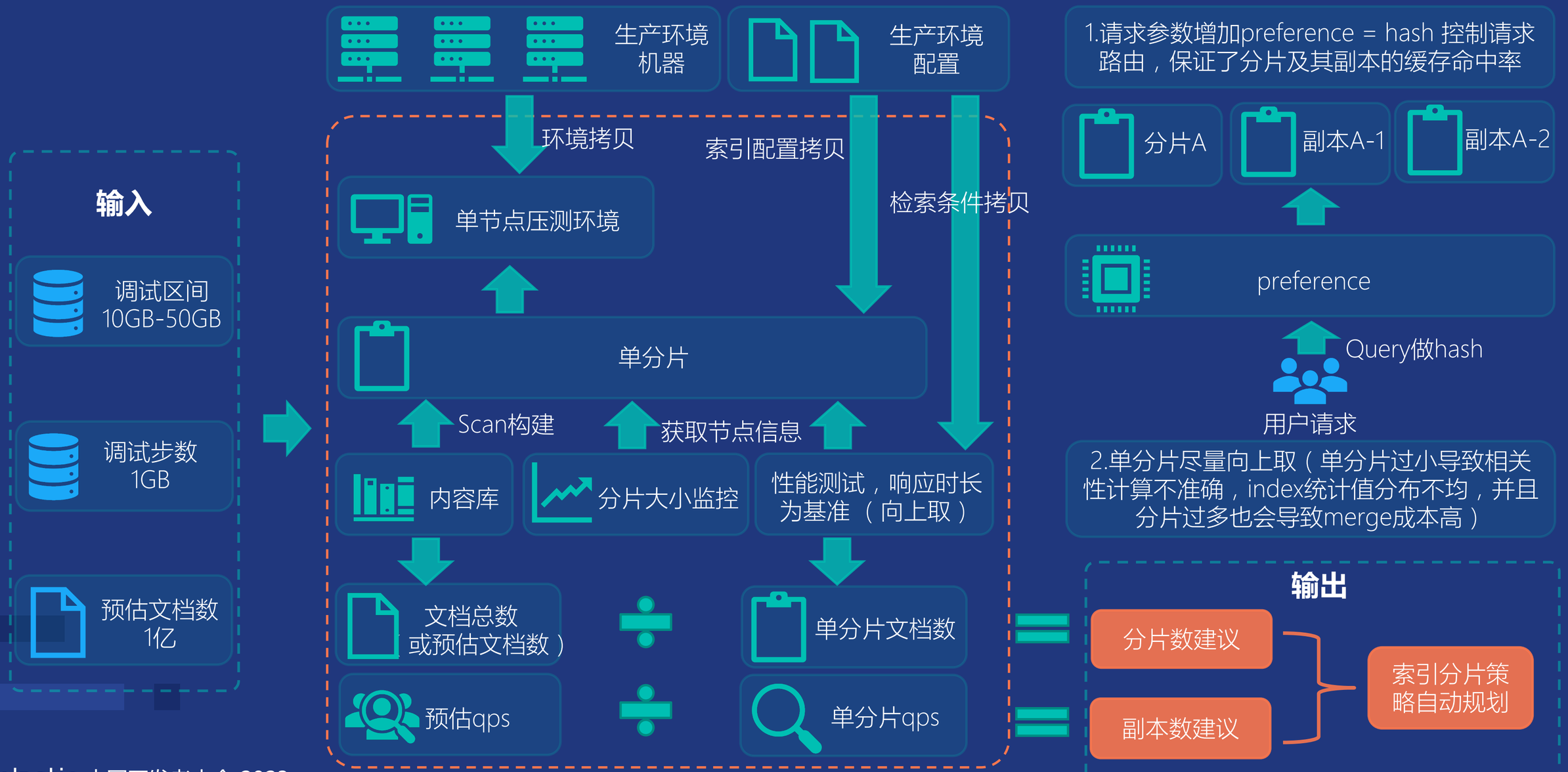
场景2：无需用到位置信息

提前分词，使用keyword，并将分词结果创建成数组的形式

场景3：需要对分词term附加权重

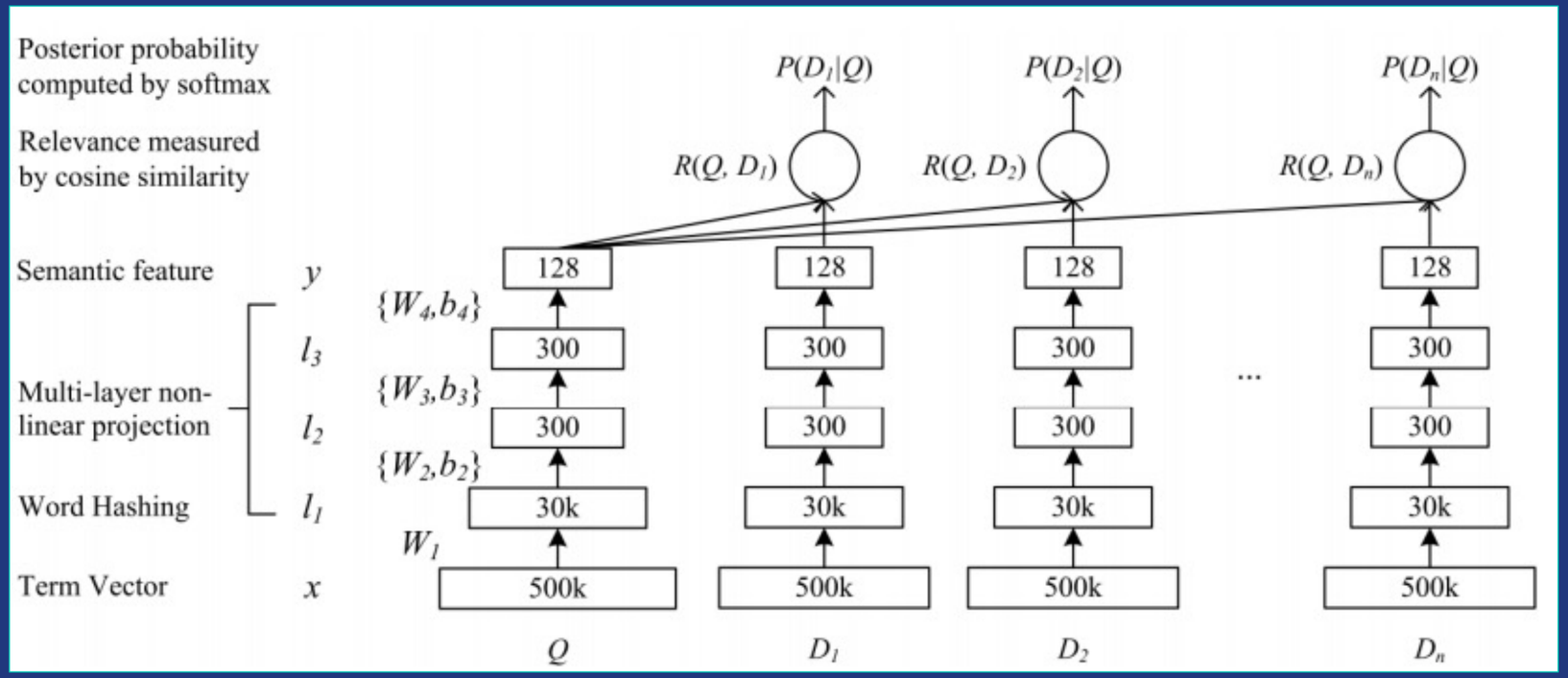
利用bool表达式+should+term查询替换match

自动索引分片规划


输入
调试区间
10GB-50GB
调试步数
1GB
预估文档数
1亿
生产环境
机器
生产环境
配置
环境拷贝
索引配置拷贝
检索条件拷贝
单节点压测环境
单分片
Scan构建
获取节点信息
内容库
分片大小监控
性能测试，响应时长
为基准 （向上取）
文档总数
或预估文档数）
单分片文档数
预估qps
单分片qps
1.请求参数增加preference = hash 控制请求
路由，保证了分片及其副本的缓存命中率
分片A
副本A-1
副本A-2
preference
Query做hash
用户请求
2.单分片尽量向上取（单分片过小导致相关
性计算不准确，index统计值分布不均，并且
分片过多也会导致merge成本高）
输出
分片数建议
副本数建议
索引分片策
略自动规划

Elasticsearch在搜索中台向量召回的实践

向量索引的构建与检索基本架构



DSSM架构

Query->Doc的转换概率（用户行为）

Embedding过程

Query和doc(比如title或其他可向量化表示的特征)

文档标题，正文

查询query

Embedding（Bert预训练模型）

文档向量化表示

query向量化表示

入库

检索

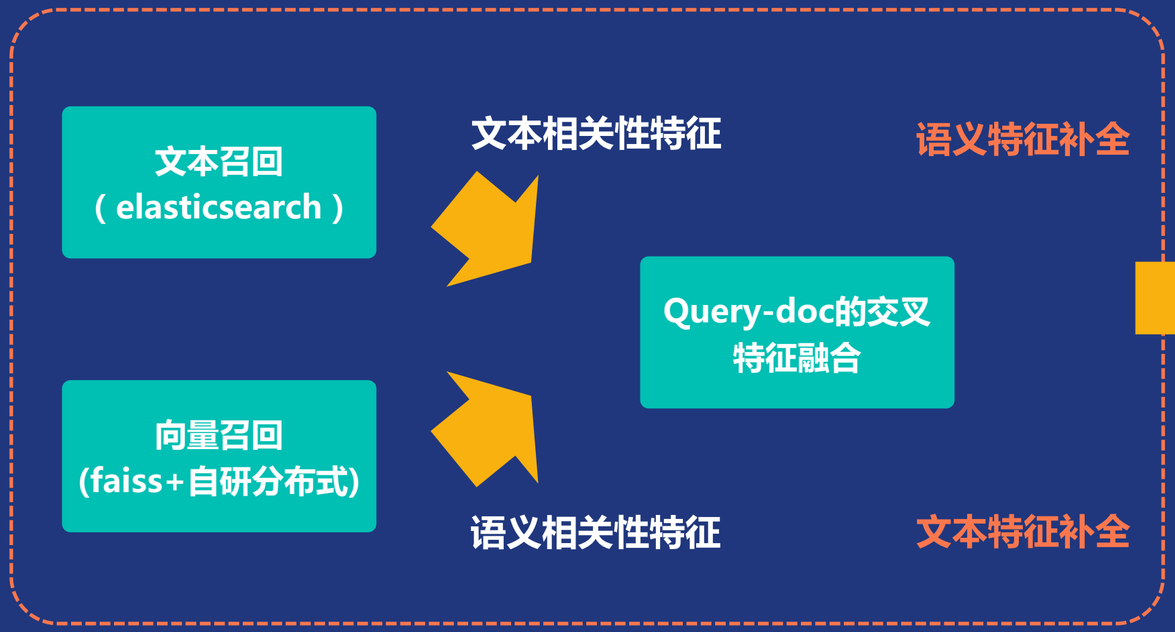
向量引擎

faiss

elasticsearch

Elasticsearch向量检索优化特征融合链路

es8之前架构



Query-doc的相关性交叉特征
需要再重新回捞计算实现成本
高，一致性难保证

knn+query语法

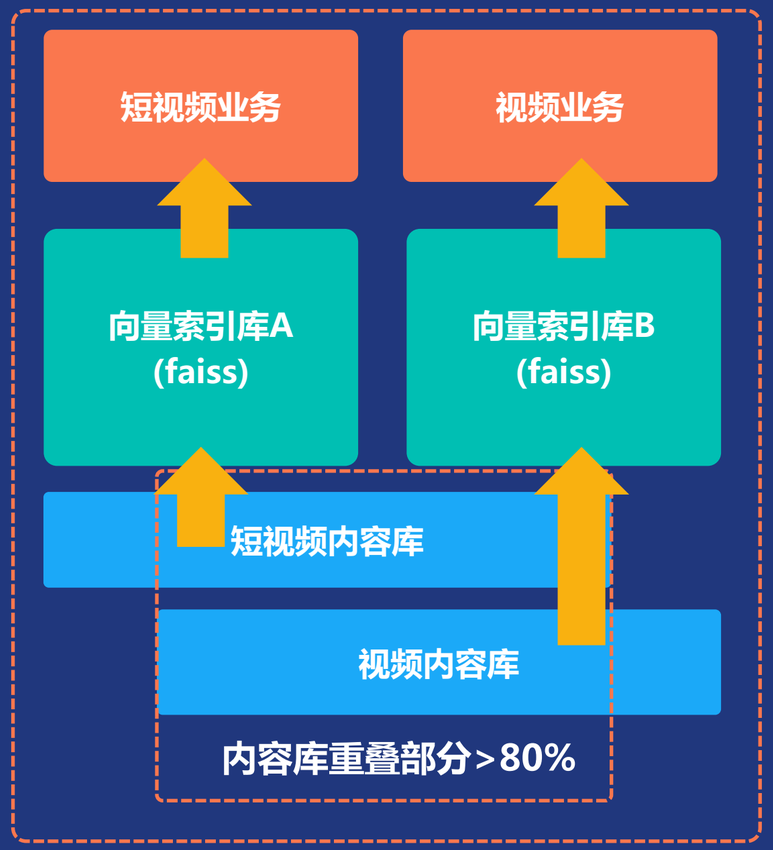
```
{
  "query": {
    "match": {
      "title": {
        "query": "测试测试",
        "boost": 0.5
      }
    }
  },
  "knn": {
    "field": "title-vector",
    "query_vector": [0.1,0.2,0.3],
    "k": 10,
    "num_candidates": 50,
    "boost": 0.5
  },
  "size": 10
}
```

es8之后架构

Query-doc的相关性特
征都交给es计算，实现
简单，一致性容易保证

Elasticsearch向量检索预过滤实践

利用传统向量引擎



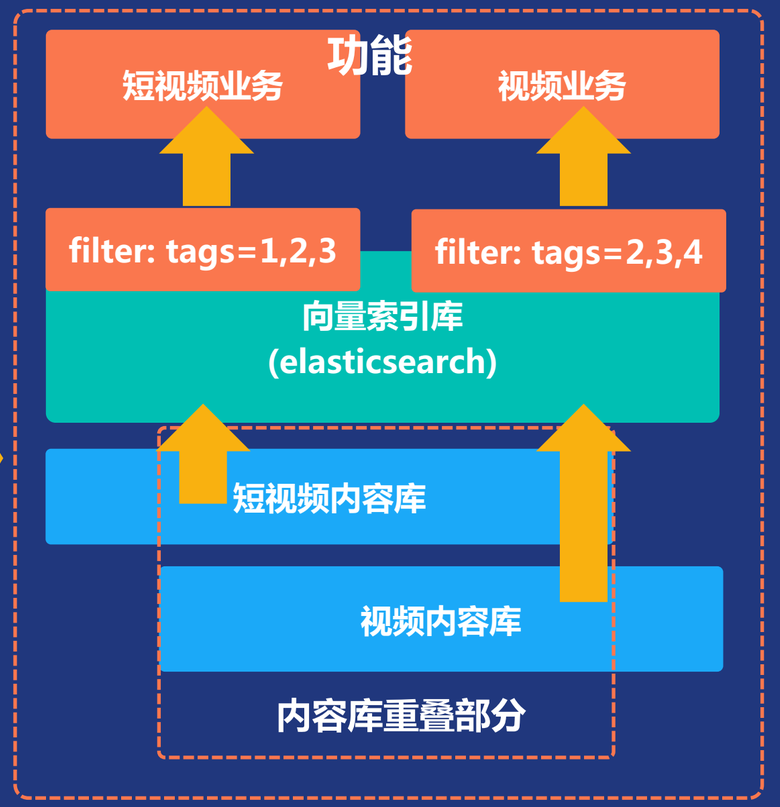
传统向量引擎，不保存正排信息也不支持过滤，如果后置过滤则导致召回结果变少甚至出现无结果的情况，尽管模型一致且内容接近，仍然部署了两个索引库

Knn过滤配置

```
{
  "knn": {
    "field": "title-vector",
    "query_vector": [0.1,0.2,0.3],
    "k": 10,
    "num_candidates": 100,
    "filter": {
      "terms": {
        "tags": [1,2,3]
      }
    }
  }
}
```

es向量索引支持在检索过程中过滤（存放正排的天然优势）

利用es向量检索过滤功能

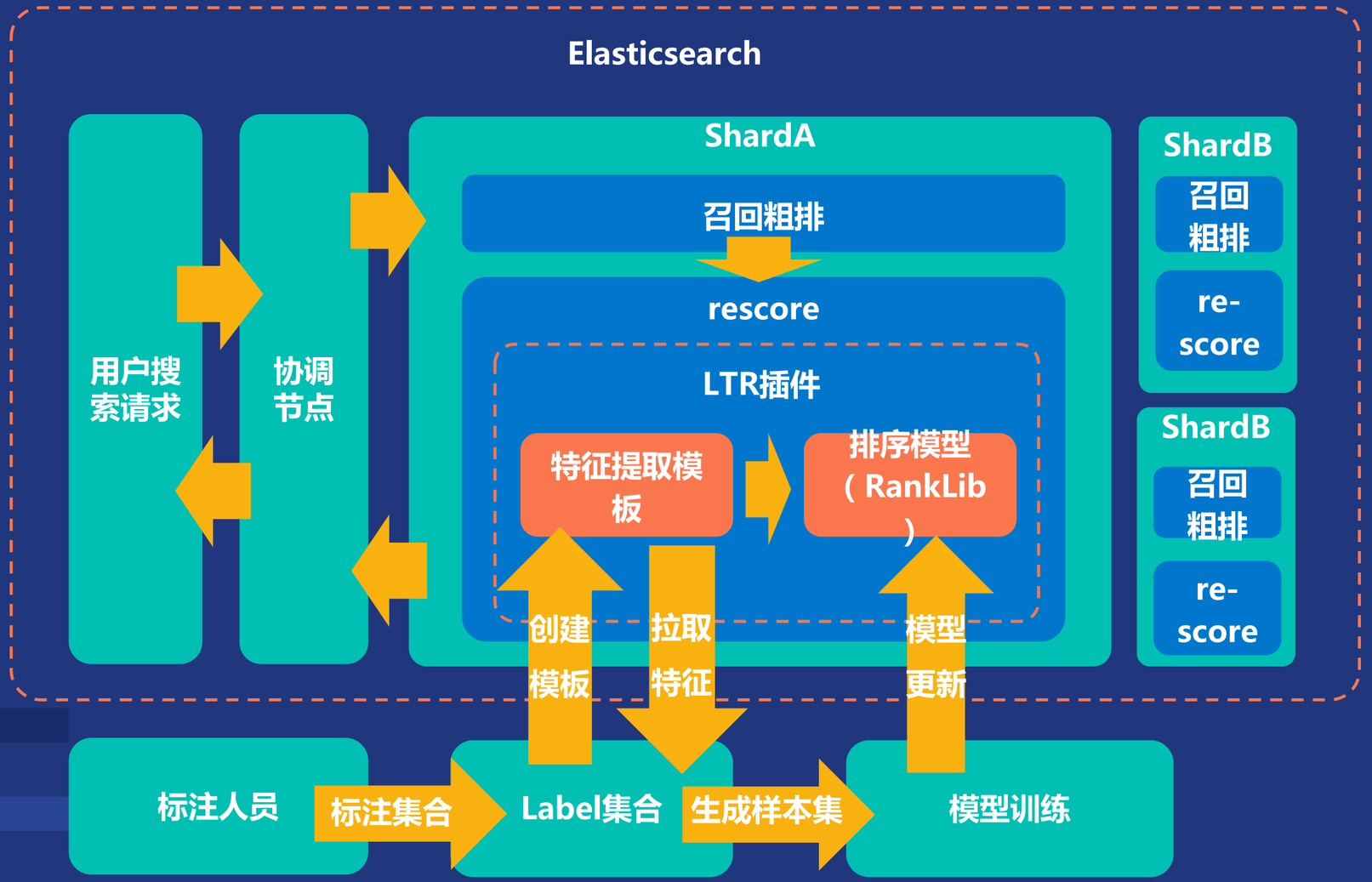


统一索引库，在上层进行过滤操作。索引库统一后节省了大量的维护成本

Elasticsearch在搜索中台精排的实践

利用rescore插件完成精排

使用LTR插件解决算法排序问题(传统机器学习模型)



优点：

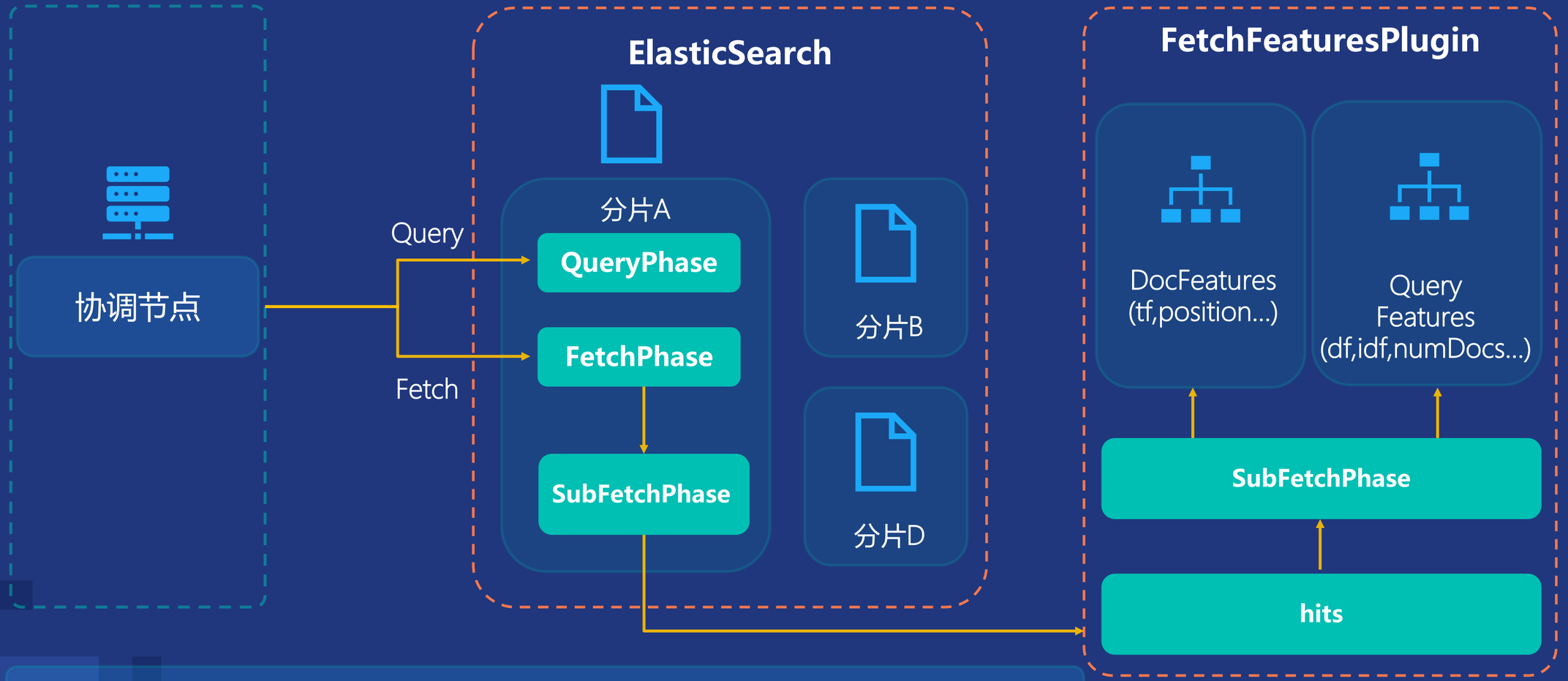
1.提供基础特征（痛点）
2.增加并行度，减少数据传输量

缺点：

1.绑定太死，多路召回不便同排
2.增加多余的性能负载（rescore）
3.接入非java预估服务实现成本高

提取相关性特征

开发特征提取插件，解决基础相关性特征无法透出的痛点。



放在Fetch阶段，减少机器负载，和网络传输量，将结果传回调用服务再进行算法调用，提升灵活性。

特征提取效果演示

```
"query": {
  "bool": {
    "should": [
      {
        "bool": {
          "_name": "title_terms",
          "should": []
        }
      }
    ]
  }
},
"ext": {
  "feature_log": {
    "items": [
      {
        "feature_name": "title_query_features",
        "named_query": "title_terms"
      },
      {
        "feature_name": "title_field_features",
        "field_name": "title"
      }
    ]
  }
},
"from": 0,
"size": 80,
```

```
"title": "分析搜索引擎对网站的....",
"_score": 64.90771,
"_fields": {
  "log_features": [
    {
      "title_query_features": {
        "搜索引擎_df": 2047,
        "搜索引擎_tf": 1,
        "搜索引擎_numDocs": 2938397,
        "搜索引擎_position": [
          1
        ],
        "origin": 15.411291,
        "搜索引擎_idf": 8.268756
      },
      "title_field_features": {
        "adtn": 16.142025,
        "dtn_dr": 11,
        "adtn_dr": 14.749546,
        "dtn": 11
      }
    }
  ]
},
"status": 1
```

查询特征

文档字段特征，比如dtn（ document_term_num ）文档该字段的term总数

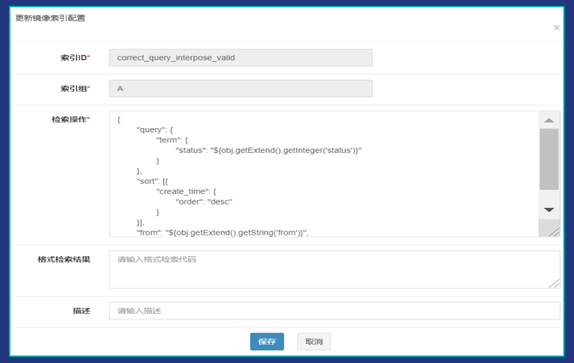
三、搜索中台业务应用总结

搜索中台编排系统

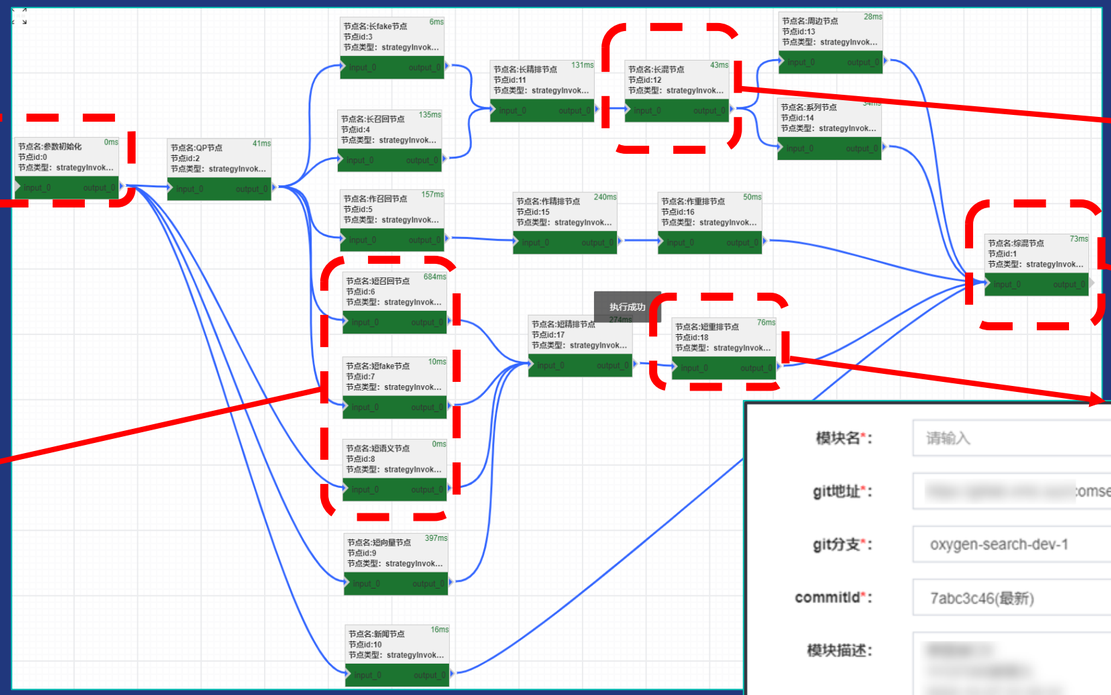
支撑若干业务的搜索中台编排系统

编排系统 搜索

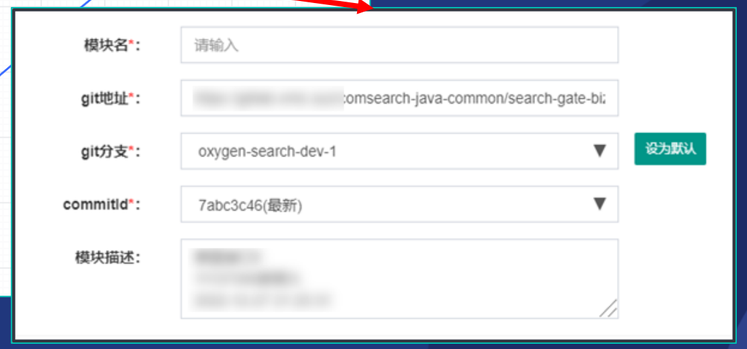
分词及意图组件



召回组件（elasticsearch,faiss 语义召回，c++自研召回）



算法预估组件



动态脚本组件

动态渲染组件

elastic 中国开发者大会 2023

感谢观看

elastic
中文社区

专业、垂直、纯粹的 Elastic 开源技术交流社区

https://elasticsearch.cn/