简单·可信赖

基于ElasticSearch构建搜索云服务实战

陈 超@CrazyJvm
七牛云 技术总监

主要内容

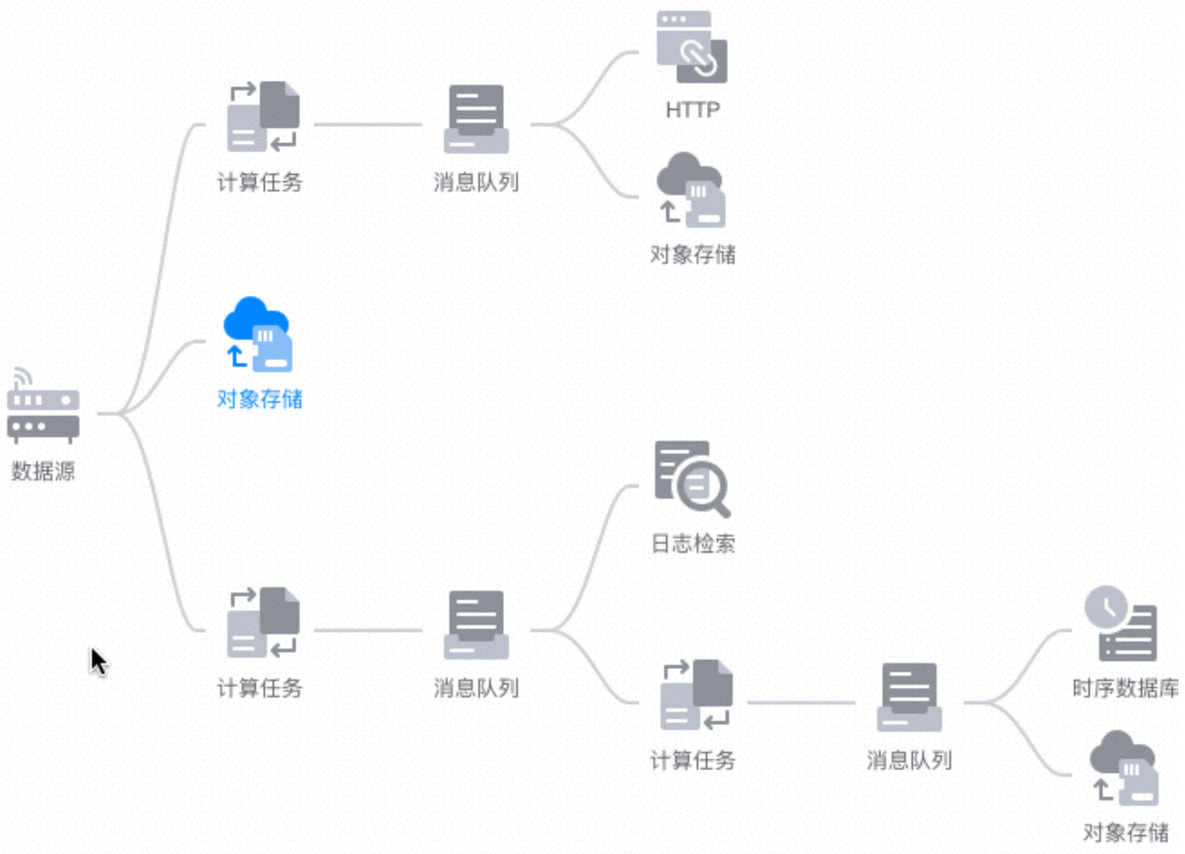
- 背景
- 设计目标
- 遇到的挑战
- 如何应对
- 成果总结

Pandora

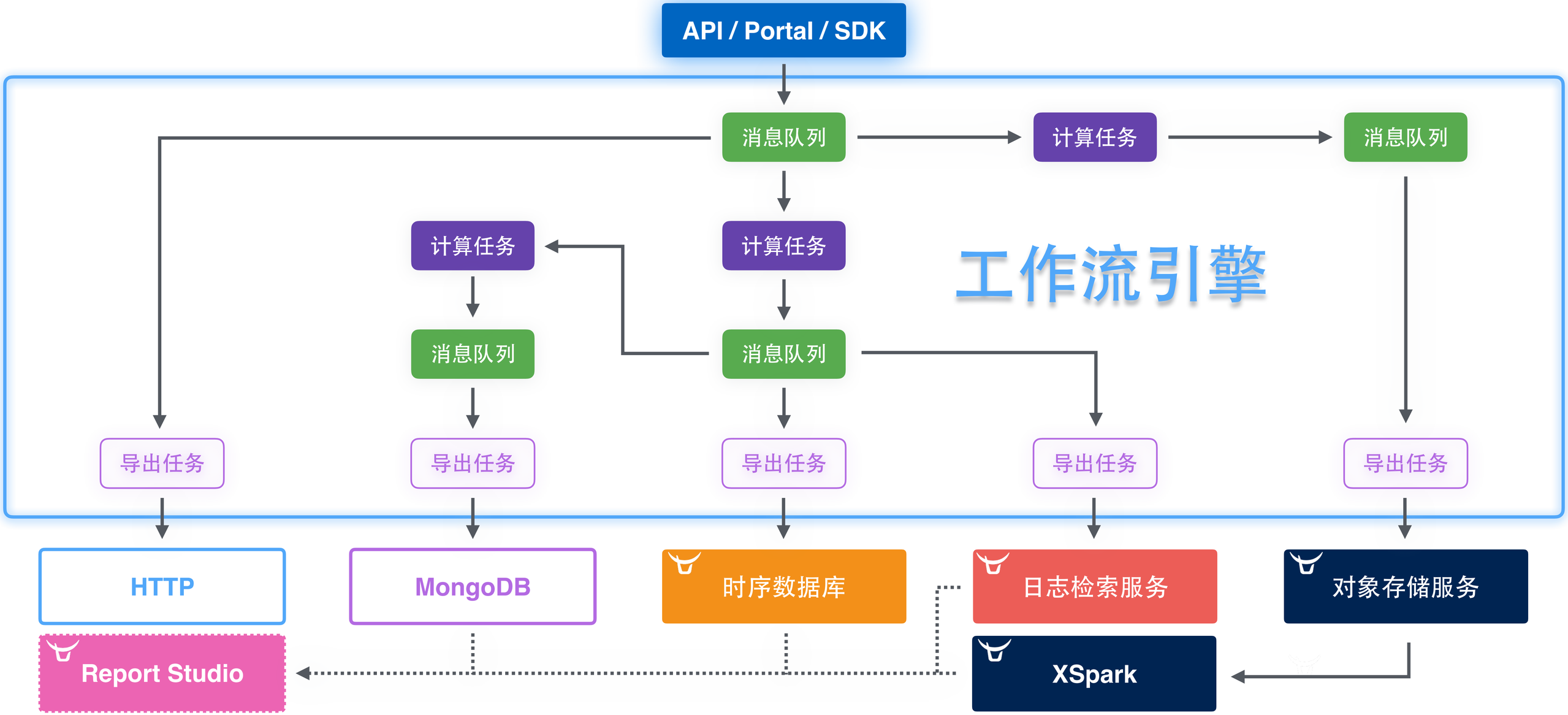
简单　　高效　　开放

Pandora 大数据平台

背景

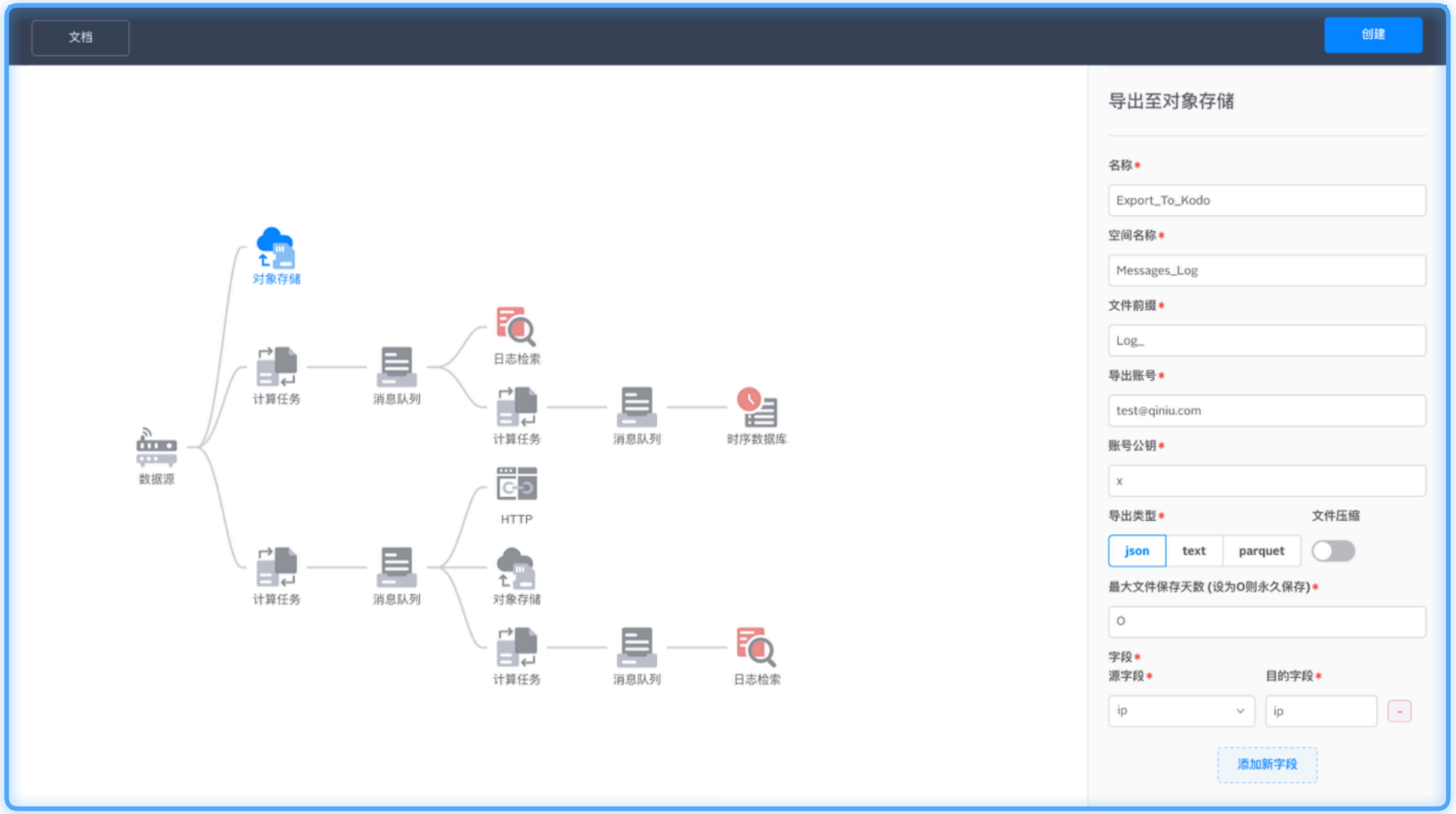


Pandora架构图


API / Portal / SDK
消息队列
计算任务
消息队列
计算任务
计算任务
工作流引擎
消息队列
消息队列
导出任务
导出任务
导出任务
导出任务
导出任务
HTTP
MongoDB
时序数据库
日志检索服务
对象存储服务
Report Studio
XSpark

工作流引擎

1. 实时接收数据
2. 实时计算
3. 实时导出
4. 业务逻辑拓扑
5. 离线计算
6. 简易离线任务调度



工作流引擎的特点

低学习、使用成本

只需会编写简单的SQL，即可轻松完成大数据生命周期管理；除计算逻辑外，无需编写一行代码！

可视化业务、数据流管理

完全拓扑的数据流架构，通过页面可以直观的感受到业务、数据的流向；同时，通过不同工作流的隔离，使业务监控报警、管理更加清晰！

时序数据库

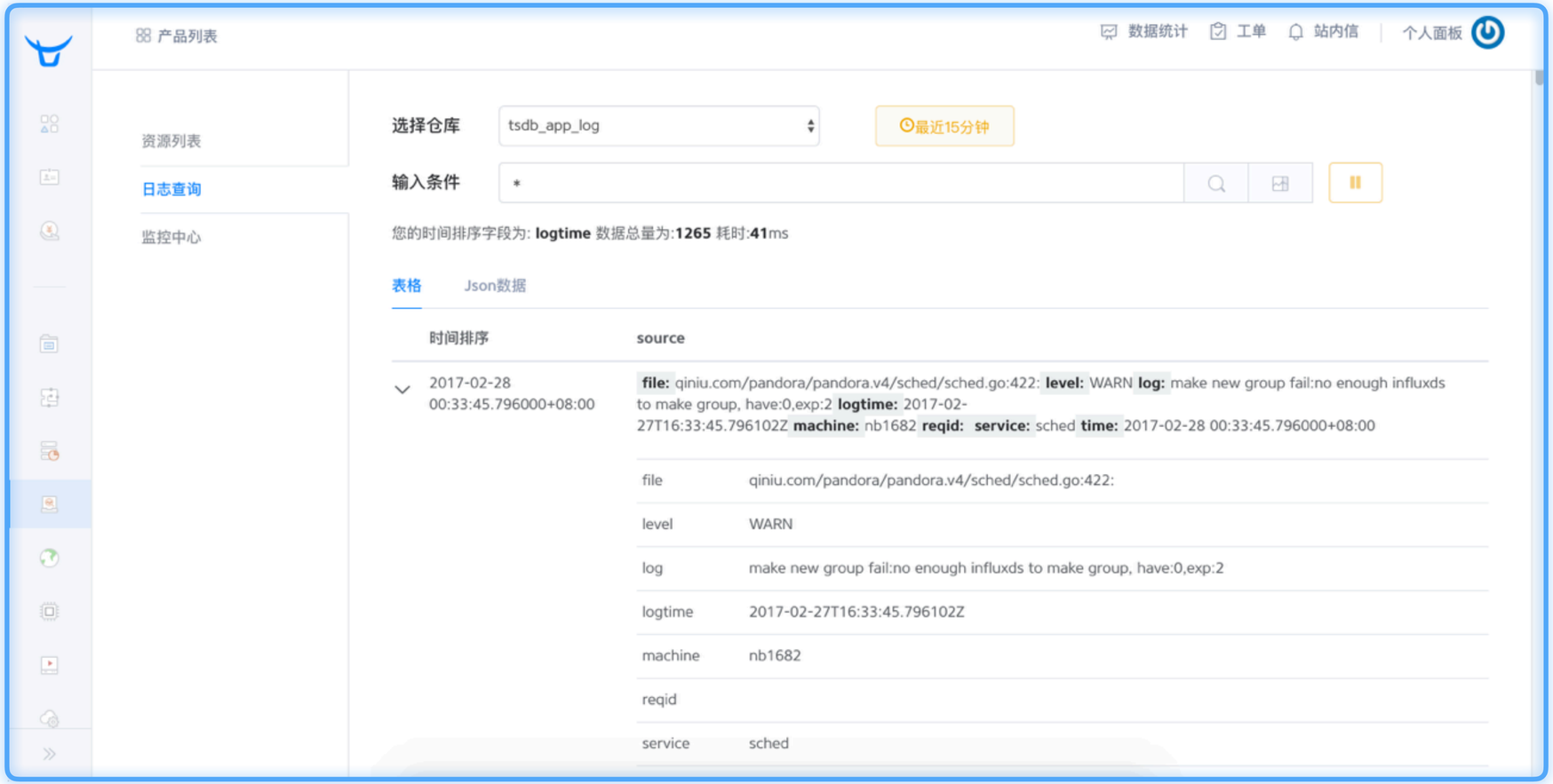
时序数据库，全名时间序列数据库（ **Time Series Database** ）。时间序列数据库主要用于处理带有时间标签（ 按照时间的顺序变化，即时间序列化 ）的数据，包括**DevOps**监控数据、物联网传感器数据等。

时序数据库的特点

- 高速数据汇入，七牛时序数据库单节点提供高达每秒 60W 的数据汇入速度；
- 即时查询，数据写入成功即可查询，无延迟；
- 持续聚合（拥有持续聚合计算能力）；
- 横向拓展，集群拓展横向叠加，性能不打折；
- 冷热分离，对数据按照时间分片，使热数据查询性能更高；
- 高速读写；
- 高比例的数据压缩；

日志检索服务

日志检索服务（LogDB）提供针对日志类数据的存储与检索服务，用户无需开发就能快捷完成数据存储、检索分析功能，帮助提升运维、运营效率，建立DT时代海量日志处理能力。



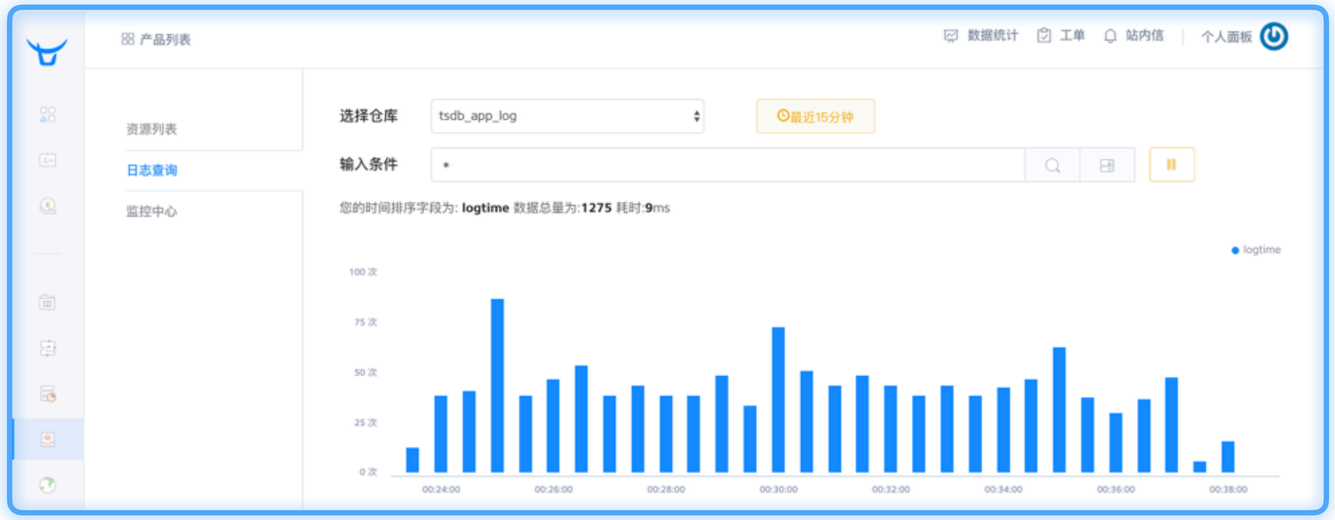
日志检索服务适用场景

日志管理

应用部署上线速度快、日志种类越来越多，每个应用都包含访问日志（Access）、操作日志（OpLog）、业务逻辑（Logic）和错误（Error）等种类。当新增应用、应用与应用之间有依赖时，日志数目也会爆发。

事件溯源

通过不同的日志进行综合分析，可以快速分析出应用的事故原因、持续时间、影响范围等重要指标，对于事后采取善后措施提供有力的数据支持。



日志检索服务的特点

便捷

页面/API简洁，5分钟上手

稳定

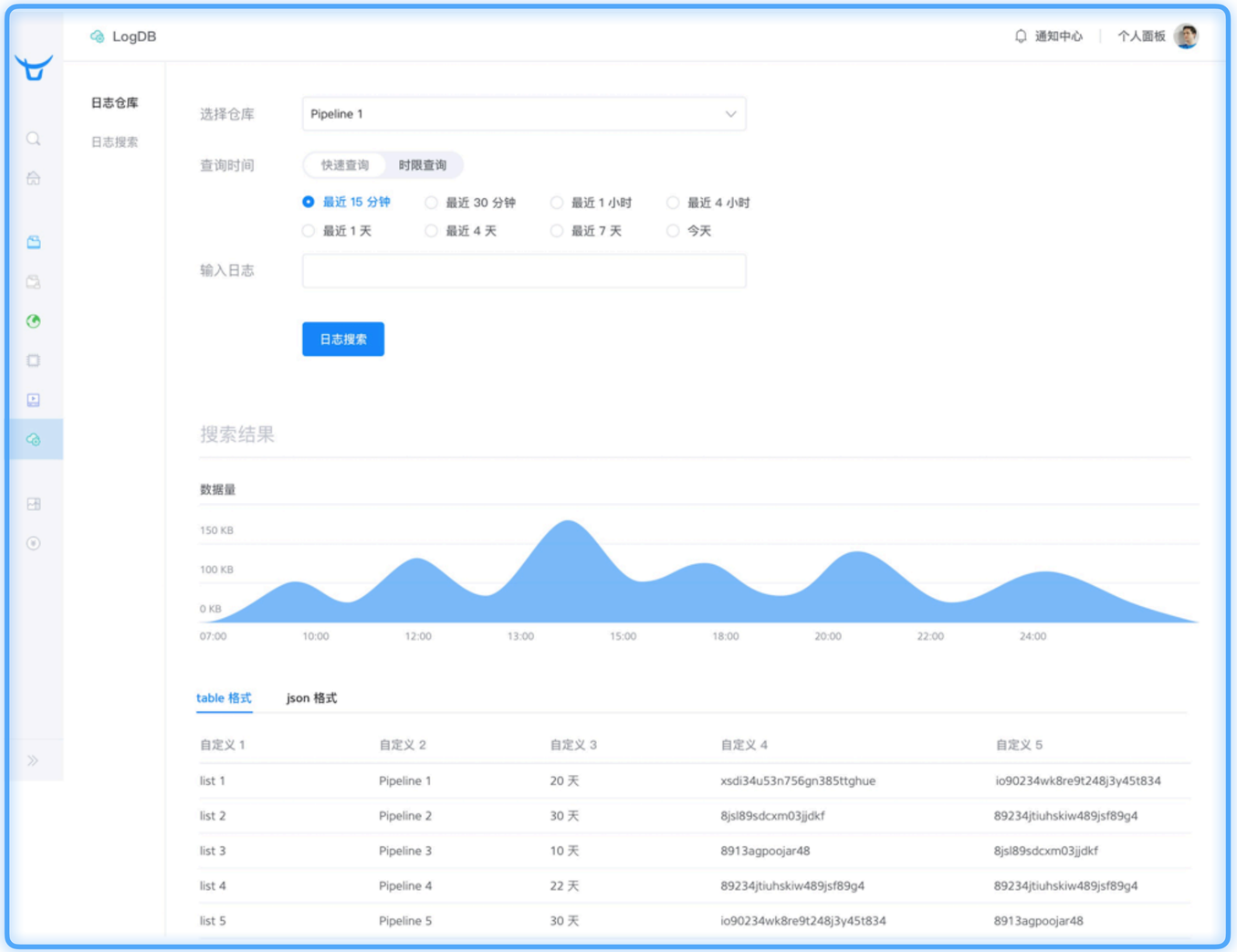
弹性容量满足从MB-100TB/天需求

>99.9% 可用性

低成本

按量计费，成本低自建60%以上

0运维/0部署/0开发

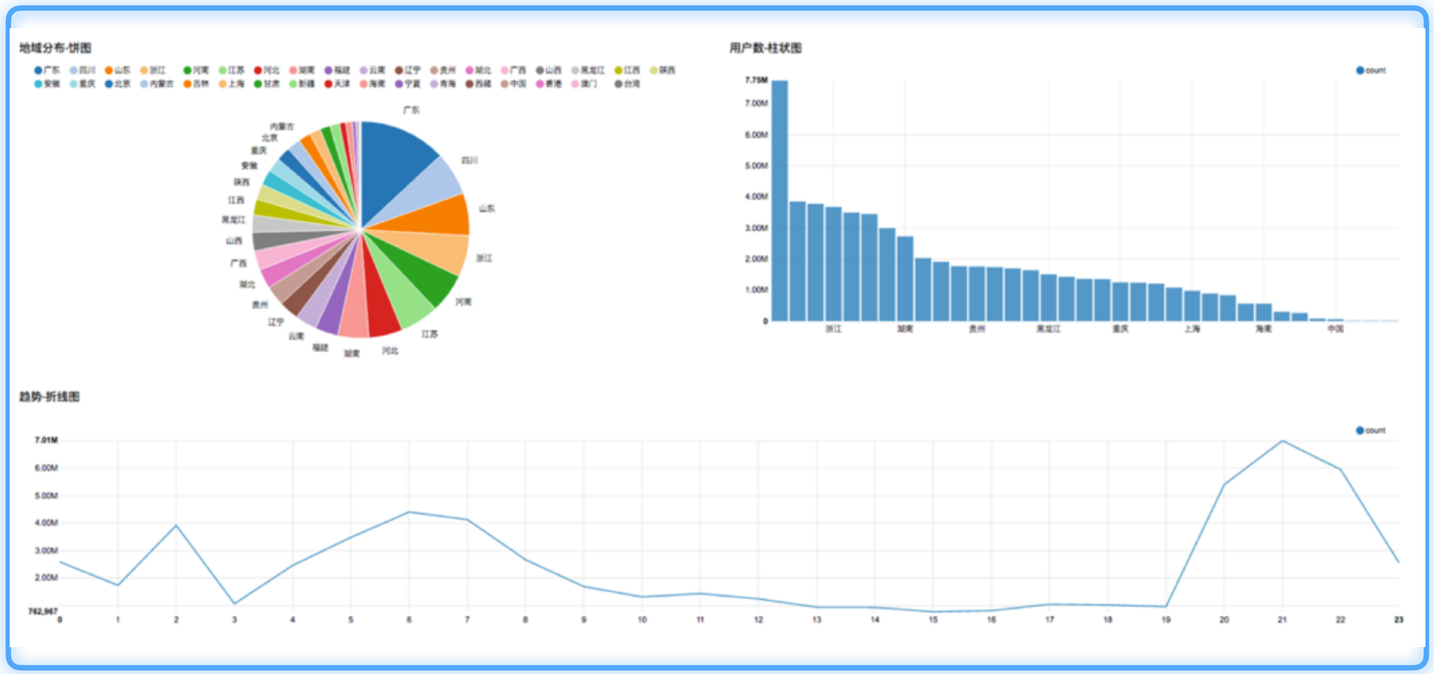


XSpark

XSpark（大数据离线分析服务）是基于容器技术的大数据（科学）分析应用。无需关注复杂的大数据技术底层服务，直接编码进行数据分析，可视化展示分析结果。

功能：

- 机器学习和数据挖掘
- 模型训练和使用
- 数据可视化与逻辑图表



XSpark的特点

便捷

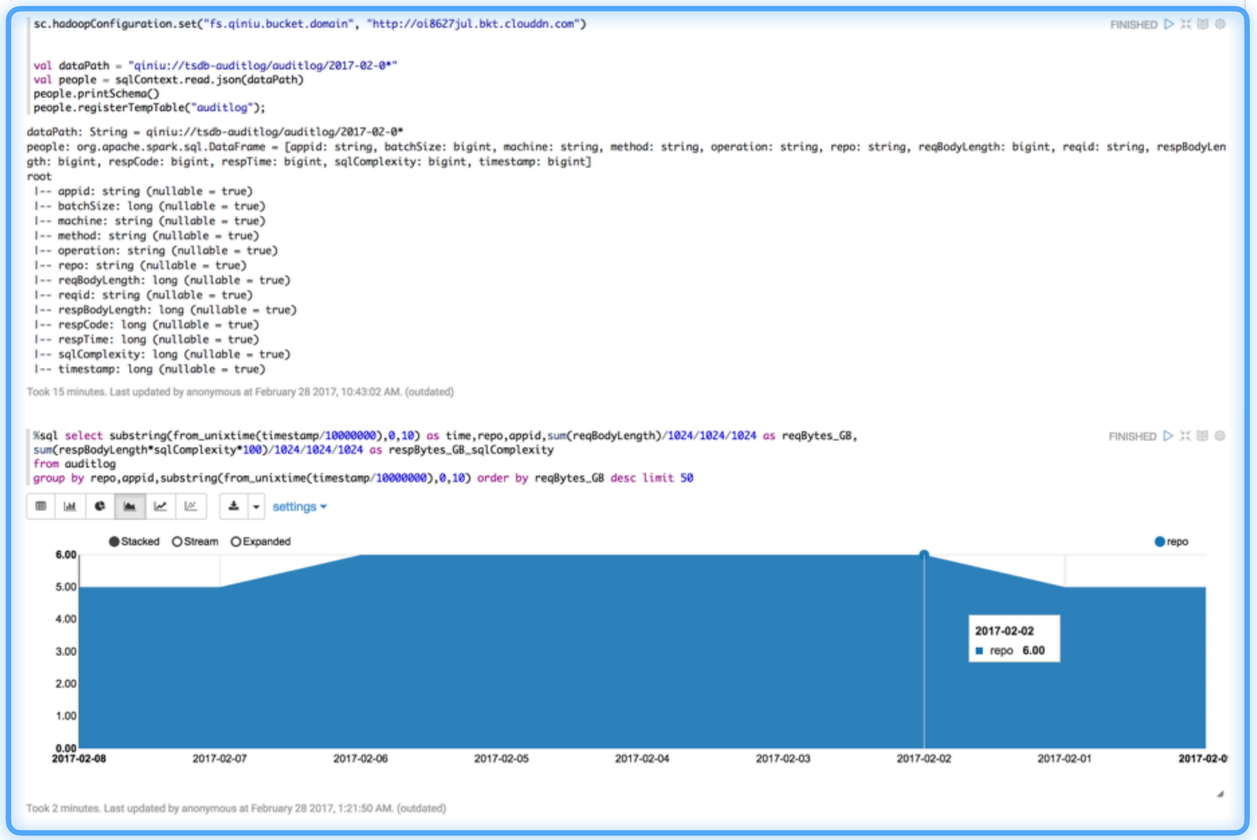
- 数据来源云存储，无需运维HDFS集群
- 多种编程语言支持

低成本

- 动态伸缩，节省资源与成本

稳定

- 基于容器的自动部署，自动灾备



背景

LogDB 定位是什么?

- 基于pandora针对日志类数据提供分析服务
- 用户5~10分钟可以完成接入
- 可以承载MB～100TB／天的日志增量
- 0运维、0开发、低成本

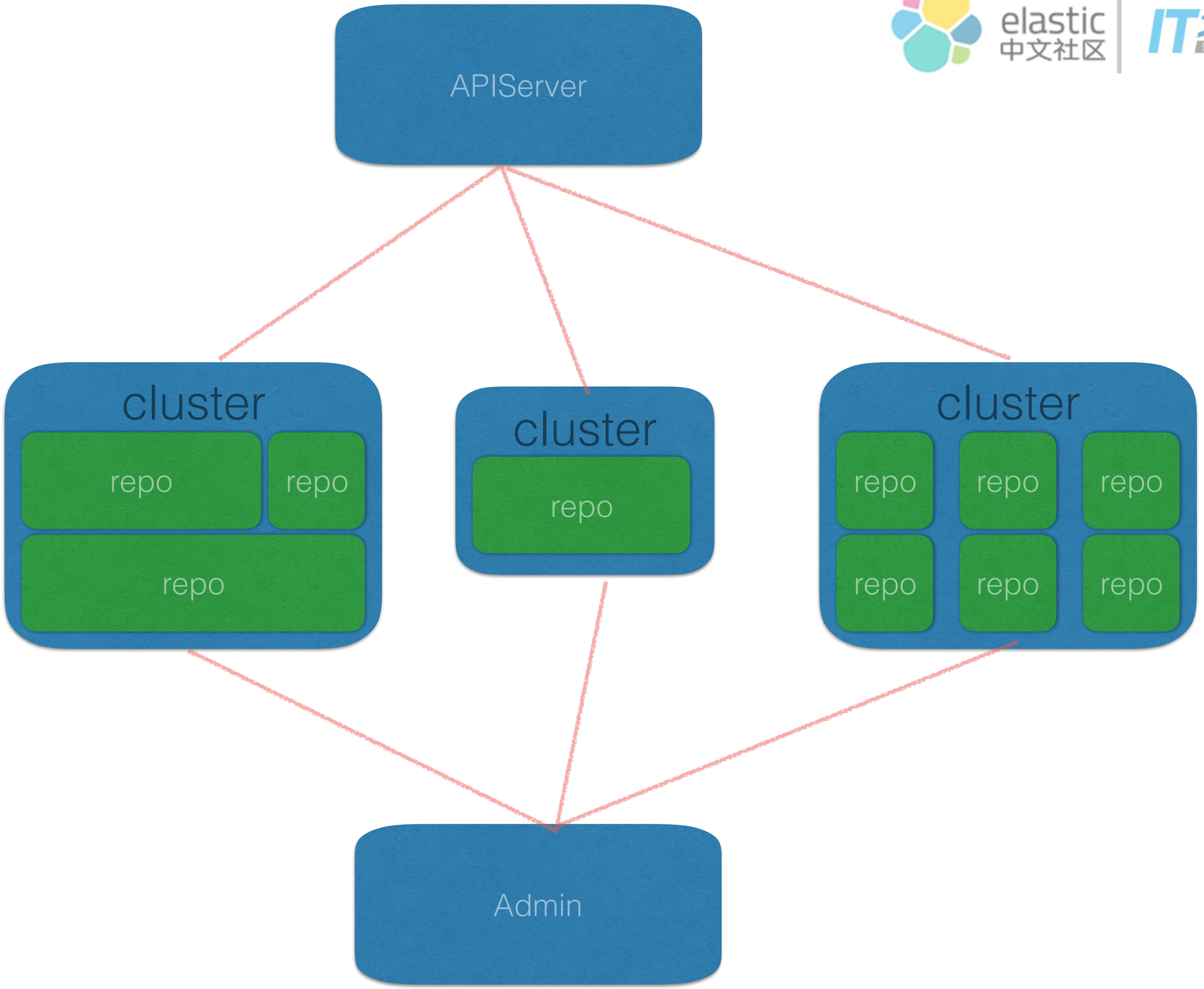
设计目标

LogDB 设计目标是什么?

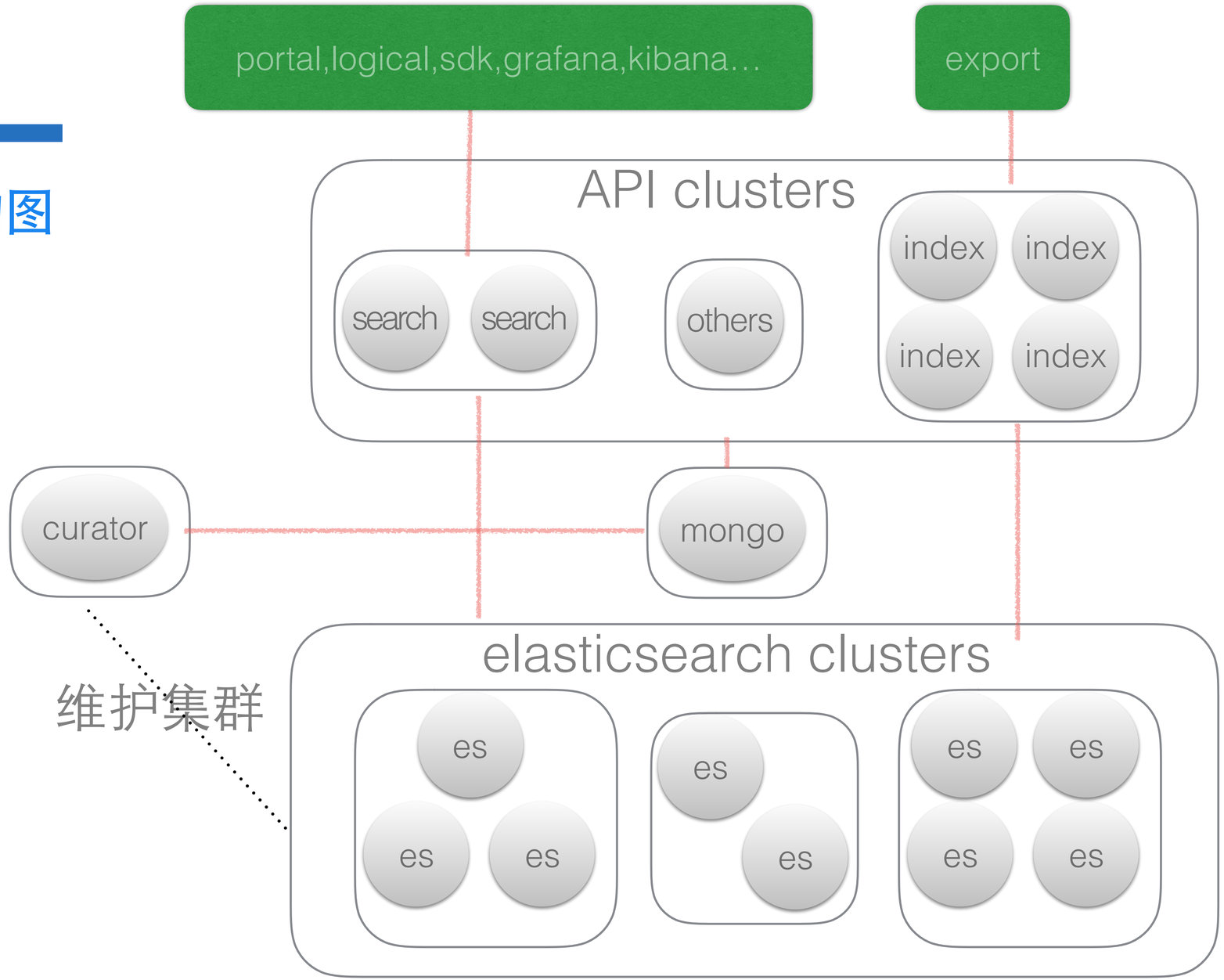
- 支持公有云海量用户
- 支持单个用户日志规模MB-PB/天
- 查询秒级响应
- 高可靠性
- 拥抱开源，可以适配kibana，grafana等

多租户模型

海量cluster



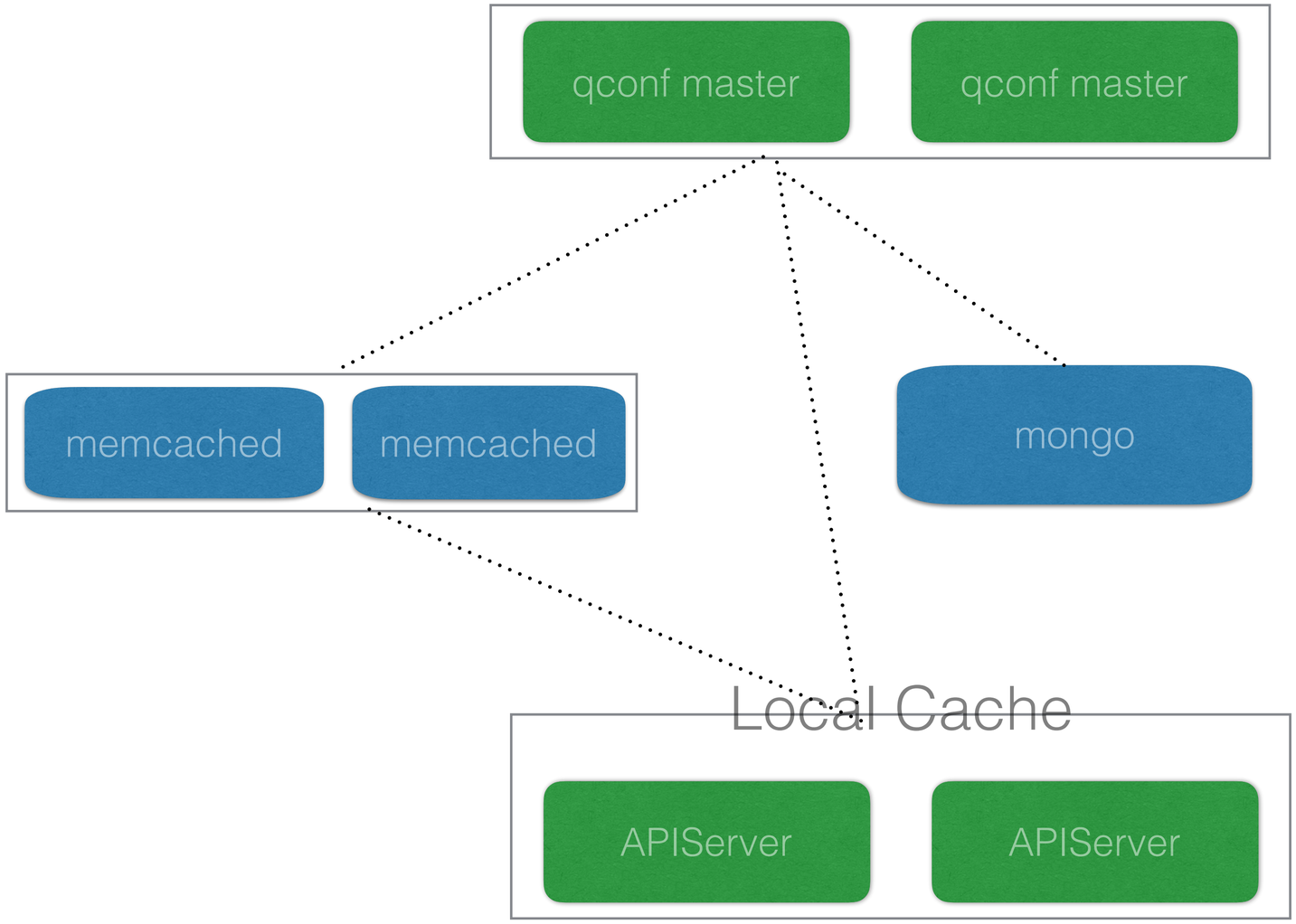
系统架构图



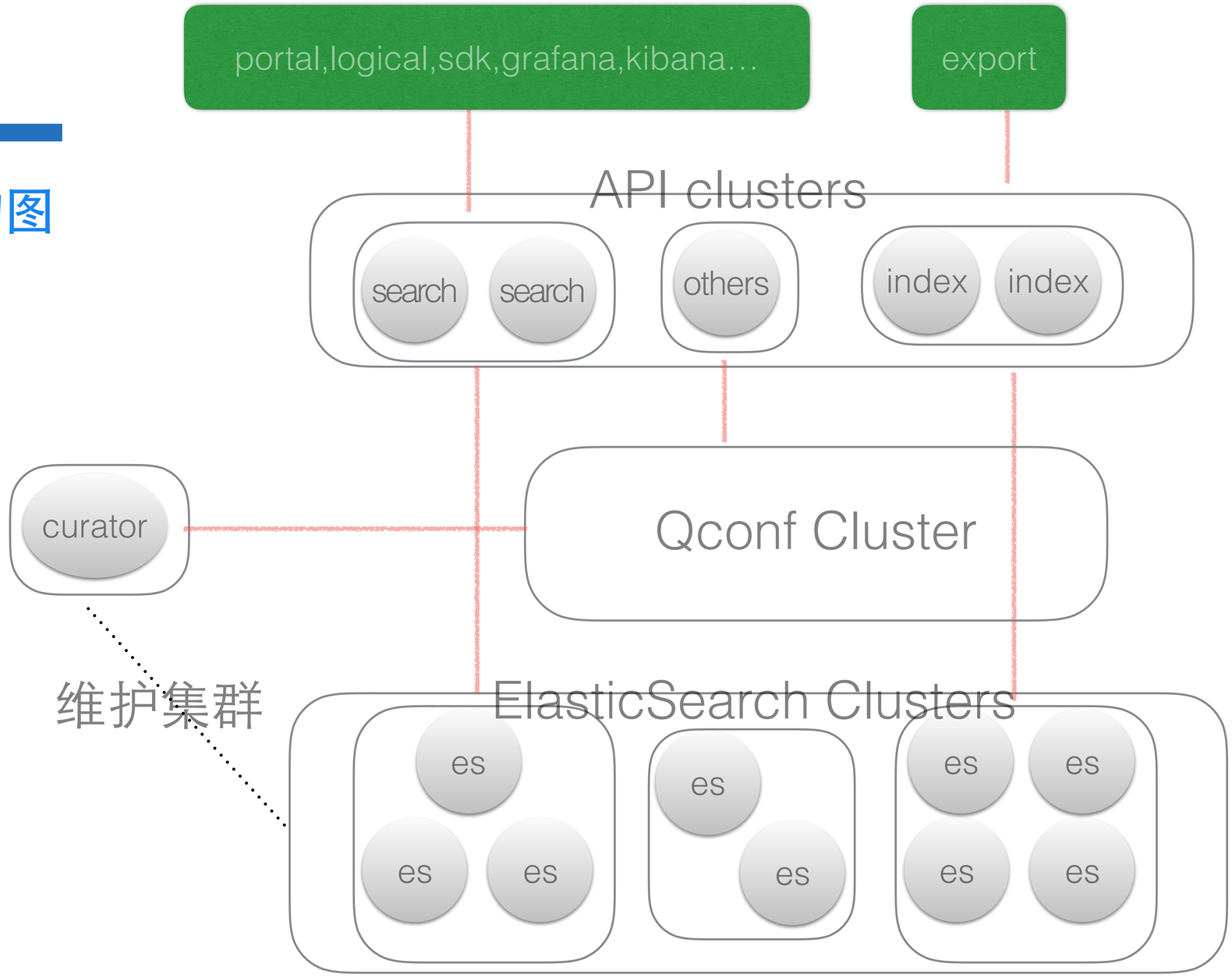
挑战1

- metadata查询压力过大，mongo遇到瓶颈

qconf 多级缓存系统



系统架构图



挑战2

- LAG

写点优化

- Benchmark
- Index程序
- Producer-多租户数据传输系统

数据传输思考

- 传输的上游拉取速度是稳定的
- 传输的下游消费速度是稳定的
- 传输的速度仅受限于上下游的影响
- 吞吐量=并发数*并发大小
- 整体吞吐量=拉取吞吐量+链路效率+推送吞吐量

链路耗损最严重

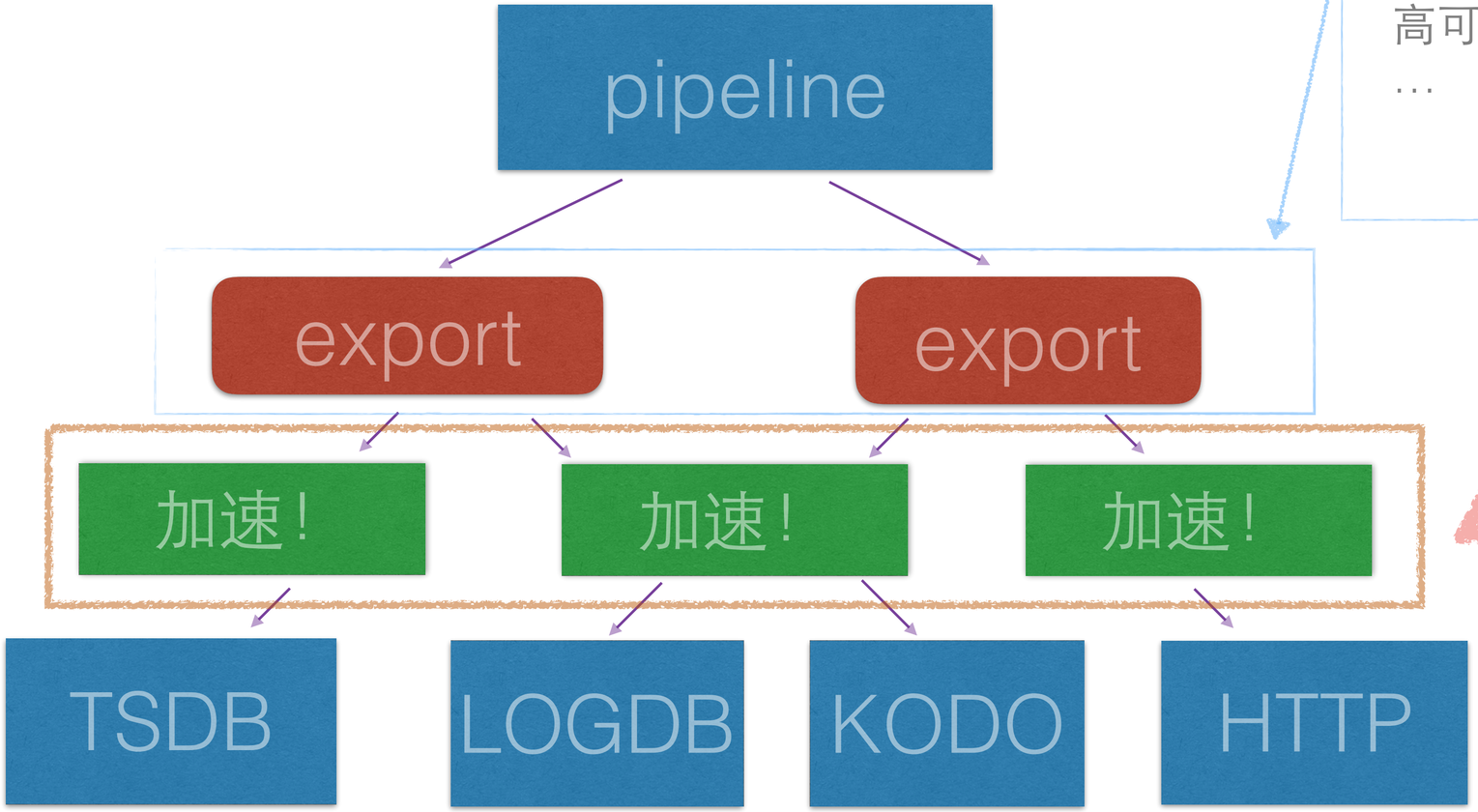
- 搞定流量：10k/s。
- 10k*3 驱动拉取（kafka）吞吐量，20k*5驱动推送（es）吞吐
- 针对logstash的架构，配置20k*5的并发度，真的能解决问题吗？我们需要更多的并发，因为整个数据链路会有损耗，导致毛刺，导致链路吞吐量上不去。may be 20k*8

可能想到的思路

- 上下游解耦：拉取与推送解耦，数据预取、队列暂存、拉取与发送并行
- 任务分割：大任务分解成小任务，小任务水平扩展
- 任务标准化：每个任务承载固定的流量，流量增加则增加任务数量
- 提升资源利用率：调度、平衡，压榨机器性能
- 提供任务管理能力：运维、运营、监控
- 更懂下游

数据导出、格式转换、过滤
精细化调度管理
轻量级分布式goroutine
高可用保证
…

构建加速系统



why not flume

```
        2. 日志类的repo支撑三种数据流向
    */
    if !userRepo.NotLogType() && userRepo.FlumeInfo.Status == FlumeTaskStatusStart {
        dataflow = DataFlowType.Flume
        var client *proxy.FlumeClient
        client, err = apiService.getFlumeClient(appID, repoName)
        if err != nil {
            err = errors.Info(ErrServiceUnavailable)
            return
        }
        err = client.SendLog(xl, checkedDocs)
        if err != nil {
            err = errors.Info(ErrServiceUnavailable).Detail(fmt.Errorf("appID:[%v],repo[%v] send
            indexResponse.FailedCount = len(checkedDocs)
            indexResponse.SuccessCount = 0
            indexResponse.TotalCount = len(checkedDocs)
            return
        }
        indexResponse.FailedCount = 0
        indexResponse.SuccessCount = len(checkedDocs)
        indexResponse.TotalCount = len(checkedDocs)
    } else if !userRepo.NotLogType() && userRepo.ProducerInfo.Status == model.ProducerTaskStatusSt
```

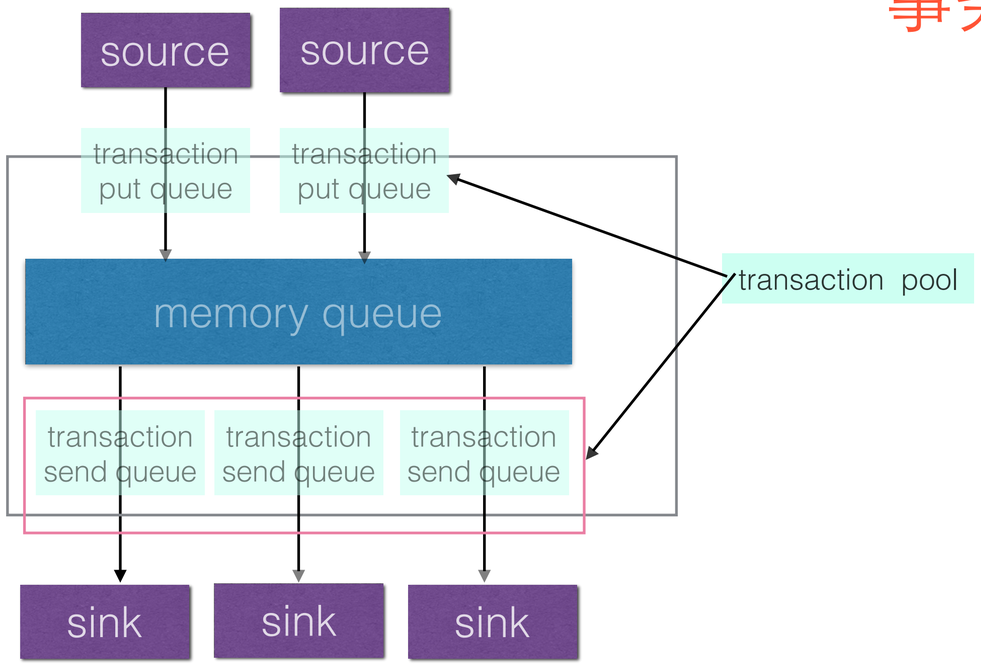
all-log.properties 108 KB

Raw | Blame | History | Permalink | Edit | Replace | Delete

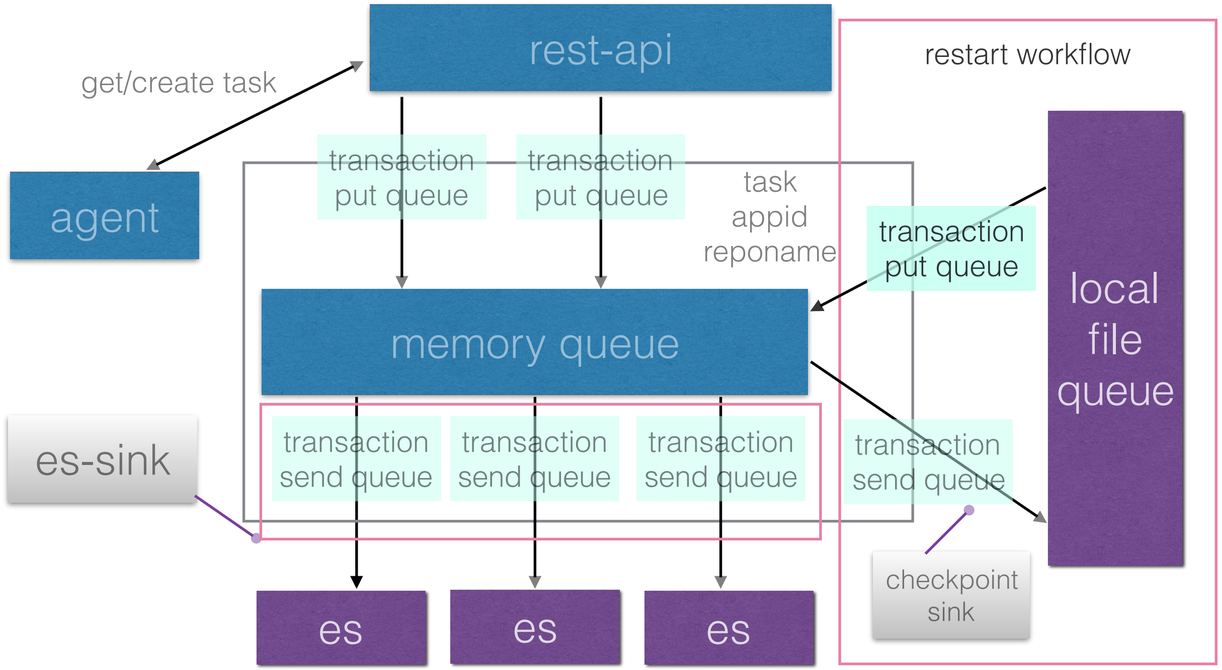
```
all-log.sources = k1 k2 k3 k4 k5 k6 k7 k8 k9 k10 k11 k12 k13 k14 k15 k16 k17 k18 k19 k20 k21 k22 k23 k24 k31 k32 k33 k34 k35 k36 k37 k38 k39 k40 k41 k42 k43 k44 k
all-log.channels = ch-null ch-all
all-log.sinks = sink-null-1 sink-null-2 es-dora-index1-1 es-dora-index1-2 es-dora-index1-3 es-dora-index1-4 es-dora-index2-1 es-dora-index2-2 es-dora-index2-3 es-
#channel
all-log.channels.ch-null.type = memory
all-log.channels.ch-null.capacity = 3000000
all-log.channels.ch-null.transactionCapacity = 30000
##storage
all-log.channels.ch-all.type = memory
all-log.channels.ch-all.capacity = 3600000
all-log.channels.ch-all.transactionCapacity = 15000
#sink
all-log.sinks.sink-null-1.type = null
all-log.sinks.sink-null-1.channel = ch-null
#sink
all-log.sinks.sink-null-2.type = null
all-log.sinks.sink-null-2.channel = ch-null

#sink  es-dora-index1
all-log.sinks.es-dora-index1-1.channel = ch-all
all-log.sinks.es-dora-index1-1.client= transport
all-log.sinks.es-dora-index1-1.type = com.frontier45.flume.sink.elasticsearch2.ElasticSearchSink
all-log.sinks.es-dora-index1-1.hostNames = nb1820:9300,nb1820:9301,nb1821:9300,nb1821:9301,nb1885:9300,nb1885:9301,nb1885:9302,nb1886:9300,nb1886:9301,nb1886:9302
all-log.sinks.es-dora-index1-1.indexName = inner-dora-1
all-log.sinks.es-dora-index1-1.indexType = inner
all-log.sinks.es-dora-index1-1.clusterName = qiniues-2
all-log.sinks.es-dora-index1-1.batchSize = 15000
all-log.sinks.es-dora-index1-1.serializer = com.frontier45.flume.sink.elasticsearch2.ElasticSearchDynamicSerializer
```

Producer 模型

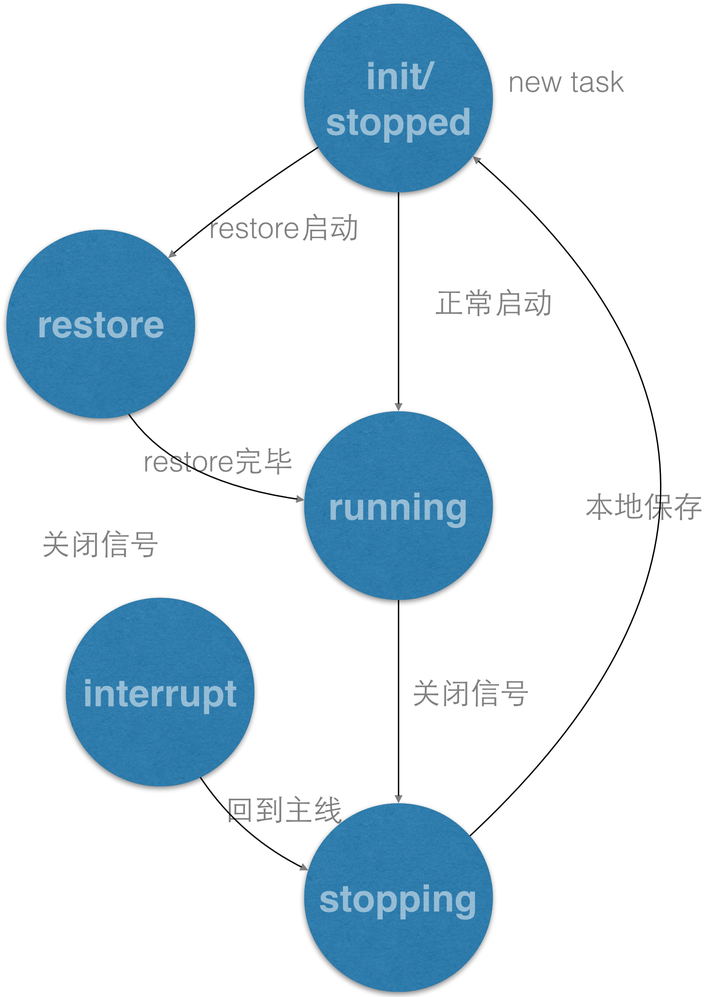
事务



Producer 架构图



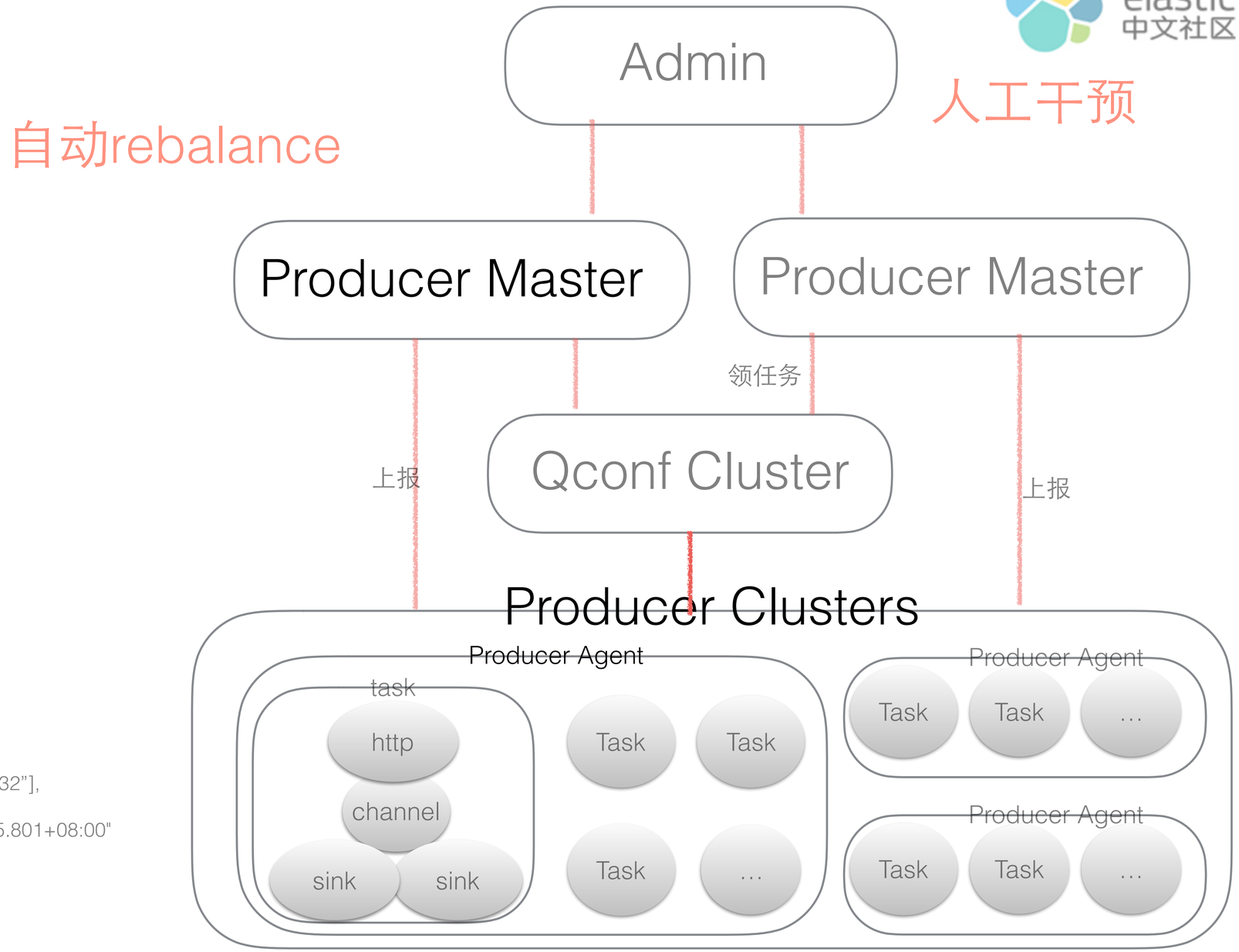
Producer状态机



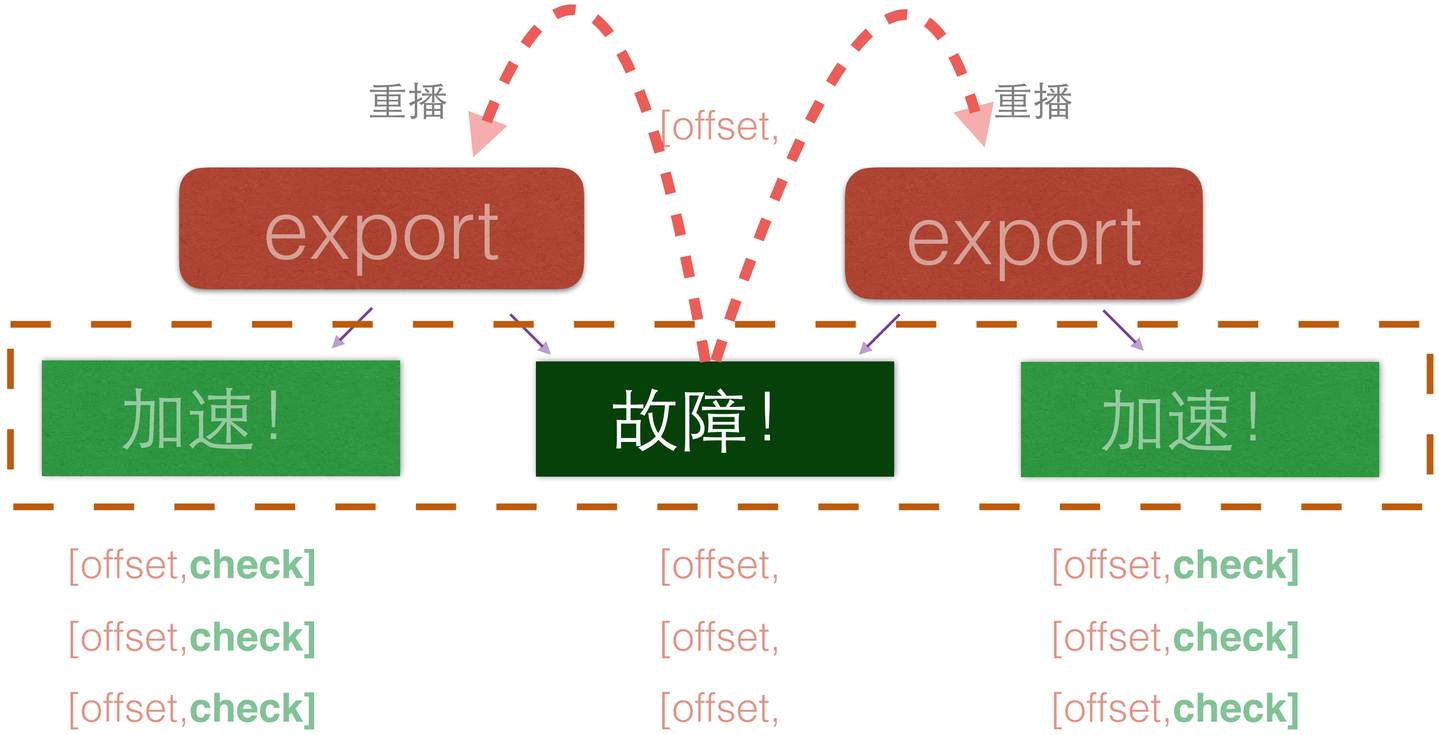
task状态机

Producer 分布式

```
{
    "status": 1,
    "custom": 1,
    "batchSize": 20000,
    "taskSize": 10,
    "concurrency": 3,
    "capacity": 1000000,
    "transactionCapacity": 20000,
    "hosts": [“nb2088:2934”,”nb2088:3232”],
    "updateTime": "2016-11-10T17:52:05.801+08:00"
}
```



数据重播



挑战3

❖ 大量查询超时

查询优化

- 搜索体验随着shard增多而恶化
- query的多样性和复杂性
- 日志query优化

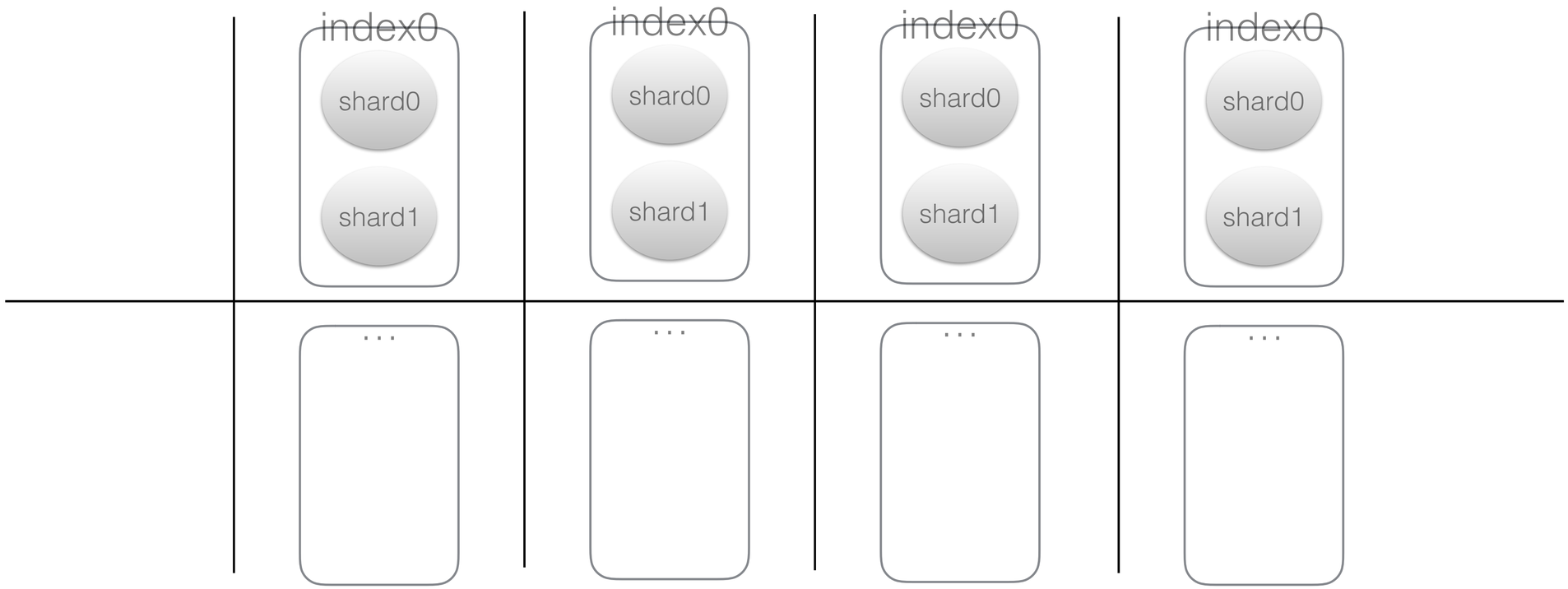
查询优化

条件满足预判

Query Executor

二级时间索引

执行计划可控



挑战4

- 24点噩梦
- GC长时间停顿，集群阻塞，Unavailable

索引预创建

❖ 提前一天平滑预创建

Shard 编排

Shard 标准化

- Shard不是免费的
- 写点
- 查询
- 集群管理

Shard编排(rebalance)

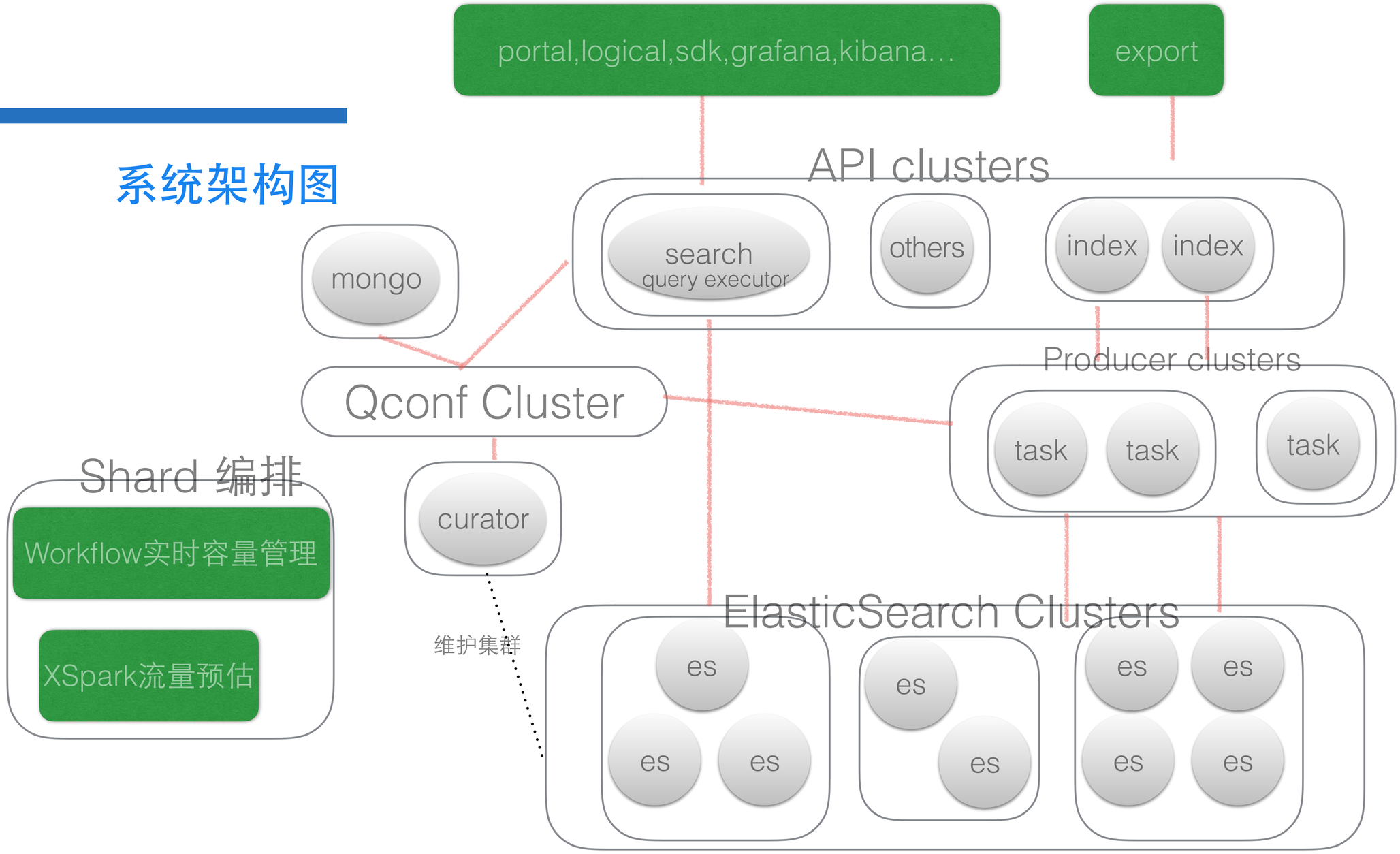
现状

- ElasticSearch自带rebalance算法有局限性
- 弹性的用户的打点流量
- 弹性的用户的查询QPS

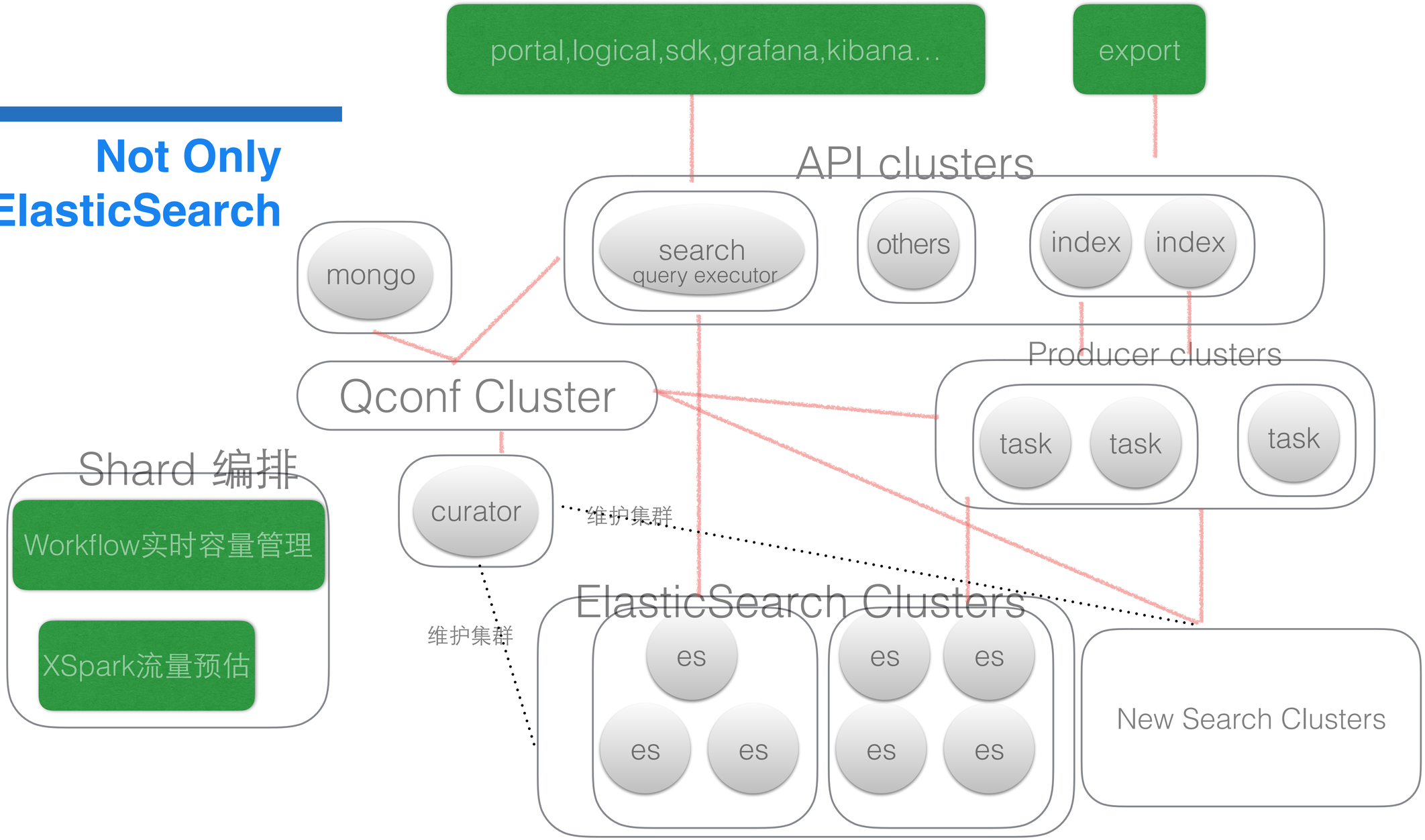
Shard编排(rebalance)

- 冷启动
- 基于机器学习算法的流量预估
- 基于Pandora Workflow实时动态扩缩容
- 稳定性策略

系统架构图


portal,logical,sdk,grafana,kibana…
export
API clusters
search
query executor
others
index
index
mongo
Qconf Cluster
Producer clusters
task
task
task
Shard 编排
Workflow实时容量管理
XSpark流量预估
curator
维护集群
ElasticSearch Clusters
es
es
es
es
es
es
es
es
es
es

Not Only ElasticSearch



成果总结

- 支撑海量用户
- 支撑每天150T+,2000亿+的增量数据
- 没有lag
- 查询秒级返回
- 接近0运维
- 可用性99.9%

谢谢！