简单 · 可信赖

基于ElasticSearch构建搜索云服务实战

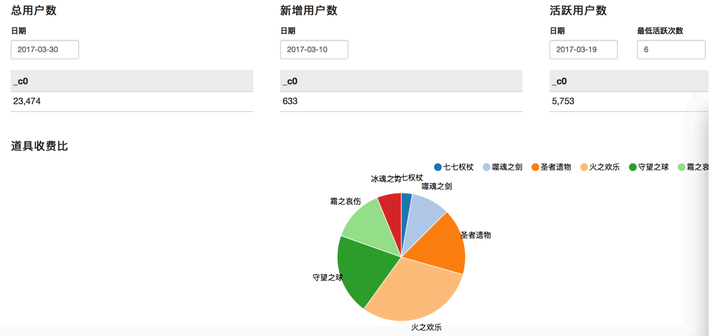
王拓

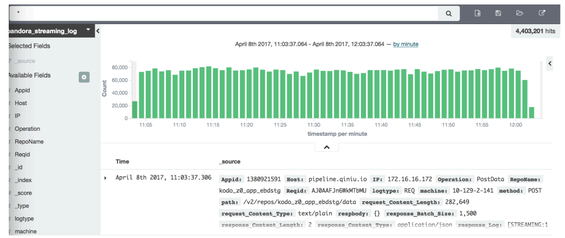
主要内容

- 背景
- 设计目标
- 遇到的挑战
- 如何应对
- 成果总结

Pandora 大数据平台

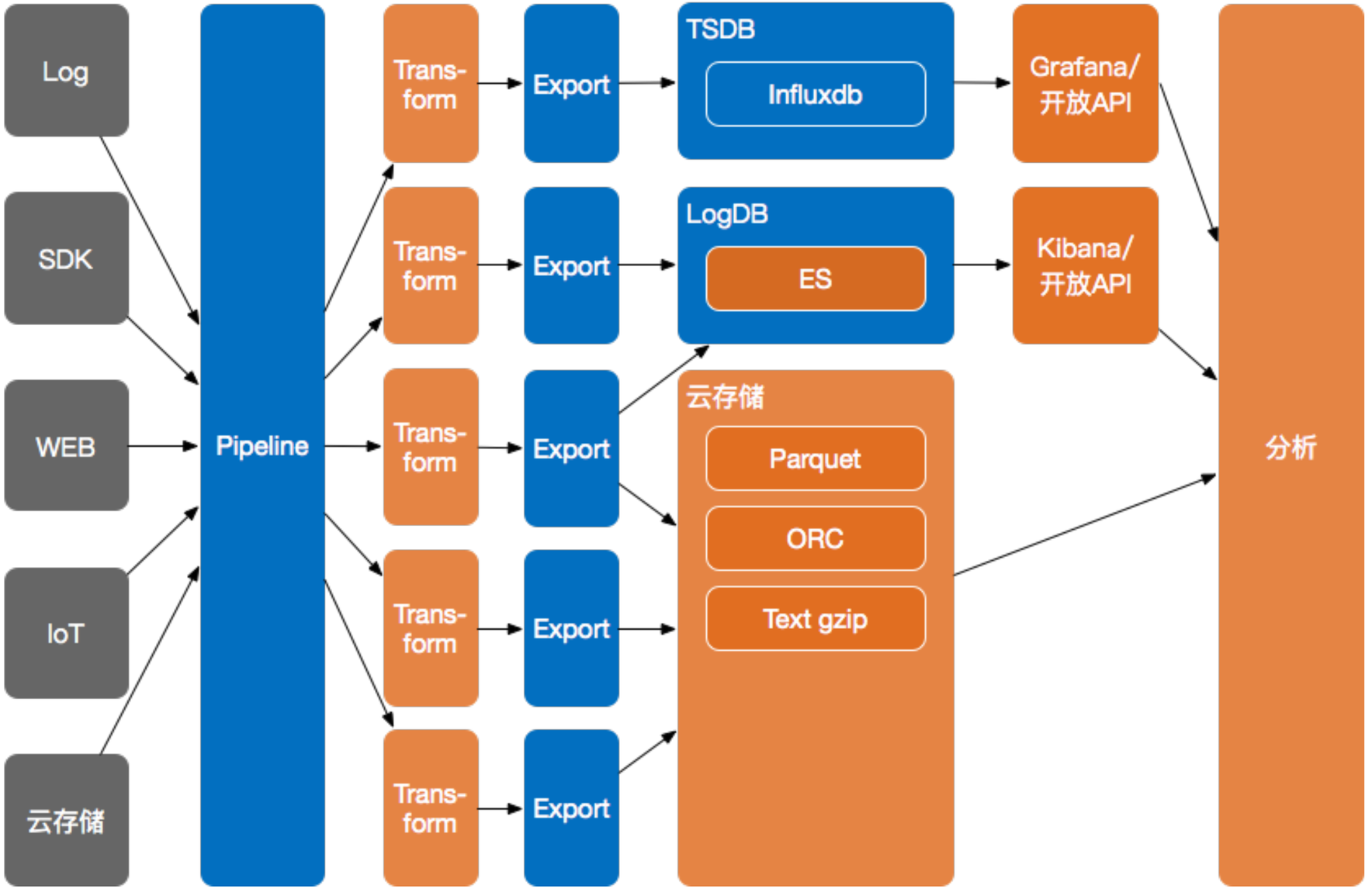
背景





Pandora 系统架构图

背景



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

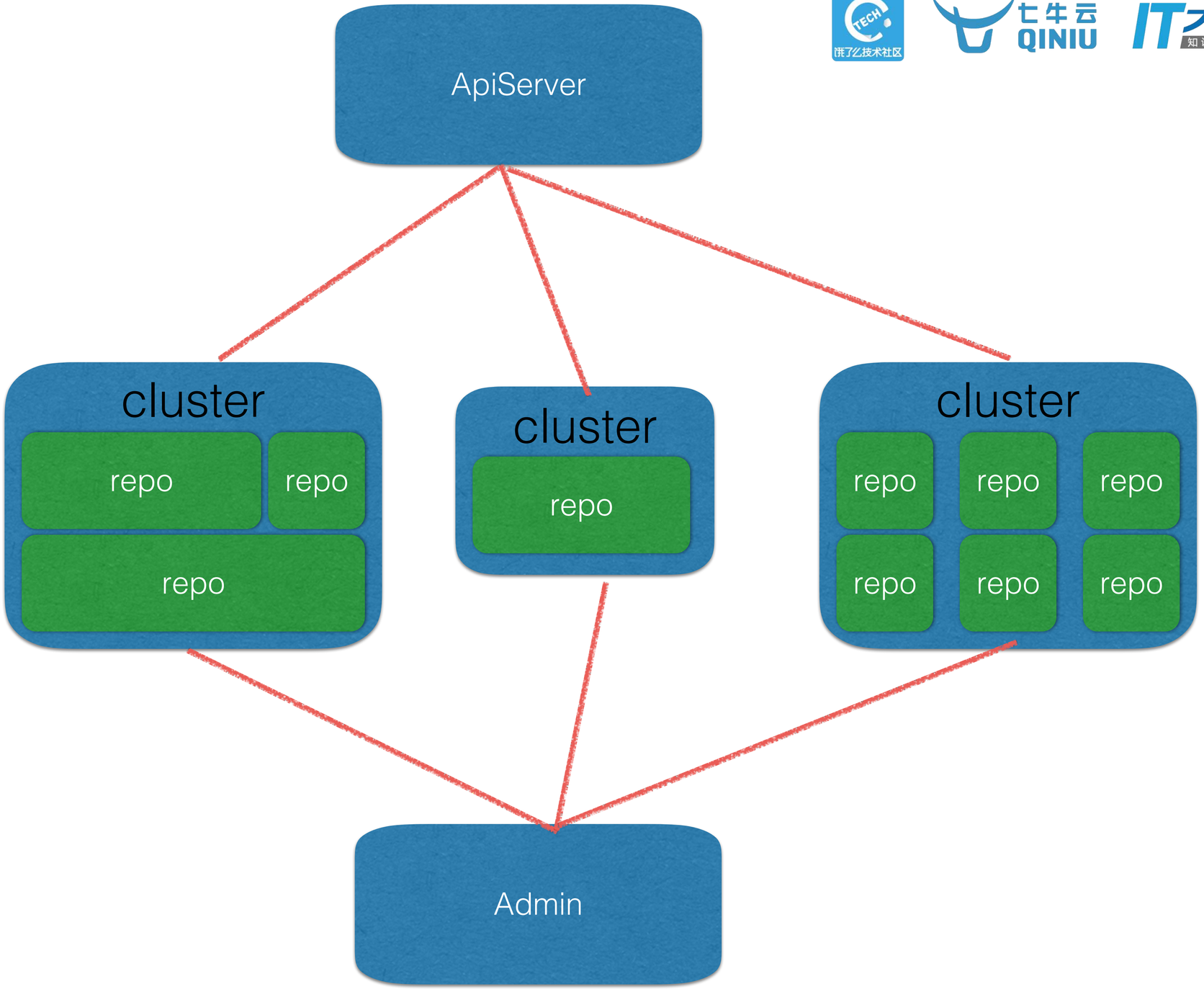
设计目标

LogDB 设计目标是什么？

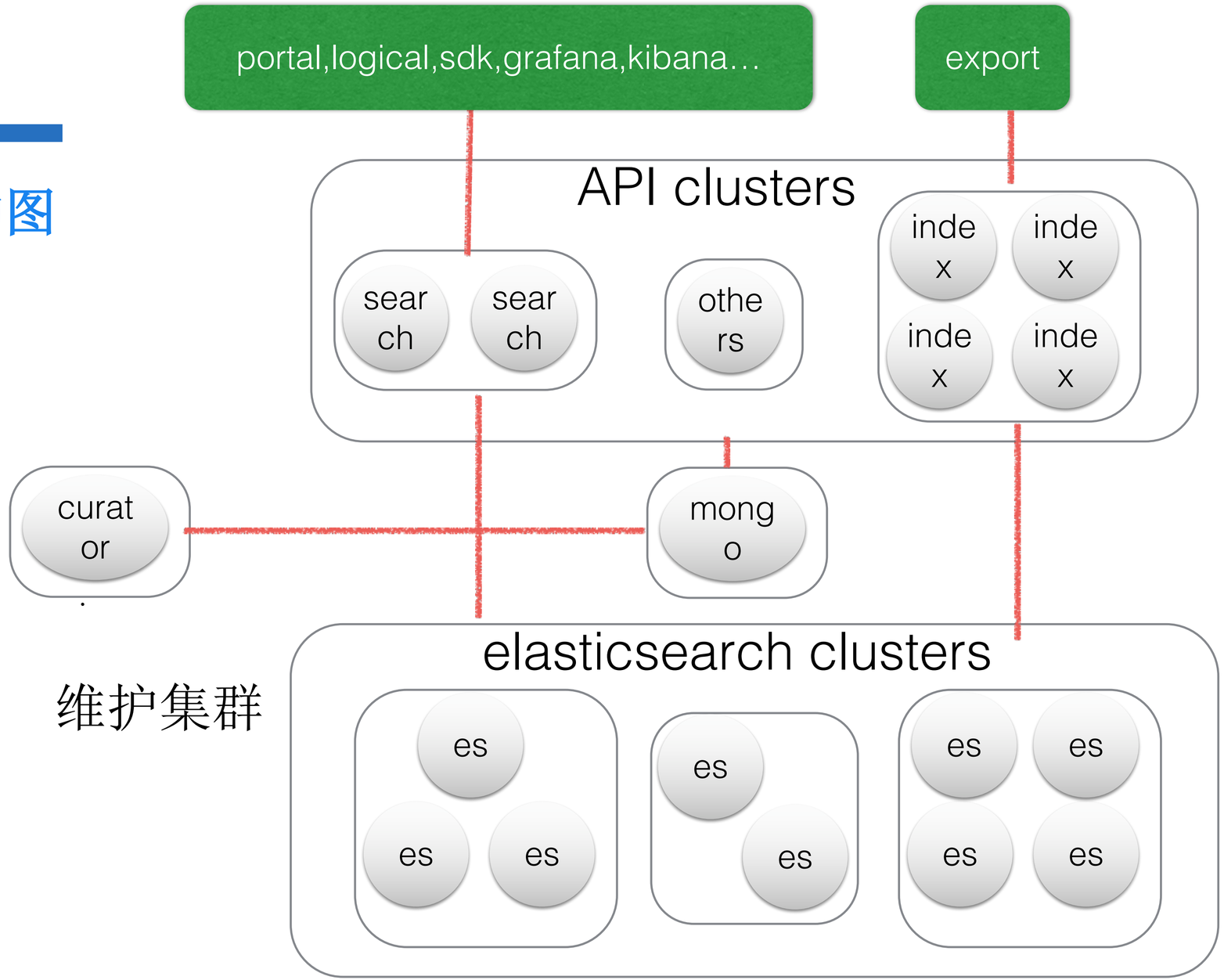
- 支持公有云海量用户
- 支持单个用户日志规模MB-100TB/天
- 查询秒级响应
- 可靠性
- 拥抱开源，可以适配kibana，grafana等

多租户模型

海量cluster



系统架构图

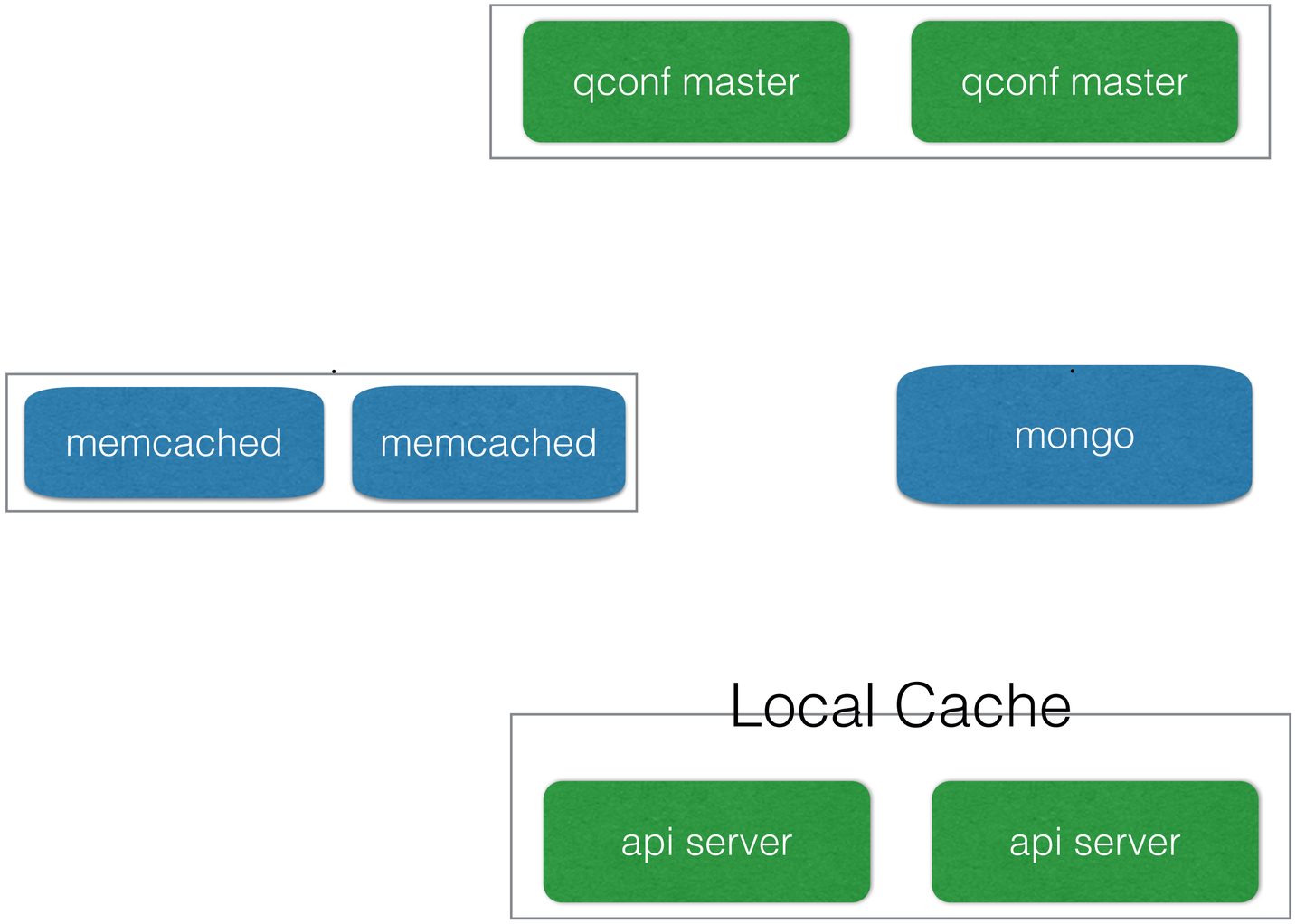


维护集群

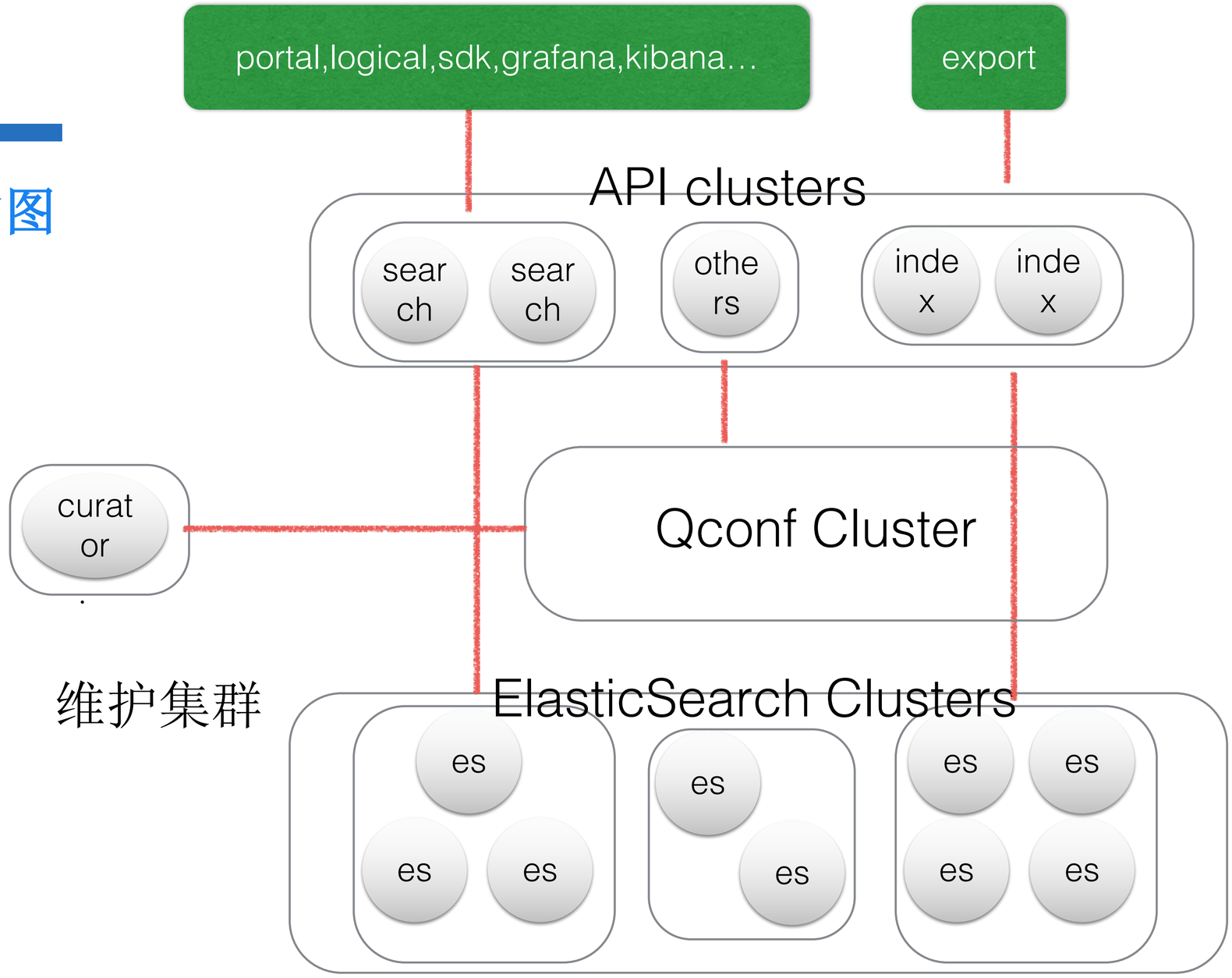
遇到的挑战

- **metadata查询压力过大，mongo遇到瓶颈**

qconf 多级缓存系统



系统架构图


portal,logical,sdk,grafana,kibana…
export
API clusters
sear ch
sear ch
othe rs
inde x
inde x
curat or
Qconf Cluster
维护集群
ElasticSearch Clusters
es
es
es
es
es
es
es
es
es
es

遇到的挑战

- 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

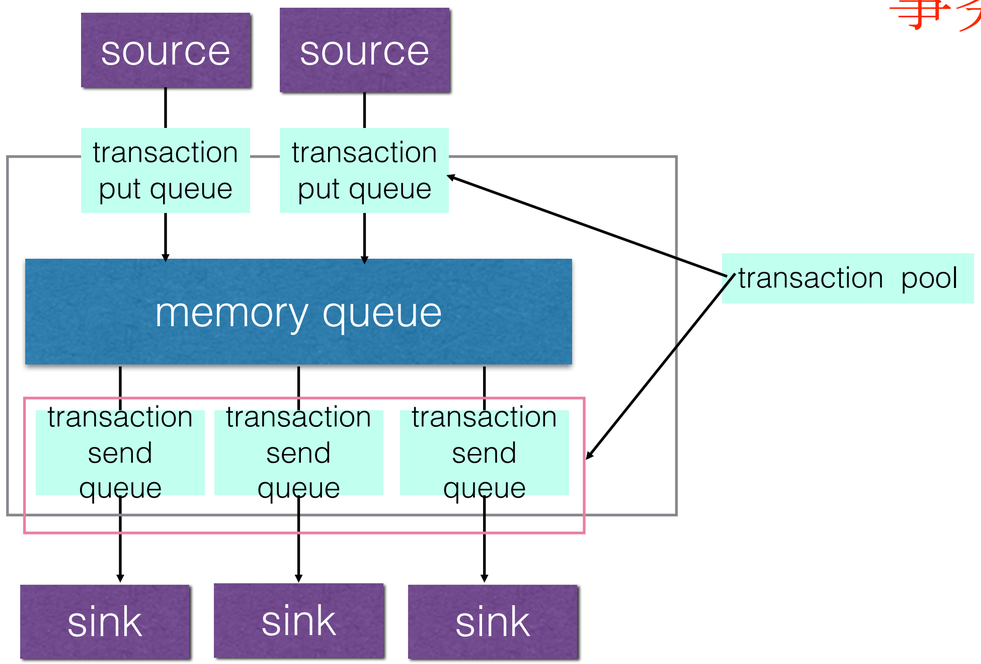
- 传输的上游拉取速度是稳定的
- 传输的下游消费速度是稳定的
- 传输的速度仅受限于上下游的影响
- 吞吐量=并发数*并发大小
- 整体吞吐量=拉取吞吐量+链路效率+推送吞吐量

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

Producer 模型

事务



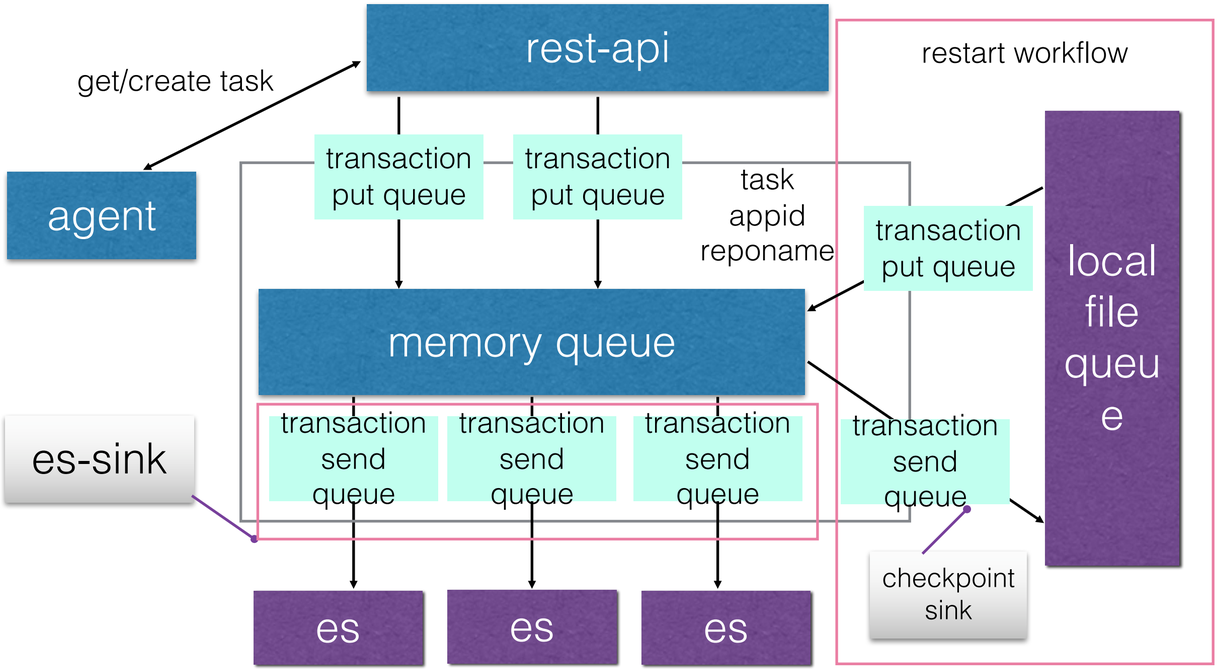
Producer 模型

```
if transaction == nil {
    transaction = abs.channel.GetTransaction()
}
defer func() {
    if err != nil {
        nerr := transaction.RollBack()
        if nerr != nil {
            resTransaction = transaction
            return
        }
    }
    transaction.Close()
}()
transaction.Begin()
var events []model.ProducerEvent
if transaction.GetQueueIsFull() {
    events = transaction.CopyGetQueue()
} else {
    for i := 0; i < abs.getBulkSize(); i++ {
        var event model.ProducerEvent
        event = transaction.GetEvent()
        if event == nil {
            break
        }
        events = append(events, event)
    }
}

err = abs.doProcess(events, abs.SCounter)
```

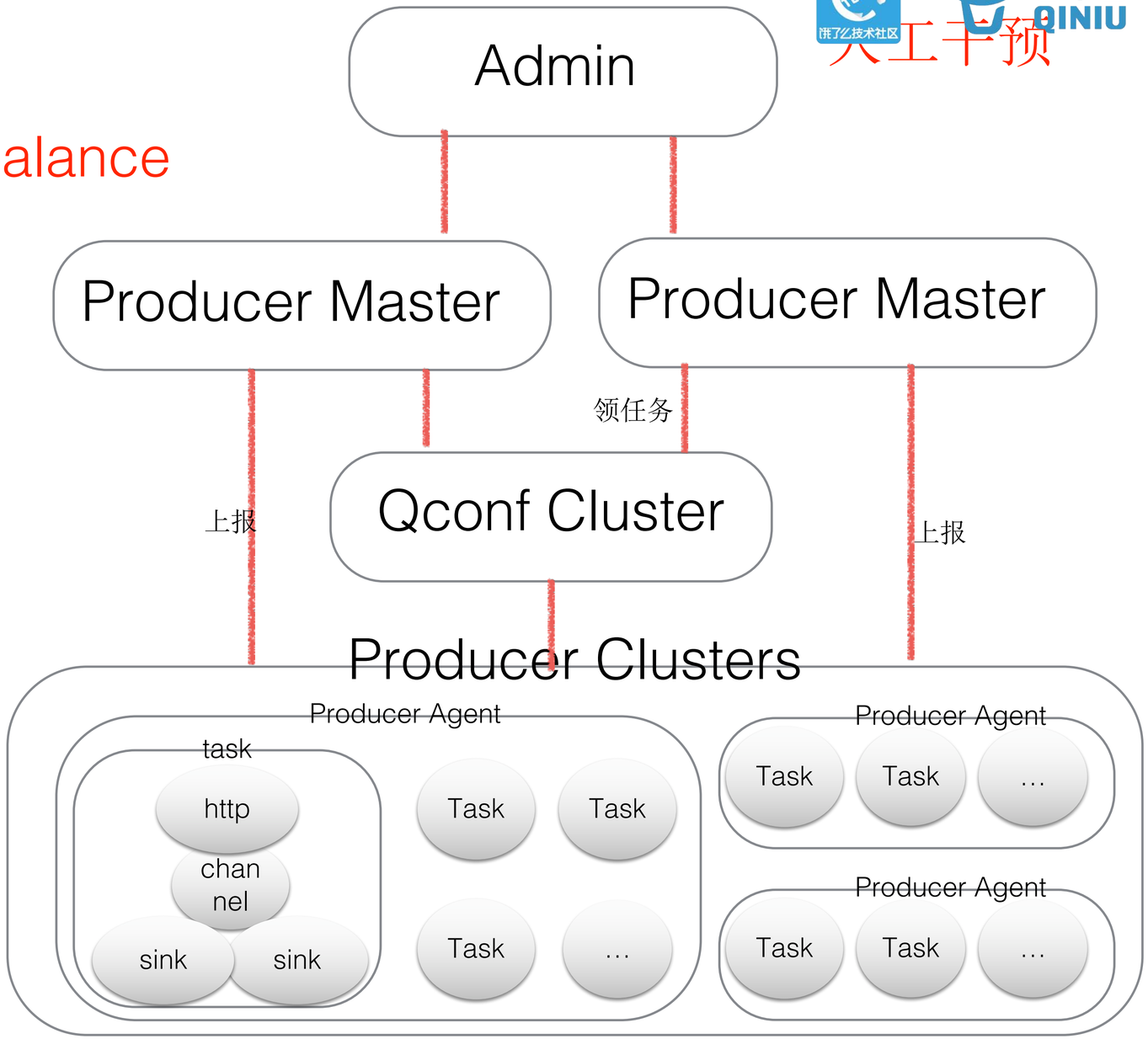
```
transaction := task.channel.GetTransaction()
transaction.Begin()
defer func() {
    if err != nil {
        transaction.RollBack()
    }
    transaction.Close()
}()
for _, v := range bulkRequest.Body {
    err = transaction.PutEvent(v)
    if err != nil {
        return
    }
}
err = transaction.Commit()
```

Producer 架构图



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"
}
```



遇到的挑战

- 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
- Benchmark
- 日志Query优化
- 控制

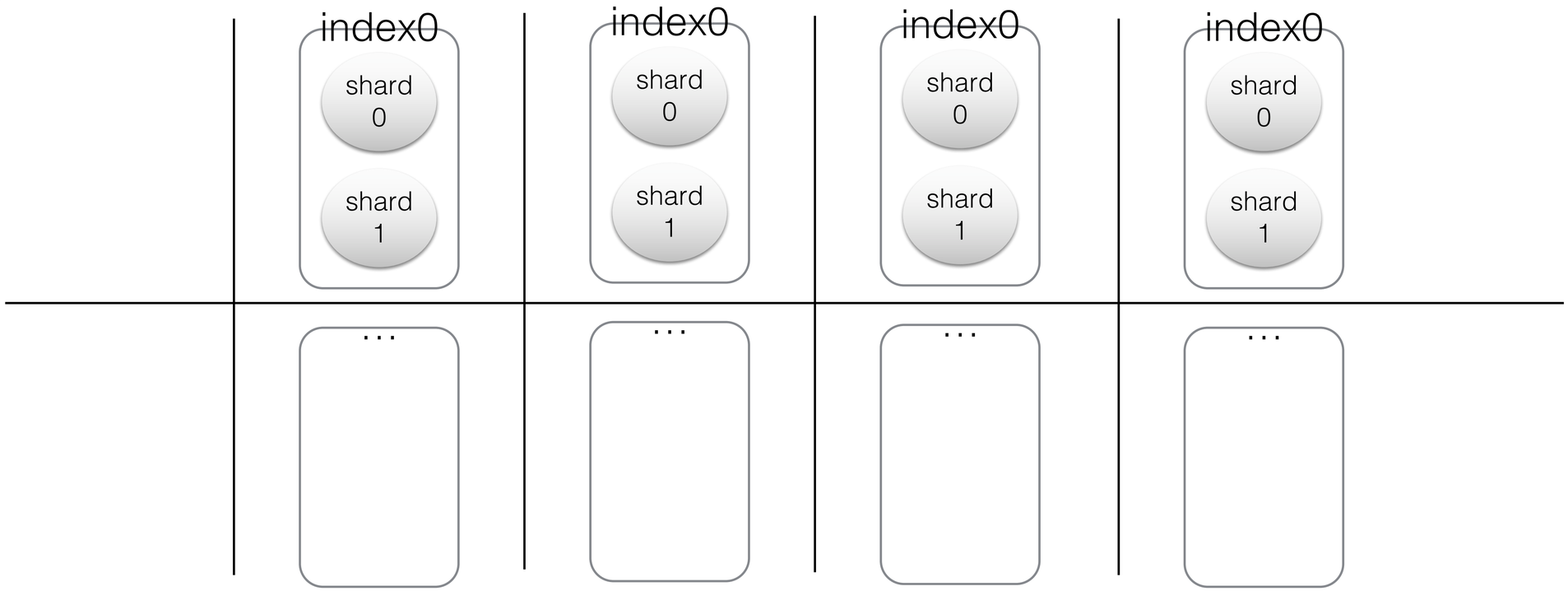
查询优化

条件满足预判

Query Executor

二级时间索引

执行计划可控



遇到的挑战

- 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)

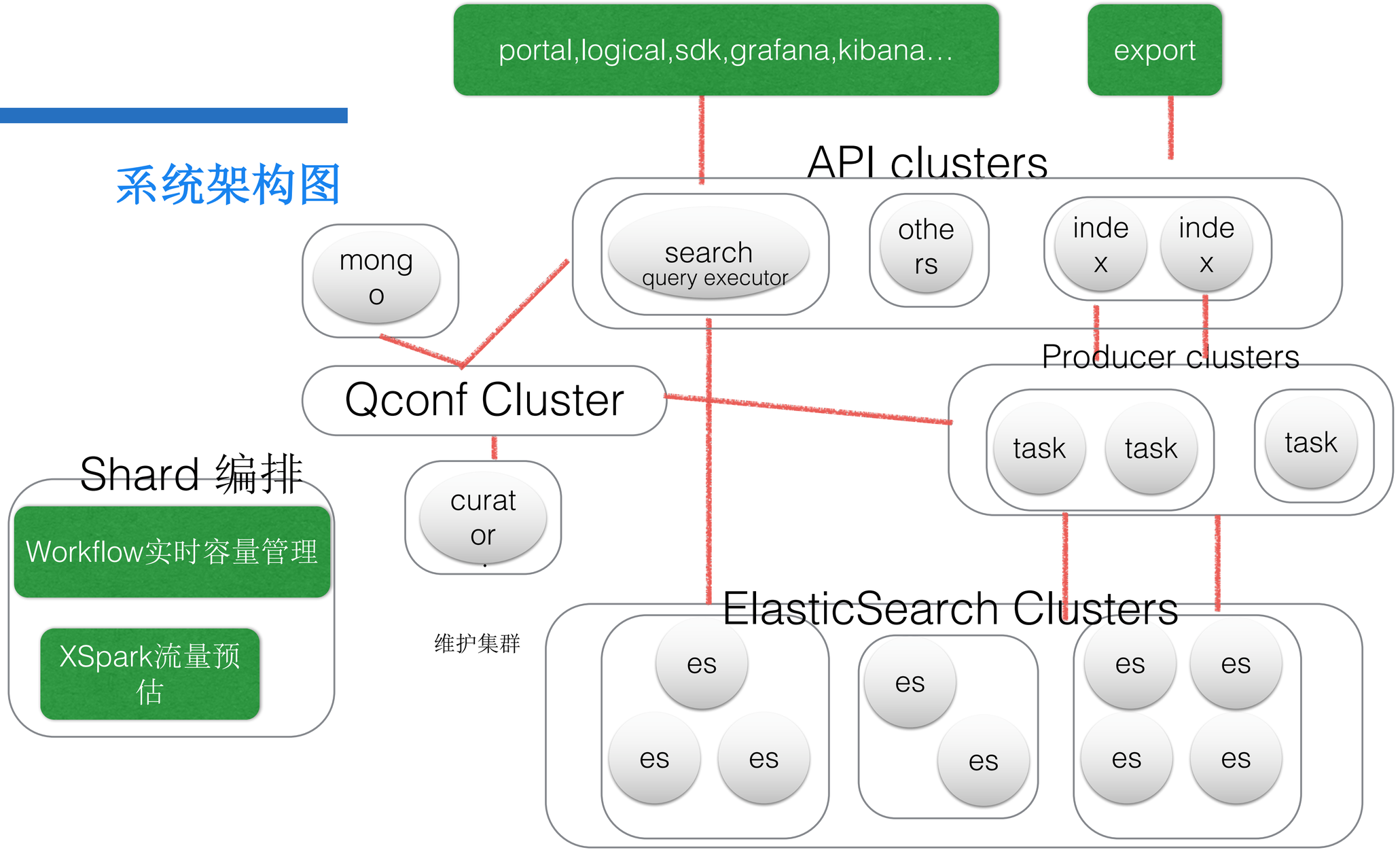
现状

- 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

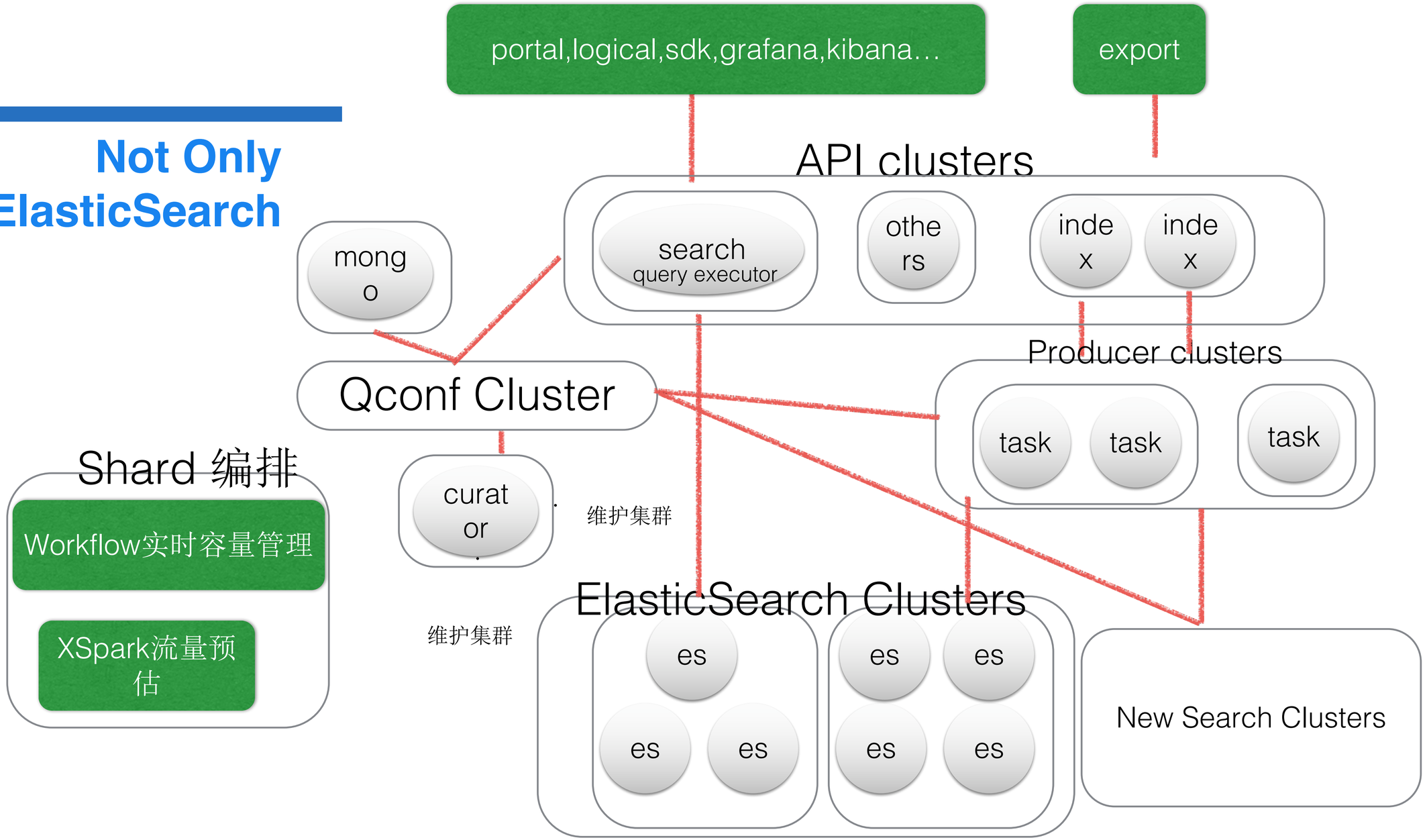
Shard编排(rebalance)

- 冷启动
- 基于离线XSpark的REPO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
othe
rs
inde
x
inde
x
mong
o
Qconf Cluster
Producer clusters
task
task
task
curat
or
Shard 编排
Workflow实时容量管理
XSpark流量预
估
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100T+,2000亿+的流量
- 没有lag
- 查询秒级返回
- 接近0运维
- 可用性99.9%

谢谢！