腾讯万亿级Elasticsearch架构实践

姜国强/腾讯存储专家

目录

01

简介

Elasticsearch(ES)：高性能、分布式的搜索分析引擎

搜索引擎

App搜索、站内搜索、电商搜索

高性能倒排索引、中文分词

可观测性

日志、Metric、APM

完整解决方案：采集–存储–可视化

安全检测

SIEM、Endpoint Security

统一的安全防御/检测：集中式 + 终端

生态发展迅速

DB-Engines

6年成为Rank 7

搜索引擎Rank 1

开源实时日志首选

开源状态

51k+ Star

17k+ Fork

月活跃贡献者90+

应用状态

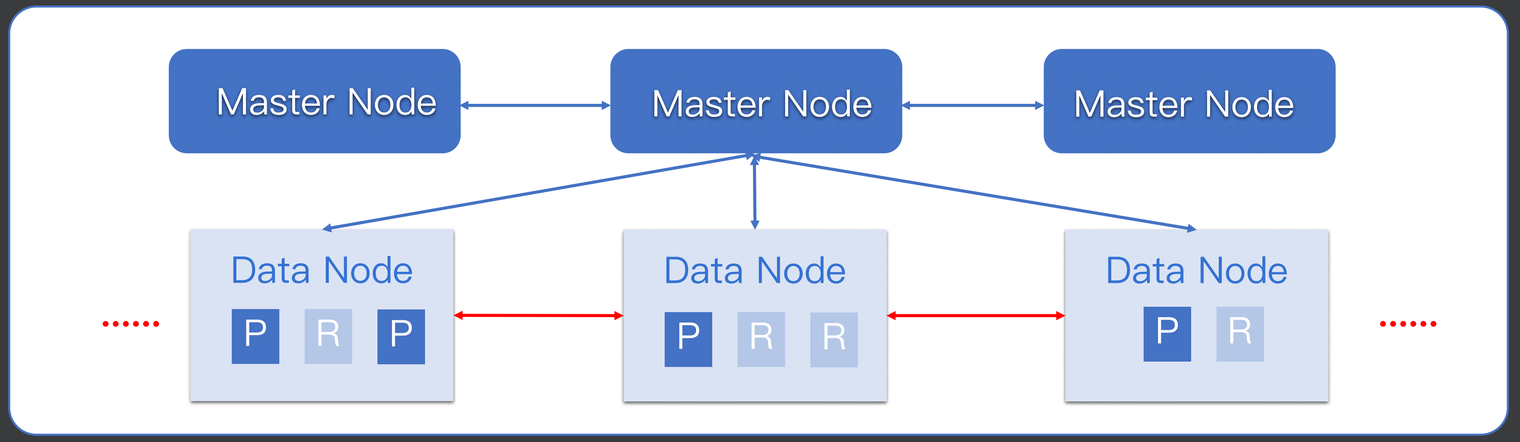
累积4+亿次

2019增长1亿次

多云厂商合作

简介 — 系统架构

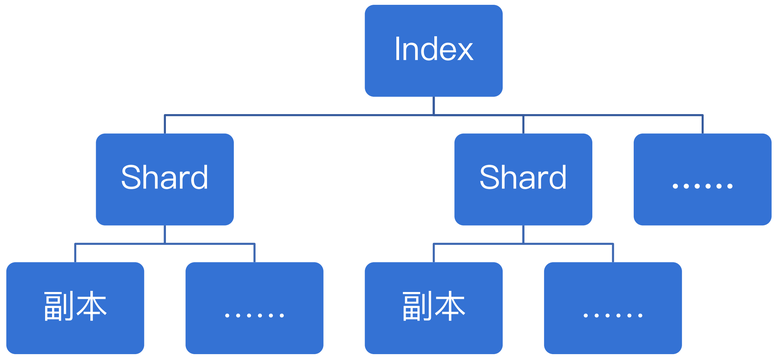
集群架构



- **Master Node：**
  - 集群管控：机器、数据
  - High Available：类Raft
- **Data Node：**
  - 数据存储：Lucene
  - 数据访问：两层汇聚
  - 线性扩展

逻辑数据模型

title	content	star
es book	hello es	30
lucene book	hello lucene	



- 文档型存储：Free Schema
- **Index：类RMDB表**
- **Shard：**
  - 数据分片，存储数据
  - 分布式打散
- **副本：动态可控**

简介 — 系统架构

物理数据模型

- 写入

```
POST /shop
{
  "title": "es book",
  "content": "hello es",
  "star": 30
}

POST /shop
{
  "title": "lucene book",
  "content": "hello lucene"
}
```

- 查询

```
GET /shop/_search
{
  "query": {
    "query_string": {
      "query": "title:book AND content:es"
    }
  }
}
```

倒排索引：用于条件过滤

book	1,2
es	1
lucene	2

字段倒排索引–title

hello	1,2
book	1,2
es	1
lucene	2

content字段倒排索引

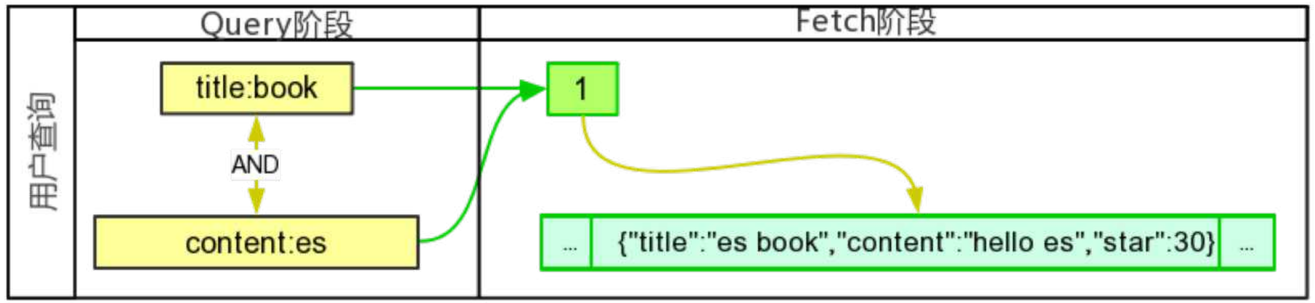
行存储：存储原始文档

{“title":"es book","content":"hello es",
"star":30}{"title":"lucene book",
"content":"hello lucene"}

列存储：用于排序、聚合

"es book", ”lucene book", "hello es",
“hello lucene”, 30,

LSM Tree – Segment



简介 — 腾讯应用现状

丰富的应用环境

公有云

丰富的场景

大量中小客户
不同的应用场景
不同的熟悉程度

内部云

超大规模

千级节点
千万级写入
十万级查询

私有云

标准化/自动化

完全隔离的环境
标准化交付
自动化运维

简介 — 腾讯应用现状 — 搜索

典型代表

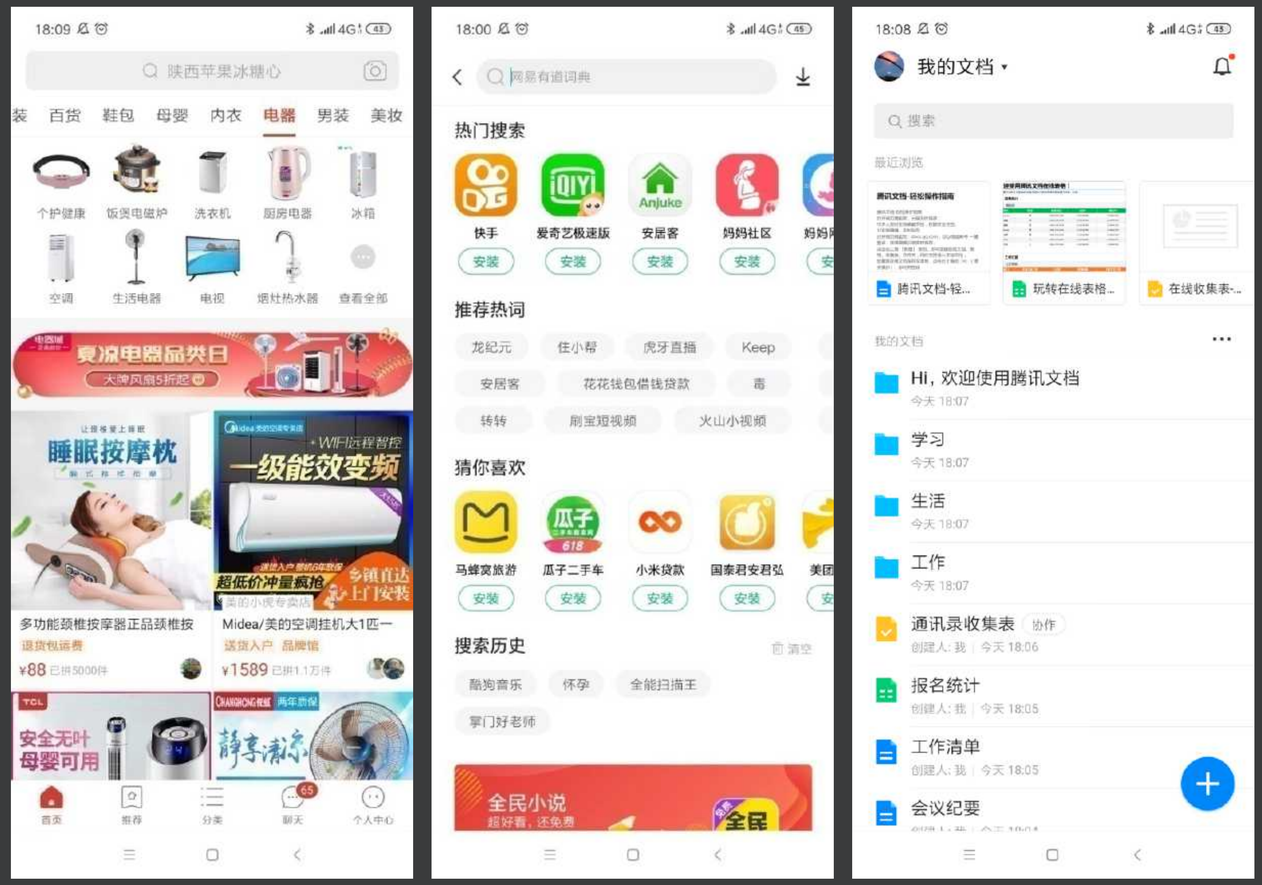
- 电商商品搜索
- 应用市场搜索
- 站内文档搜索
- ……

主要特点

- 高性能：10w级QPS，20ms级平响
- 强相关：搜索结果高度匹配用户意图
- 高可用：可用性达9999，跨机房容灾

关联：大搜索

- 轻量、垂直化的搜索解决方案



简介 — 腾讯应用现状 — 日志实时分析

典型代表

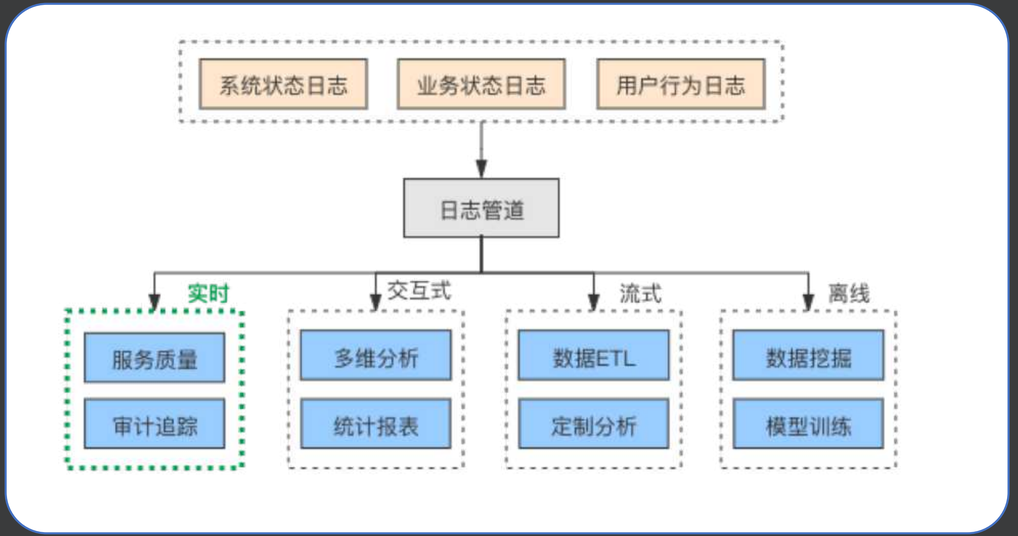
- 系统状态日志：服务质量
- 业务状态日志：运营分析
- 用户行为日志：用户画像分析、操作审计

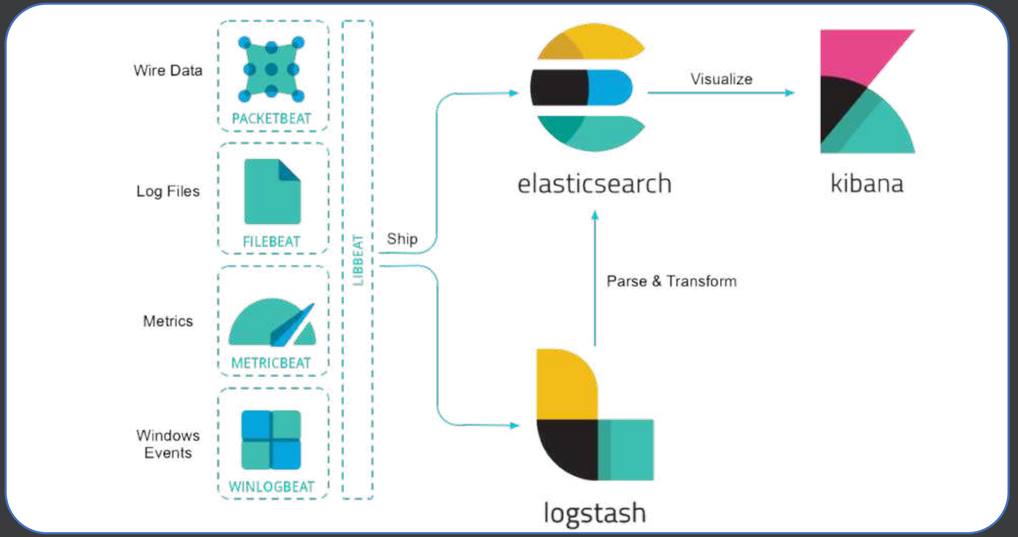
主要特点

- 时效性：从日志产生到可访问，十秒级
- 高性能：万亿级日志，秒级响应
- 灵活性：接口易用灵活，类搜索引擎

关联：大数据解决方案

- 解决延时高、性能差、使用复杂等痛点





简介 — 腾讯应用现状 — 时序数据处理

典型代表

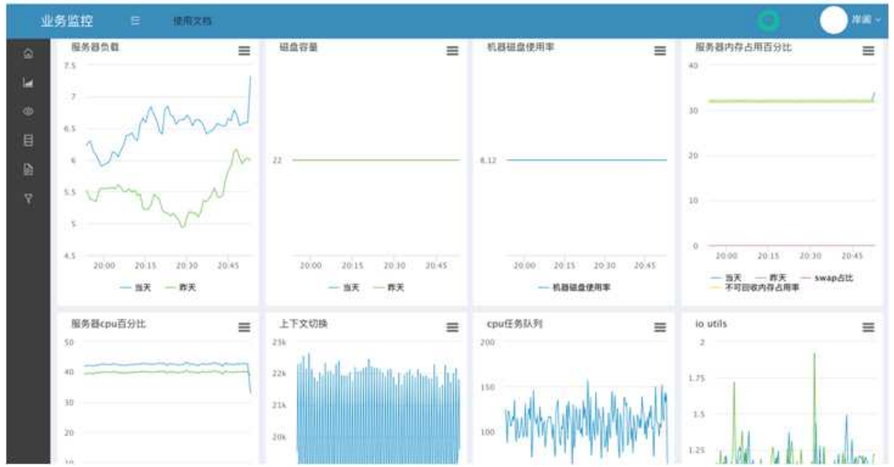
- Metrics
- APM
- 物联网数据

主要特点

- 高并发写入：千万级吞吐
- 高查询性能：十毫秒级
- 多维分析：灵活、多维度的统计分析

关联：专用型时序数据库

- 近似水平的压缩比、性能
- 统一的技术栈，更低的维护成本



目录

02

技术挑战 — 可用性

代表场景

- 搜索、日志、时序等
- 分布式系统共性问题

可用性状态(初期)

- 月可用性低：99%
- 运营压力大：2.5人力
- 用户首要诉求：可用性

社区状态

- ES：月提交400~，Bug 38
- Redis：月提交20~，Bug 2

问题分类

架构设计不足	扩展性瓶颈	健壮性低	负载不均
容灾方案欠缺	数据丢失	网络分区	
内核缺陷	边缘场景	新特性引入	

具体样例

- 数据分片限制
- 读写压力过载
- 集群多节点/盘不均

- 安全攻击
- 误操作
- 自然灾害

- Master任务饿死
- 均衡分布式死锁
- 链接异常

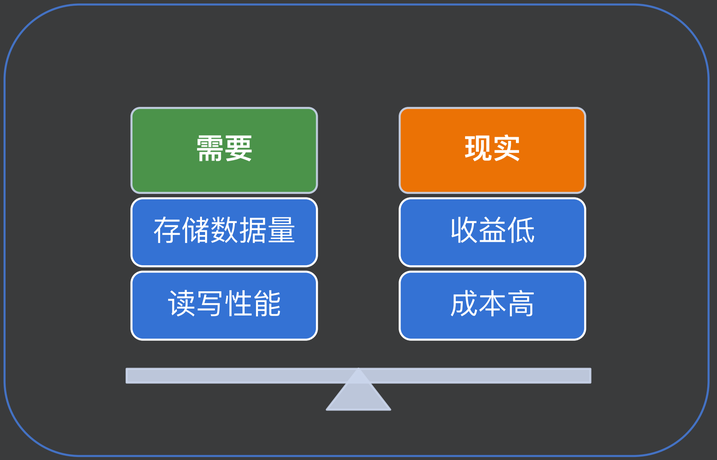
技术挑战 — 成本

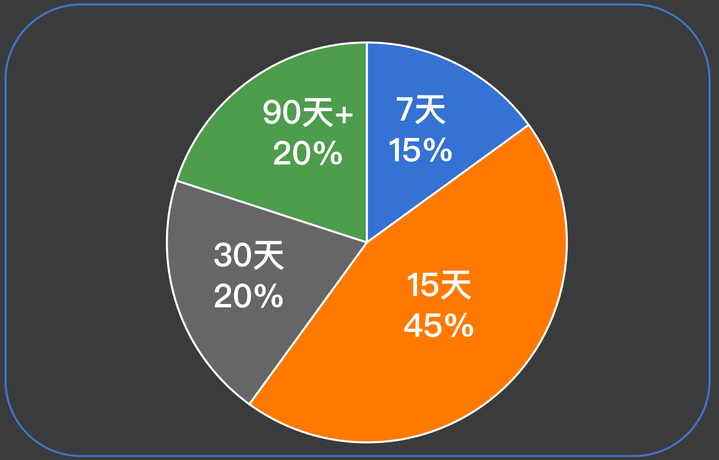
代表场景
- 日志实时分析
- 时序数据处理

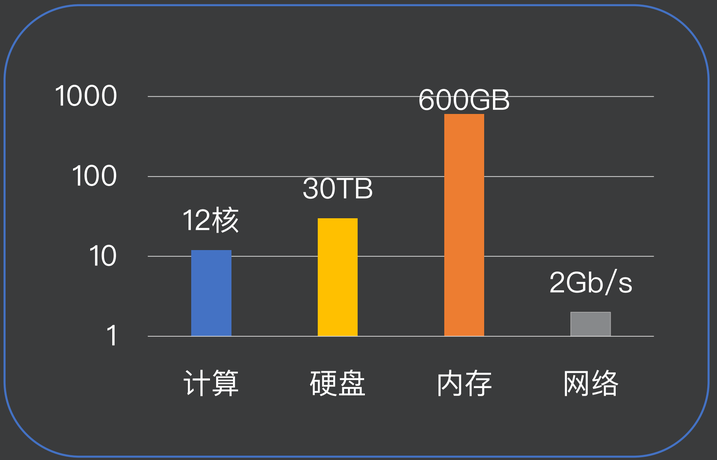
成本矛盾
- 海量数据：百万级TPS，PB级数据
- 业务价值：监控、审计等价值较低

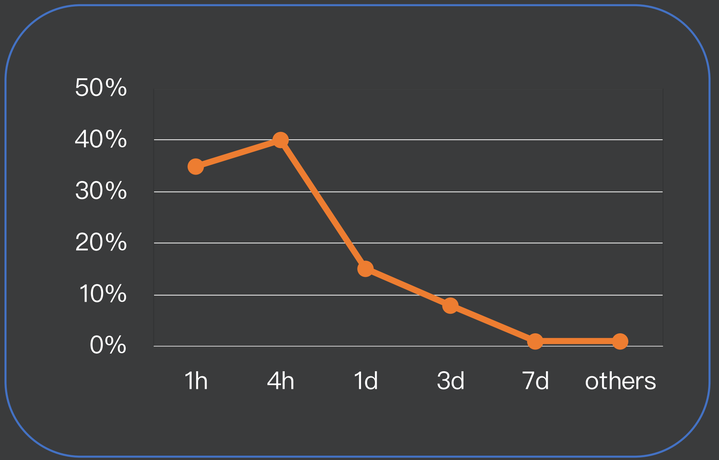
资源需求 (15天)
- **内存：** 矛盾严重
- **硬盘：** 保留时长加剧矛盾
- **计算：** 写入吞吐决定计算开销

使用分析
- 保留时长统计：Top 10大客户
- 冷热特性明显：访问近多远少 (15天)









技术挑战 — 性能

代表场景

- 搜索服务：电商、站内搜索等
- 时序处理：监控曲线

需求

- 平响：10ms级
- 毛刺：P99.5小于100ms
- 吞吐：10w级 QPS

矛盾

- 延时：平响100ms级，毛刺5s+
- 吞吐：千级，线性扩展差

指标	期望水平	矛盾
低平响	Avg 10ms级	平响不达标 随机IO、Scan、执行不优
低毛刺	P99.5 低于100ms	长尾严重 硬件异常、后台任务、GC
高吞吐	QPS 10w级	量级差距 单机能力、线性扩展

目录

03

可用性优化 — 解决方案

架构设计

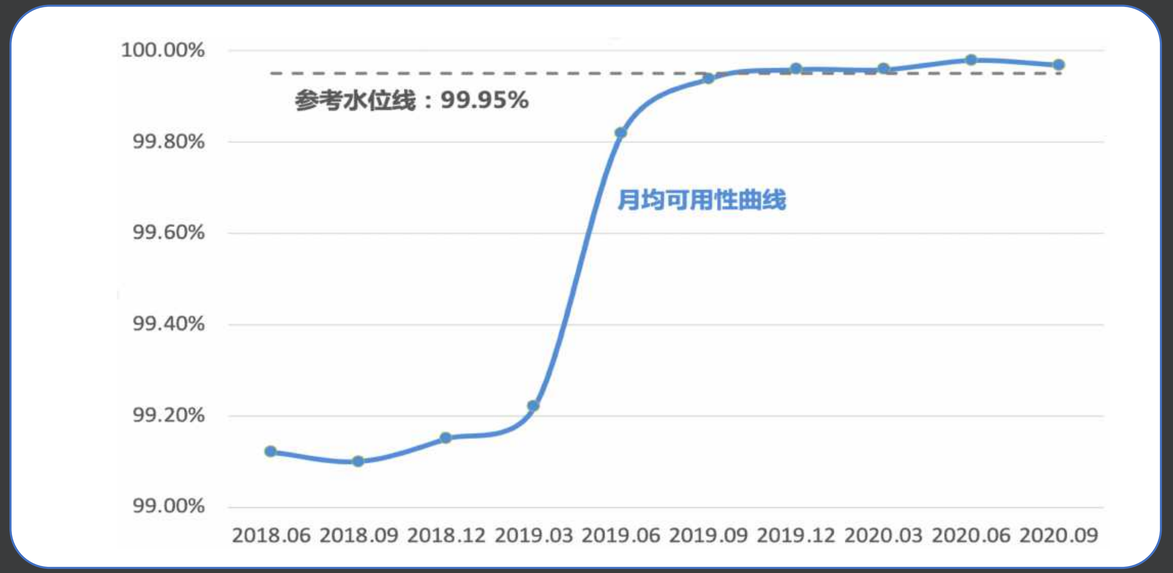
- 可扩展性：支持十万级表、百万级分片（P16）
- 健壮性：容忍压力过载、硬件故障等（P18）
- 集群均衡：多节点、多盘间压力均衡

容灾方案

- 灾难预防：跨可用区容灾
- 数据可恢复：备份、垃圾桶等

总体效果

- 可用性：**由99%提升至99.95%+**
- 运营压力：业务量增长10+倍，运营压力不变



可用性优化 — 集群扩展性

分布式扩展性瓶颈：

- 元数据：分片数达到3w，变更需15s
- 节点数：建议不超过100

线上问题： PB级日志业务按时间分区

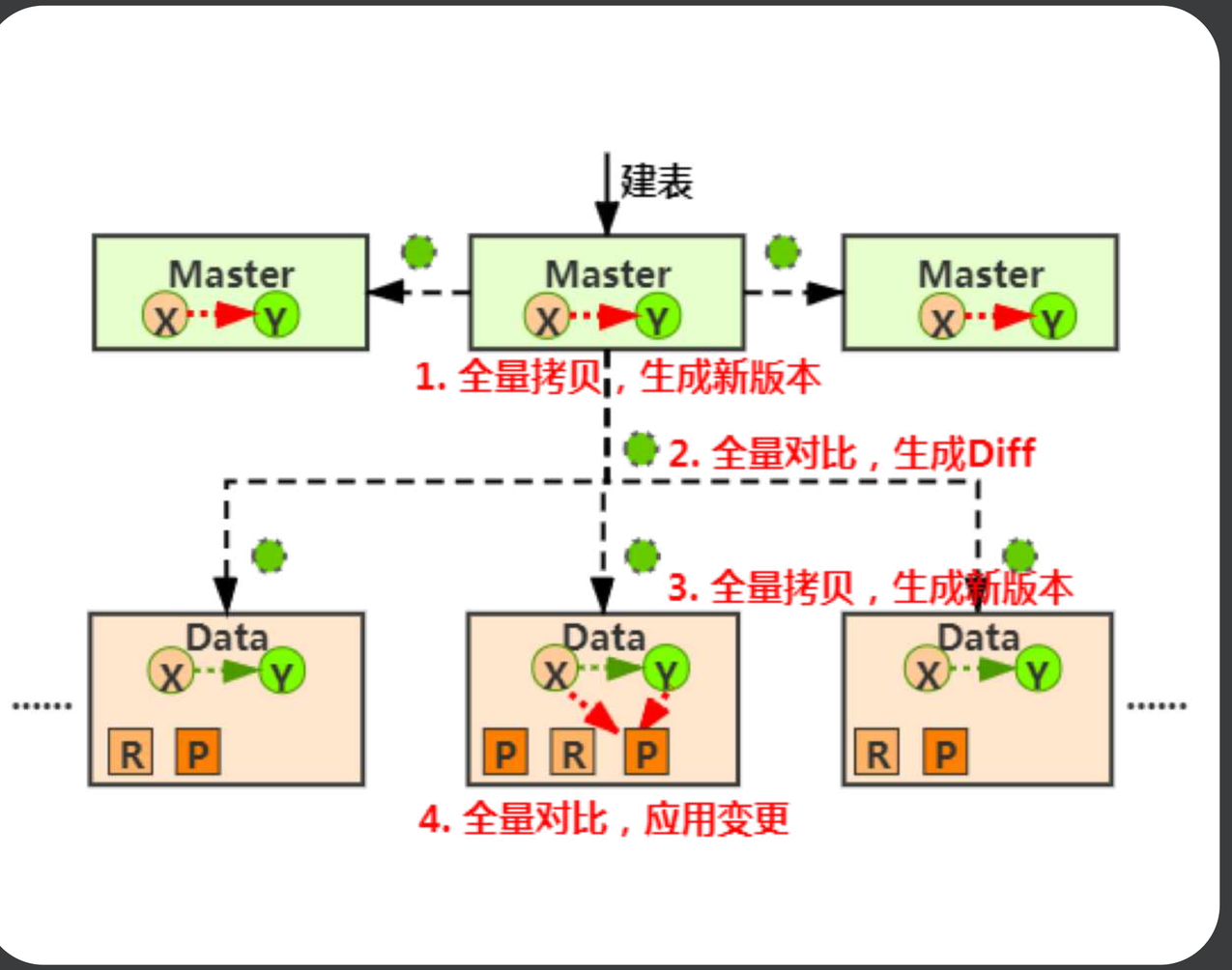
- Master堵塞：元数据变更慢
- 写入拒绝：建表卡住
- 大量中小集群：运营成本高

原因分析： 通过元数据变更实现集群管理

- Master分配分片
- Master产生元数据Diff
- 一阶段：全量分发
- 二阶段：提交应用，等待全量或超时

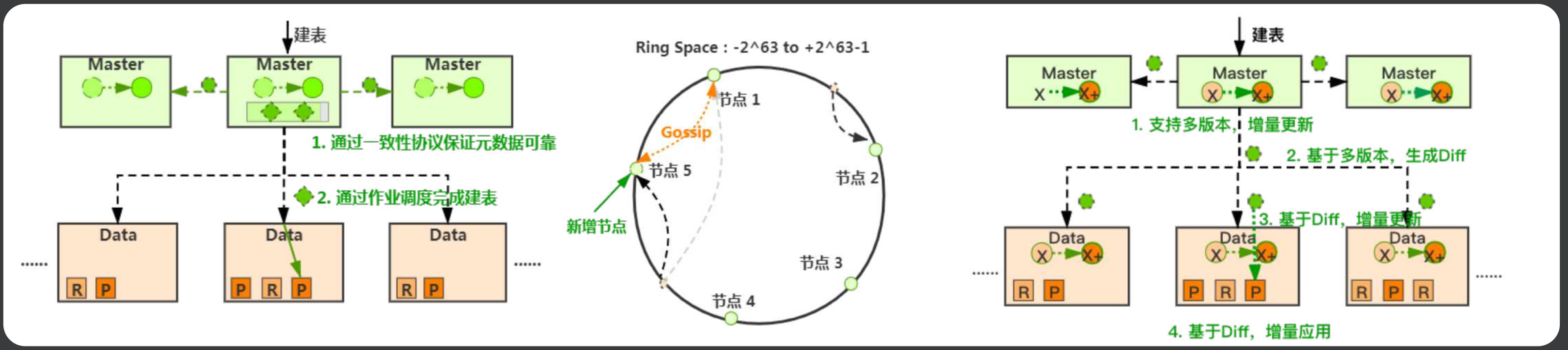
如何解决?

- ES官方：单节点分片<1k，集群<30k，节点数<100



可用性优化 — 集群扩展性

方案



方案	中心化架构（BigTable/Kudu等）	对等架构（Cassandra/Dynamo）	中心化架构（TencentES）
原理	Master HA：分布式一致性协议 集群管理：异步任务调度	所有节点对等：保存全量路由信息 集群管理：Gossip等协议通信	Master HA：分布式一致性协议 多版本元数据，优化元数据瓶颈 变更提交避免全量等待，优化节点瓶颈
优点	效率高、易扩展(百万分片、千级节点)	架构简单清晰，无中心化瓶颈	效率高 (百万分片、千级节点) 、开源兼容
缺点	中心节点易为瓶颈(可解)	效率低，集群扩展性弱(数百节点)	

关键点：1. 元数据多版本的实现；2. 避免全量等待后的路由获取；3. 元数据损坏预案。

可用性优化 — 健壮性架构

具体问题

- 大查询：聚合内存压垮节点
- 高并发点查询：触发拒绝、雪崩
- 高并发批量写：触发抖动、雪崩
- 硬件异常：导致抖动、雪崩

资源分析

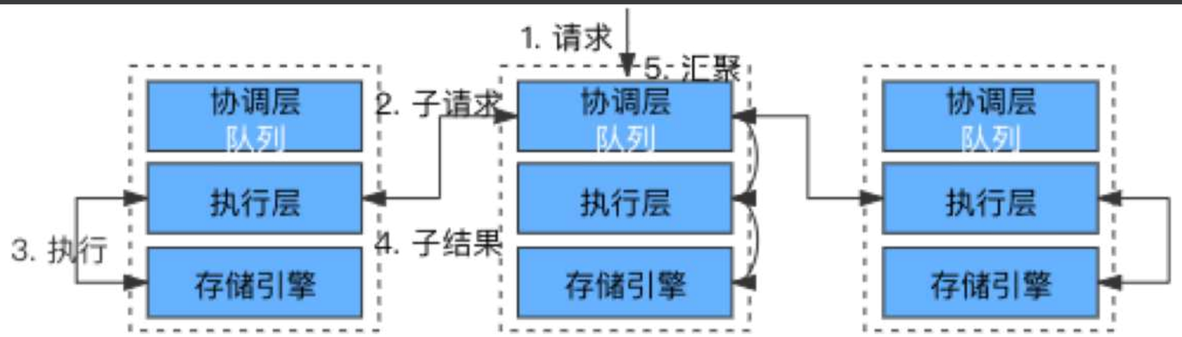
- 瓶颈：**内存** > 计算 > 硬盘 > 网络

如何解决

- Elastic：漏斗限流，功能不完善

思路

健壮性架构：**服务限流** + **异常容忍** + 异常恢复

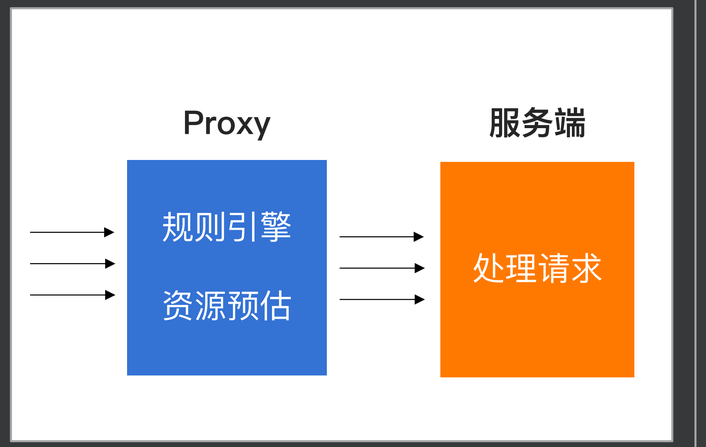


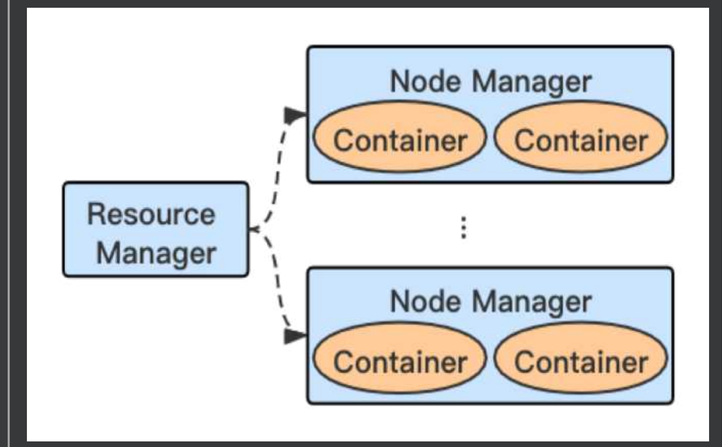
	计算	内存	硬盘	网络
大查询	***	*****	***	*
高并发点查询	****	***	****	*
高并发批量写	****	****	*	*

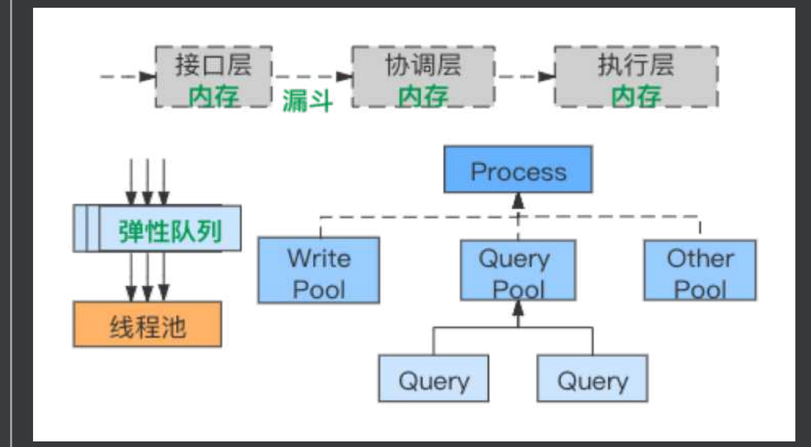
集群资源类	硬盘资源	硬盘资源	……
集群异常类	读写拒绝	节点失联	……
使用规范类	副本数	分片数	……

可用性优化 — 健壮性架构：服务限流

方案







方案	代理限流（微服务）	任务容器（Yarn）	全链路内存熔断 + 弹性漏斗（TencentES）
优点	独立组件，变更易	资源隔离性优	精准控制CPU、IO、内存，充分利用资源
适用	高并发、轻量请求	类MR型长周期任务	混合场景：高并发、混合查询
不足	无法准确预估资源消耗 内存矛盾难解决	资源申请分配消耗高 资源利用率不充分	实现相对复杂

关键点： 1. 如何准确统计大量中小查询的内存开销？2. 服务限流如何保持高吞吐？3. 如何及时熔断，避免OOM?

Patch：ES-46751 ES-46962 ES-47806

可用性优化 — 健壮性架构：异常容忍

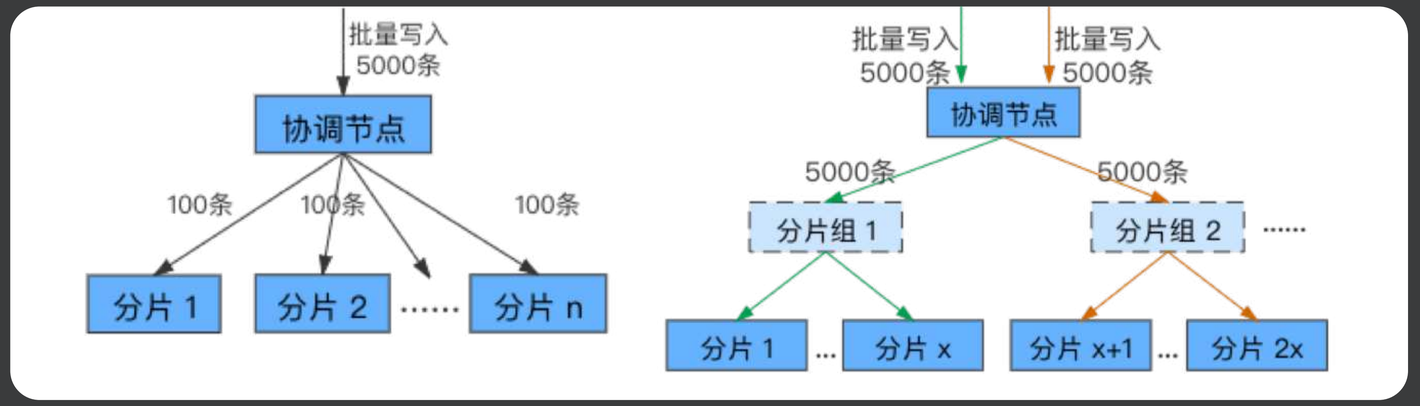
问题
- 日志/时序：百万级TPS，易拒绝、资源压力低
- 搜索：10w级QPS，长尾严重

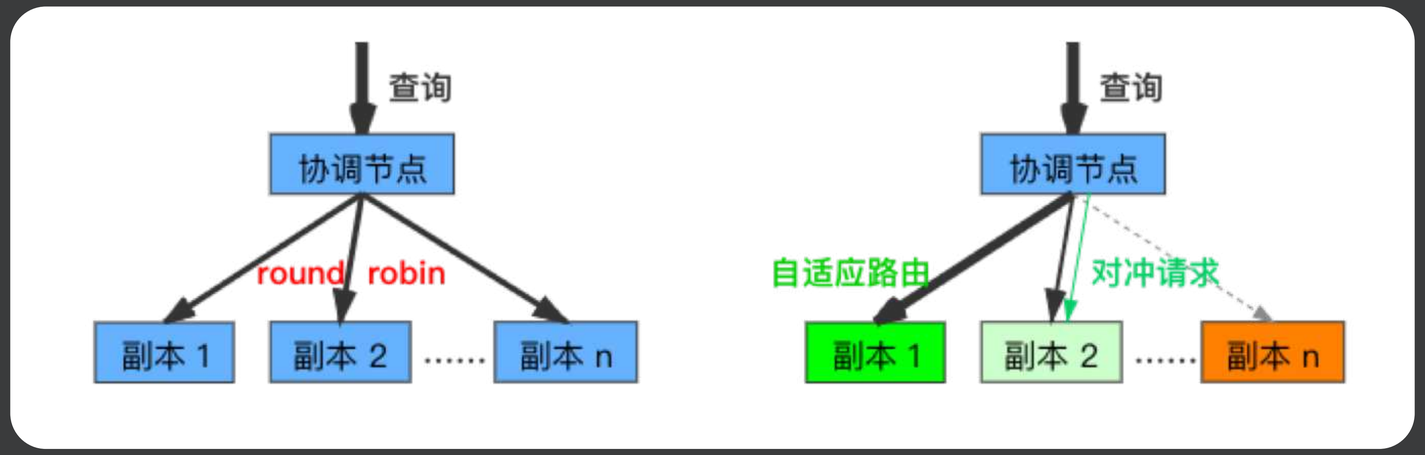
原因
- 异常难消除：硬件异常、后台任务、GC等
- 分布式读写：高扇出放大影响

解决方案
- 写入：**分组路由**，读写扇出平衡
- 查询：**自适应路由 + 对冲请求**

效果
- 写入：TPS提升1+倍，资源利用率提高
- 查询：长尾大幅降低，P99.5延时<100ms

经验：在不可靠环境中，提供可靠服务





	写吞吐(TPS)	CPU	拒绝率	读延时(Avg)	读延时(P99.5)
开源版本	76w	31%	3‰	23ms	1350ms
Tencent ES	169w(+121%)	49%(+58%)	0‰	18ms	85ms

Patch：ES-62020

成本优化 — 解决方案

存储

- 内存优化：降低内存消耗80%+（P22）
- 冷热分层：降低单位存储成本
- 数据上卷：预计算换取存储，数量级（P23）

计算

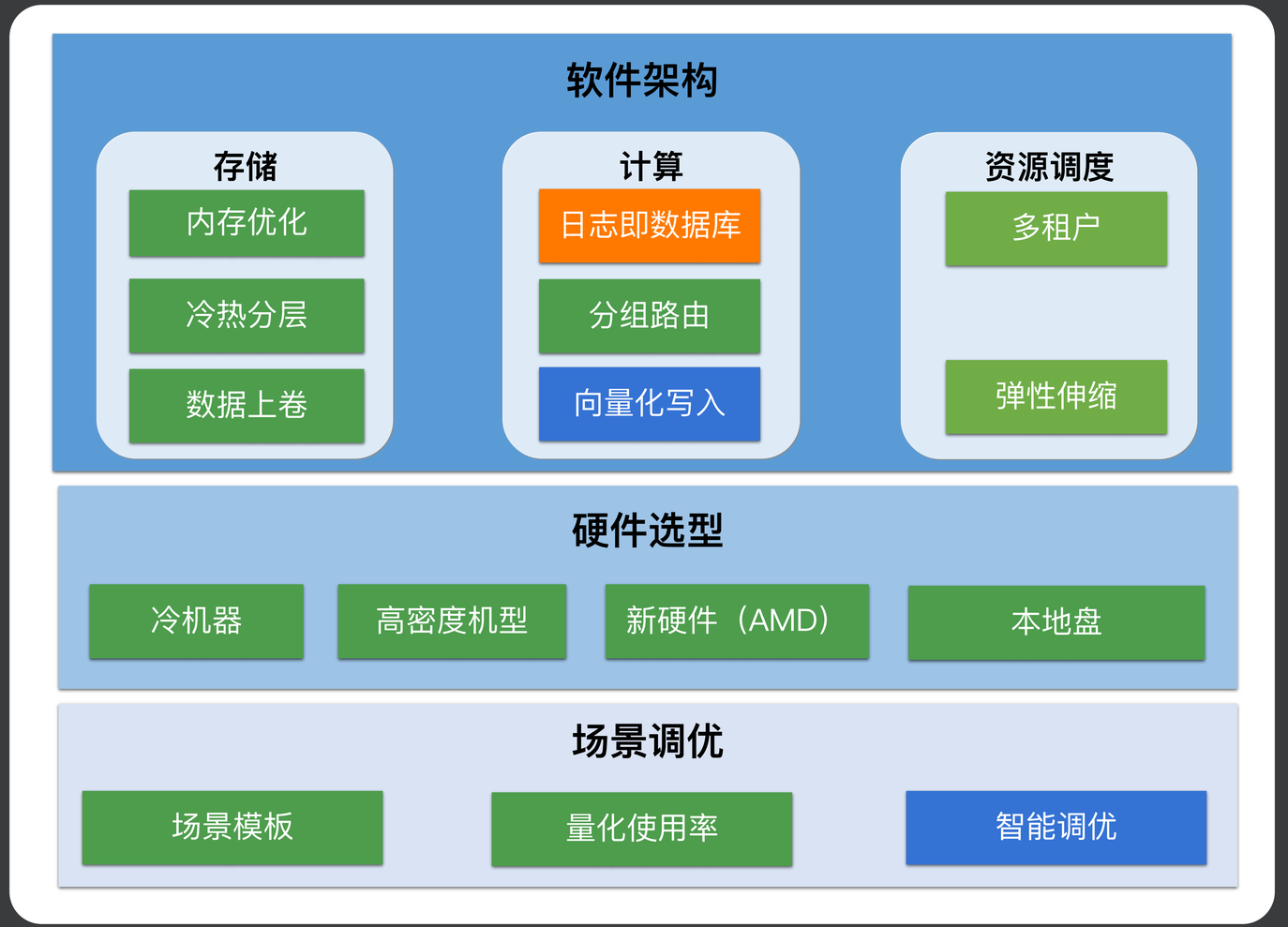
- 日志即数据库：降低1倍计算开销（P24）
- 分组路由：提升1+倍计算利用率

资源调度

- 资源利用率：多租户、弹性伸缩

总体效果

- 日志场景：软件架构**成本降低70%**
- 时序场景：降成本可达一个数量级



成本优化 — 内存优化

内存消耗

- 索引：常驻内存，64G内存支持3T硬盘

内存使用分析

- 历史数据75%+，访问频率低
- 主键索引使用少、占比高

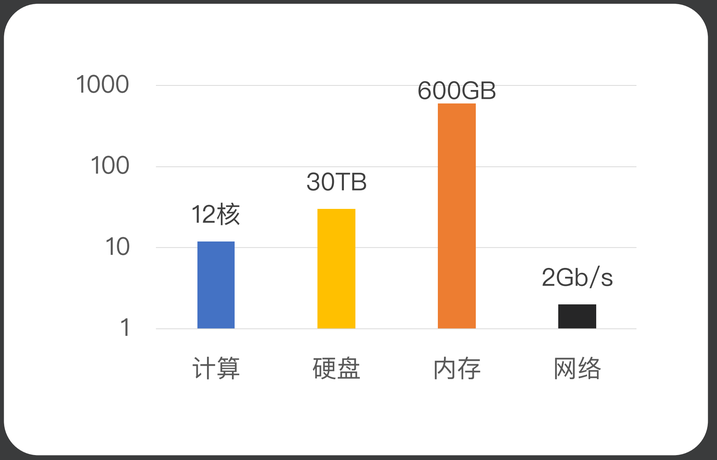
挑战： 提升利用效率，保持查询性能

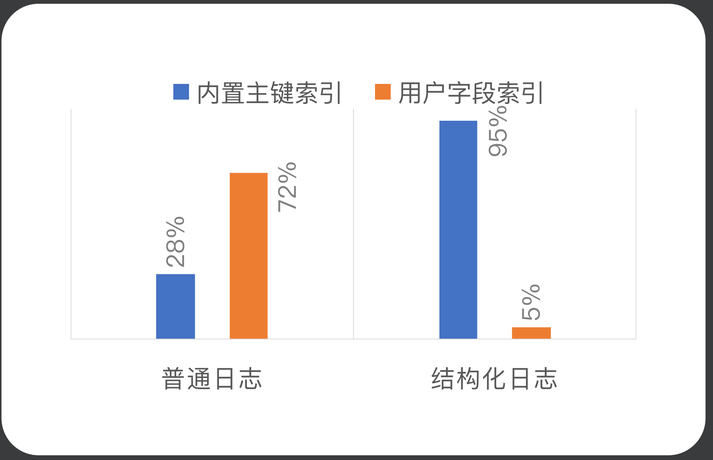
方案：内存Cache化

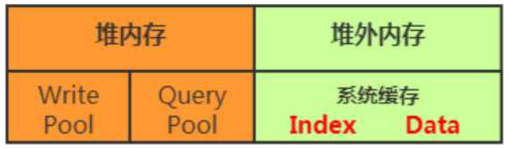
- 精准淘汰策略：主键索引、Merge任务
- 性能开销优化：二级缓存、堆外内存

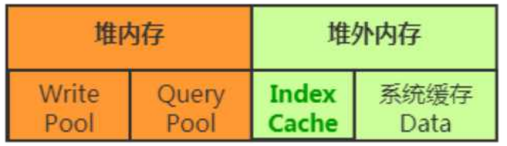
效果

- 内存降低80%，命中率99%+









方案	系统缓存（ES原厂）	独立Cache（TencentES）
原理	依赖系统缓存，索引按需加载	基于独立Cache 精确淘汰策略、性能开销优化
优点	实现简单	性能稳定，接近全内存
不足	系统缓存不区分索引、数据 点查询性能有5+倍的抖动	耦合性较强

成本优化 — 数据上卷：计算置换存储

监控等时序场景：

- 超长周期存储：半年+

用户诉求：如何兼顾?

- 成本低
- 访问速度快

业界思路

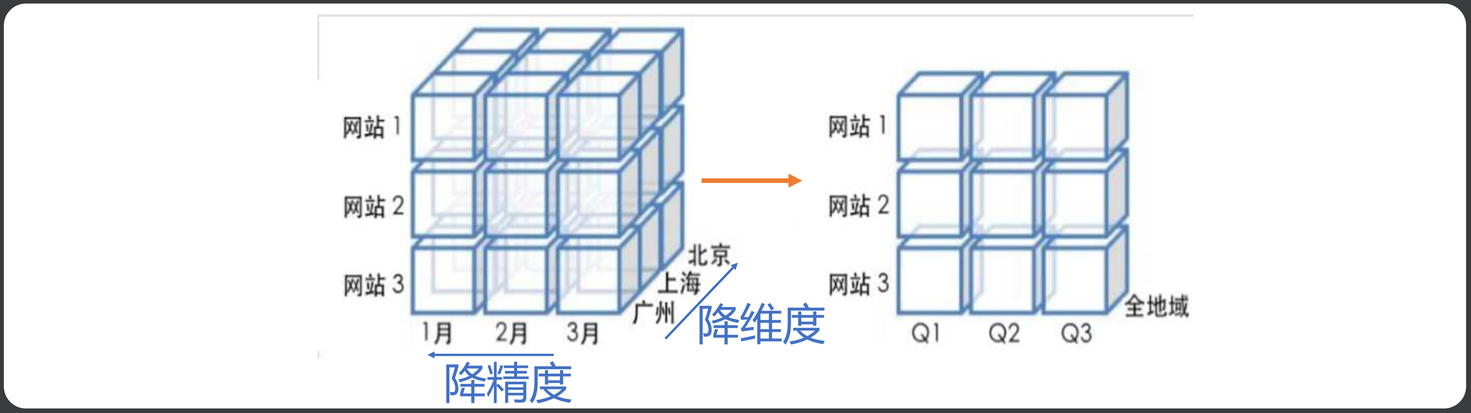
- 数据上卷：预计算 置换 存储

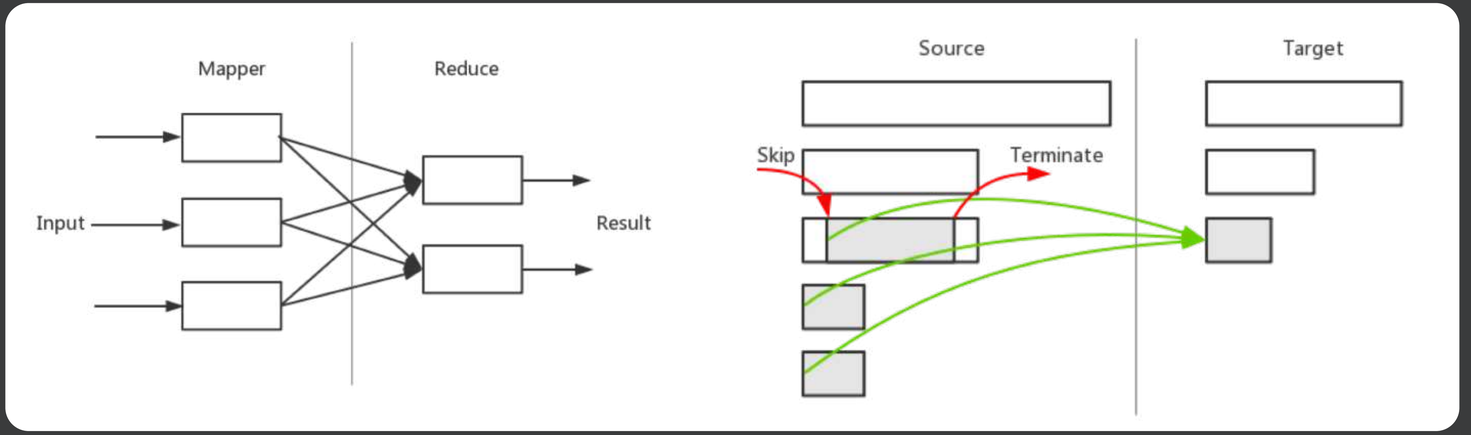
方案

- 离线计算、聚合查询、Merge任务
- 选择：流式多路归并

效果

- **数量级降低存储成本**，性能同步提升





方案	离线计算（Kylin）	流式多路归并（TencentES）
原理	基于Hadoop/Spark进行预计算	异步流式任务，多路归并原始数据
优点	实现简单	计算<10%写入，内存100MB级
缺点	计算开销高，依赖复杂	数据延迟5分钟

成本优化 — 日志即数据库

原生方案：对于主从副本

- 独立对等：WAL + Mem + Disk
- 一致性：全同步 或 Quorum

日志/时序场景特性

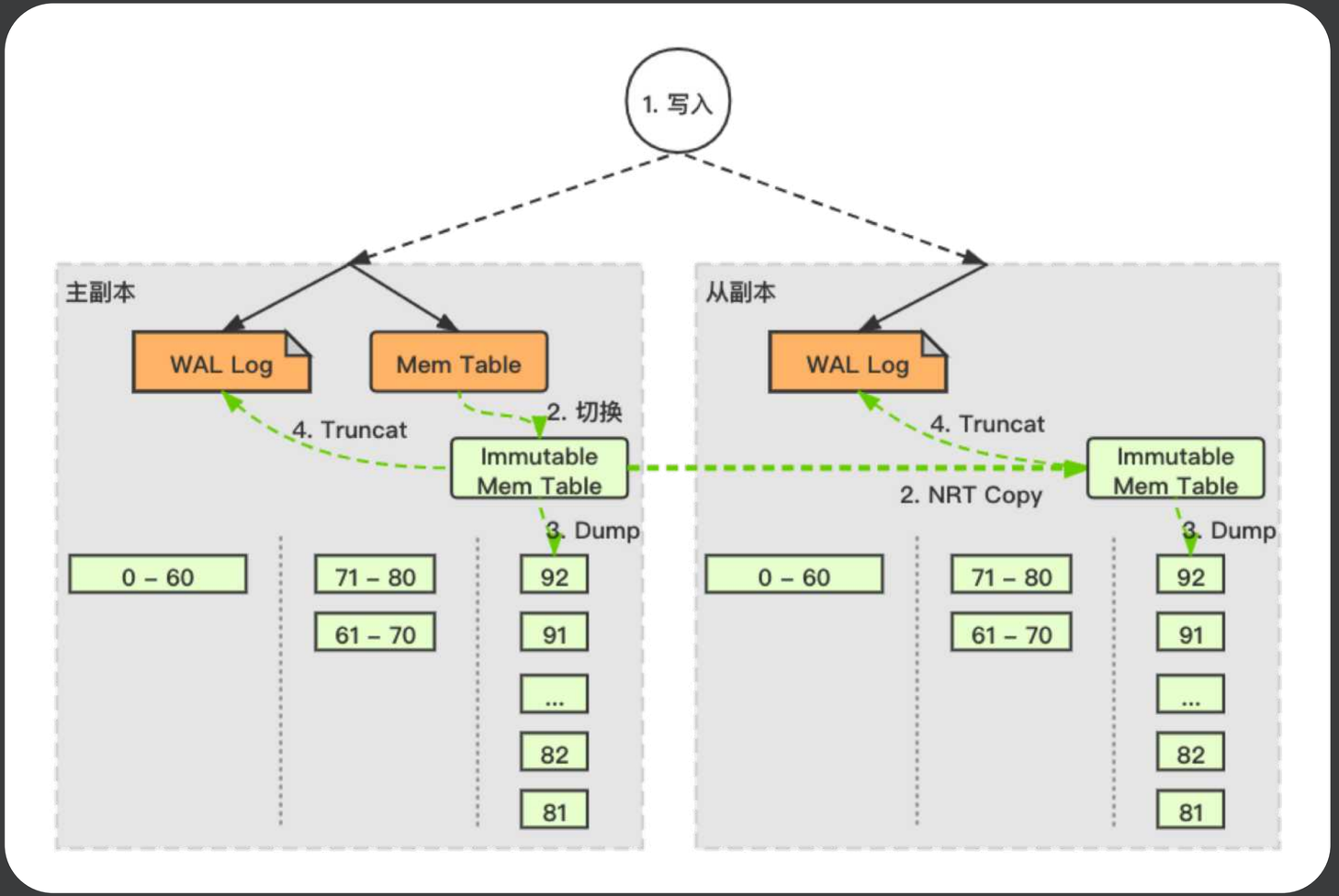
- 写入吞吐高：CPU开销高
- 一致性要求低

优化：日志即数据库

- 主副本：WAL + Mem + Disk
- 从副本：WAL + Disk Segment Copy

效果

- 写入性能（吞吐）提升1倍



性能优化 — 解决方案

存储引擎

- 时序Merge：优化随机IO、裁剪（P26）
- 数据上卷：计算换取性能

执行器

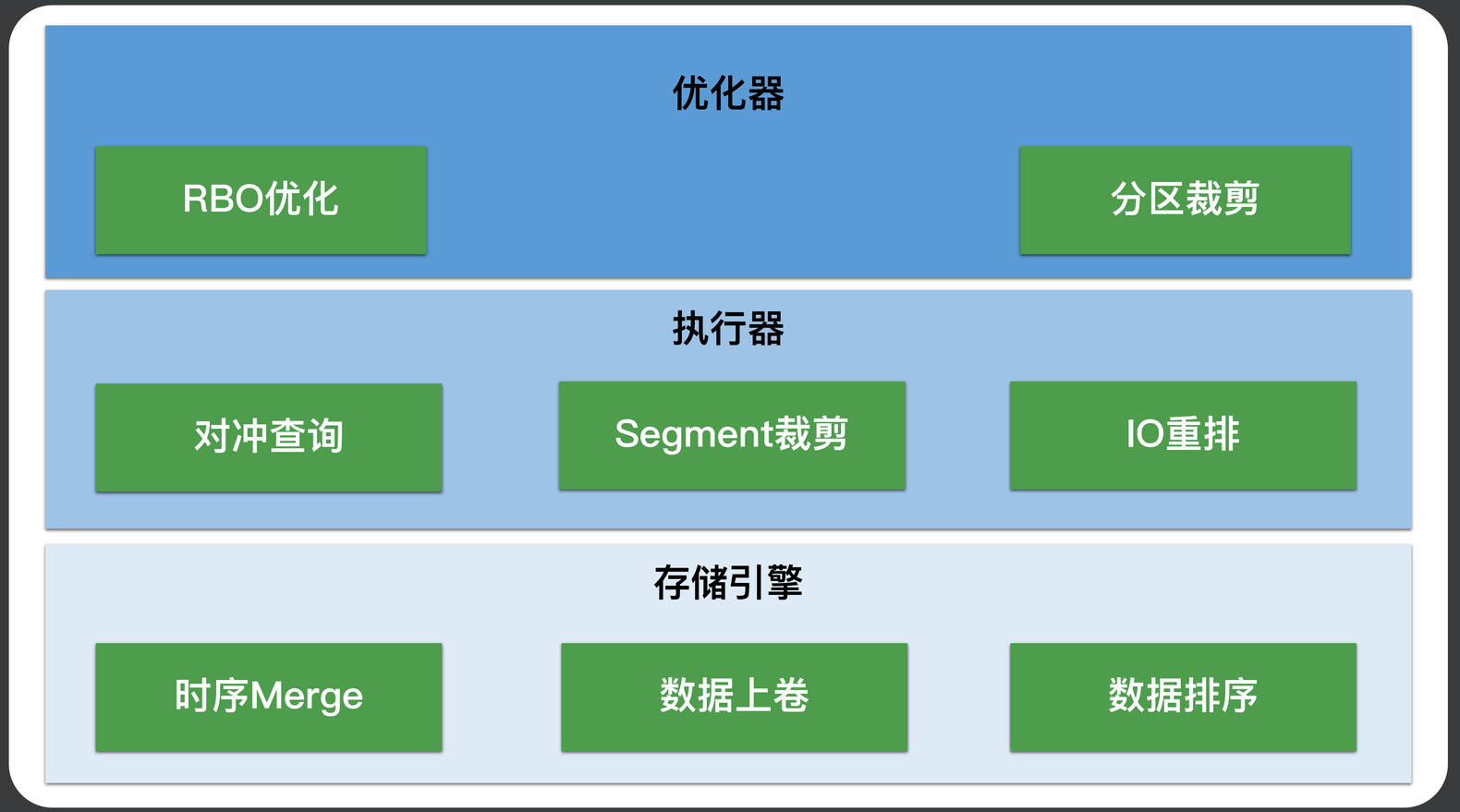
- 对冲查询：数量级降低长尾影响
- 文件裁剪：大幅降低Scan数据量

优化器

- RBO优化

总体效果

- 查询延时：平响降1倍，毛刺降数量级
- 查询吞吐：10w级，线性扩展



• Patch：ES-48399 ES-57273 ES-45765 ES-47790 Lucene-884 Lucene-940

性能优化 — 时序Merge

原生方案

- Merge效率：大小相似性 + 大小上限
- 单分片文件数量：30 +

问题

- 数据时间错乱：不利于查询
- 小文件过多：随机IO严重

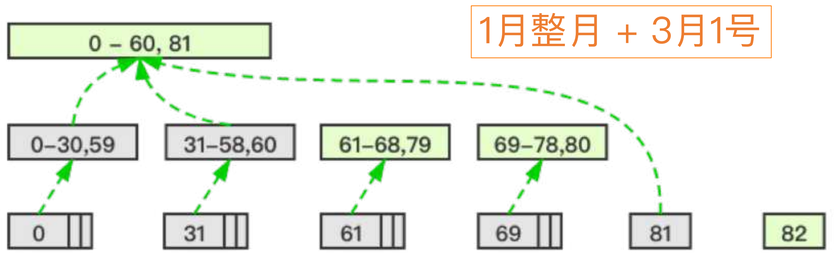
方案

- 时间序Merge：增加时间邻近特性
- 冷数据Merge：收敛文件数

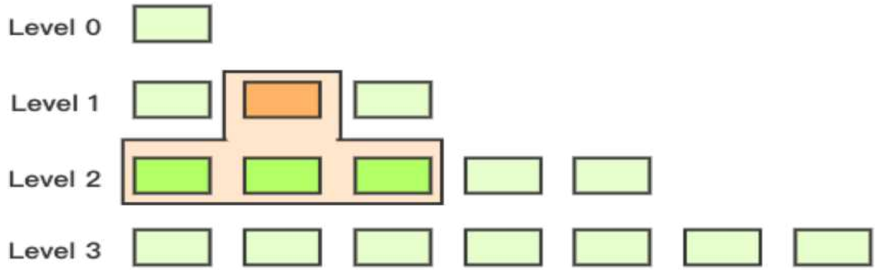
效果

- 搜索场景：性能提升近1倍

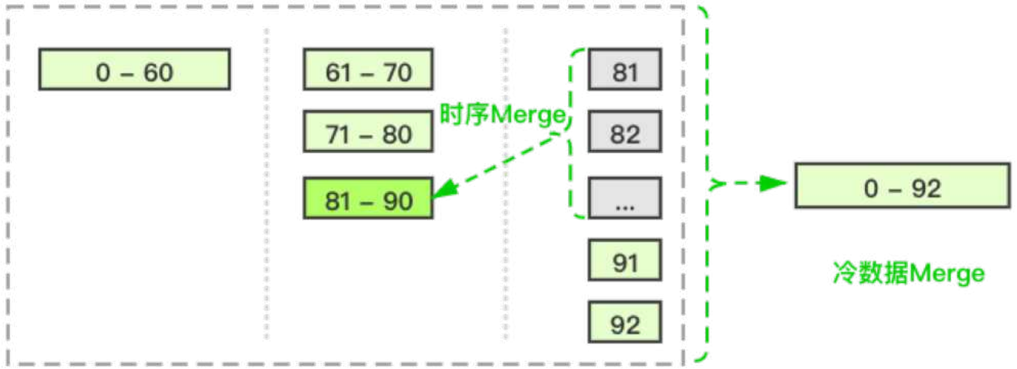
原　生



LevelDB



TencentES



目录

04

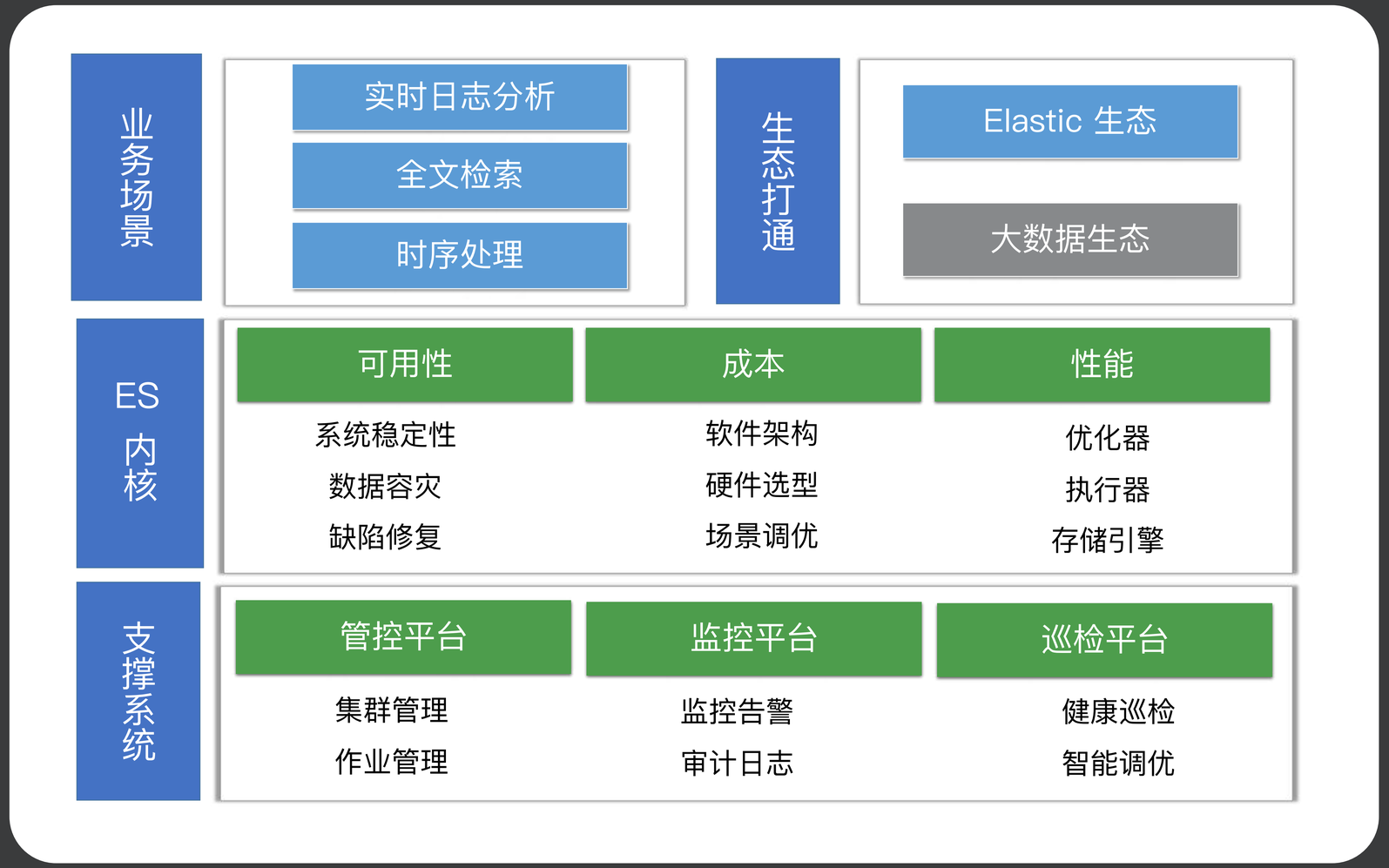
现状总结

ES内核
- 可用性
- 成本
- 性能

支撑系统
- 完整的托管能力

业务发展
- 业务体量：50w核
- 日志实时分析、搜索、时序处理

生态融合
- Elasitc生态：完整兼容



未来发展

成本优化：实时分析痛点

- 软件架构：存算分离、模式感知压缩等
- 资源调度：多租户、无服务器化

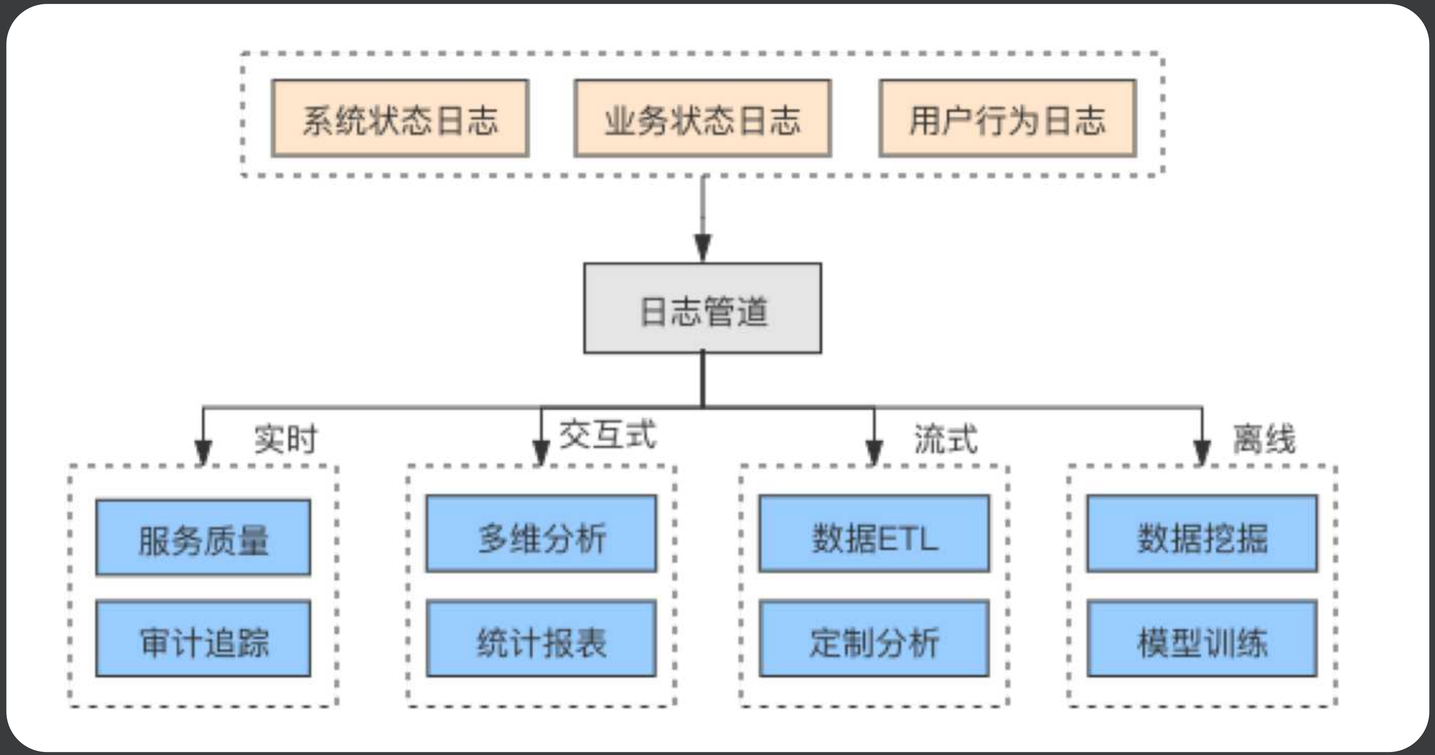
➡ **持续降低客户成本**

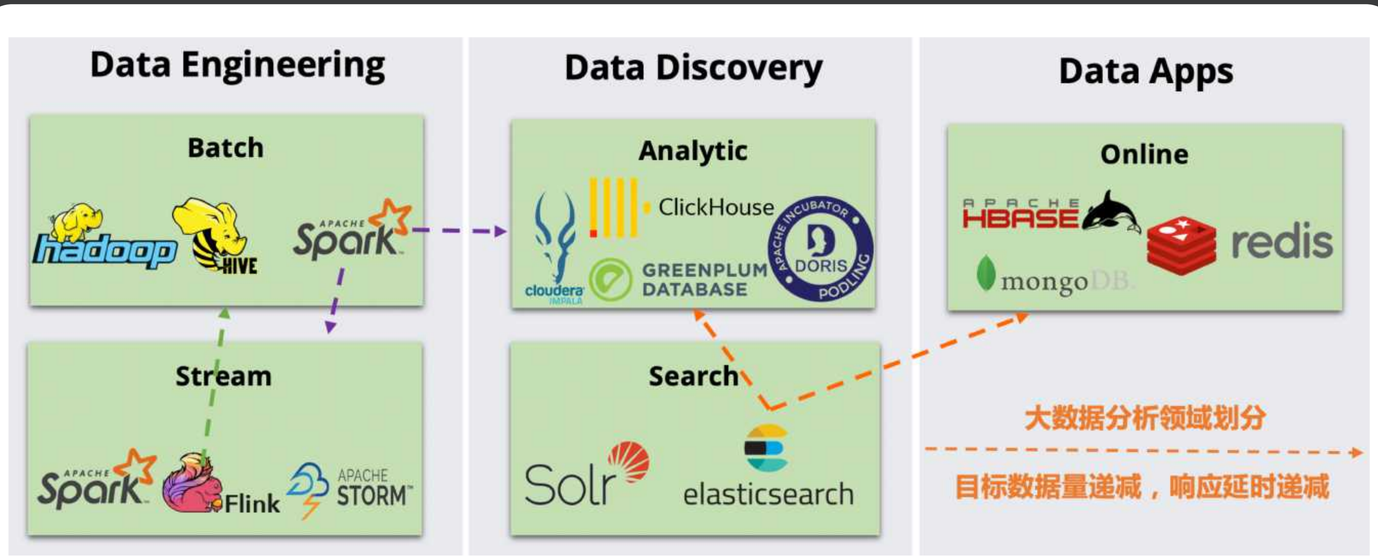
生态打通：大数据

- 提供完整服务体系：用户诉求
- 拓展数据来源

场景拓展：交互式分析

- 平台：闭环PB级日志处理，**挖掘潜在价值**
- 用户：多套系统资源、维护成本高





开源协同

开源贡献

- 源码反馈：80+ Patch，6 贡献者，亚太区第一
- 社区活动：参与或组织开发者大会、Meetup等
- 技术文稿：技术征文、文档翻译等

技术收益

- 版本管理：降低维护成本，持续跟进社区
- 人才培养：规范、高效、开放的研发环境

影响力收益

- 社区认可：Elastic CEO、Developer的认可
- 社区组织：中文社区主席团成员
- 人才吸引、产品发展

目标

- 合作共赢：开源社区、企业、个人

> We have recently made a major and ingenious improvement to Elasticsearch, which was proposed by a developer of Tencent. This improvement makes Elasticsearch Some types of write speeds have increased by about 20%, and we are very much looking forward to continuing this good relationship with Tencent Cloud.
>
> — Elastic CEO : Shay Banon

THANKS