科大讯飞 iFLYTEK | elastic 中文社区

基于容器云平台的ES和ELK服务在讯飞云实践

主讲人：欧开亮

目录

01

PART 01

现状

现状

集群规模

- 4个可用区
- 47个ES集群
- 45个ELK集群
- 单集群最大25w/s
- 单集群最多15T

业务落地

- 50+业务线
- 日志搜索与分析
- 指标数据分析
- 数据库
- APM

02 PART 02 云ES/ELK服务建设

云ES/ELK服务介绍

- 依托讯飞云（集团云），底层基于k8s容器云平台、ceph存储
- 上层编排服务，提供ES/ELK集群页面化管理
- 2018年初开始研发，经过技术选型、多伦迭代和测试，下半年开始尝试推广
- 2019年版本迭代、增加新功能、提高稳定性和故障处理能力
- 2020年进一步推广和迭代优化
- 提供6.3.1/6.7.0版本，多种集群架构，完全兼容官方协议

云上提供的功能

服务	功能
ES/ELK	集群快速创建、扩容、重启、销毁
ES/ELK	滚动扩容、存储和节点数扩容不影响集群服务
ES/ELK	监控、节点日志、参数配置、回收站
ES/ELK	自带Kibana访问
ES/ELK	域名方式访问集群服务
ES	数据备份和恢复、自定义中文分词、数据定时清除策略
ES	支持选择专有主节点和协调节点

技术选型

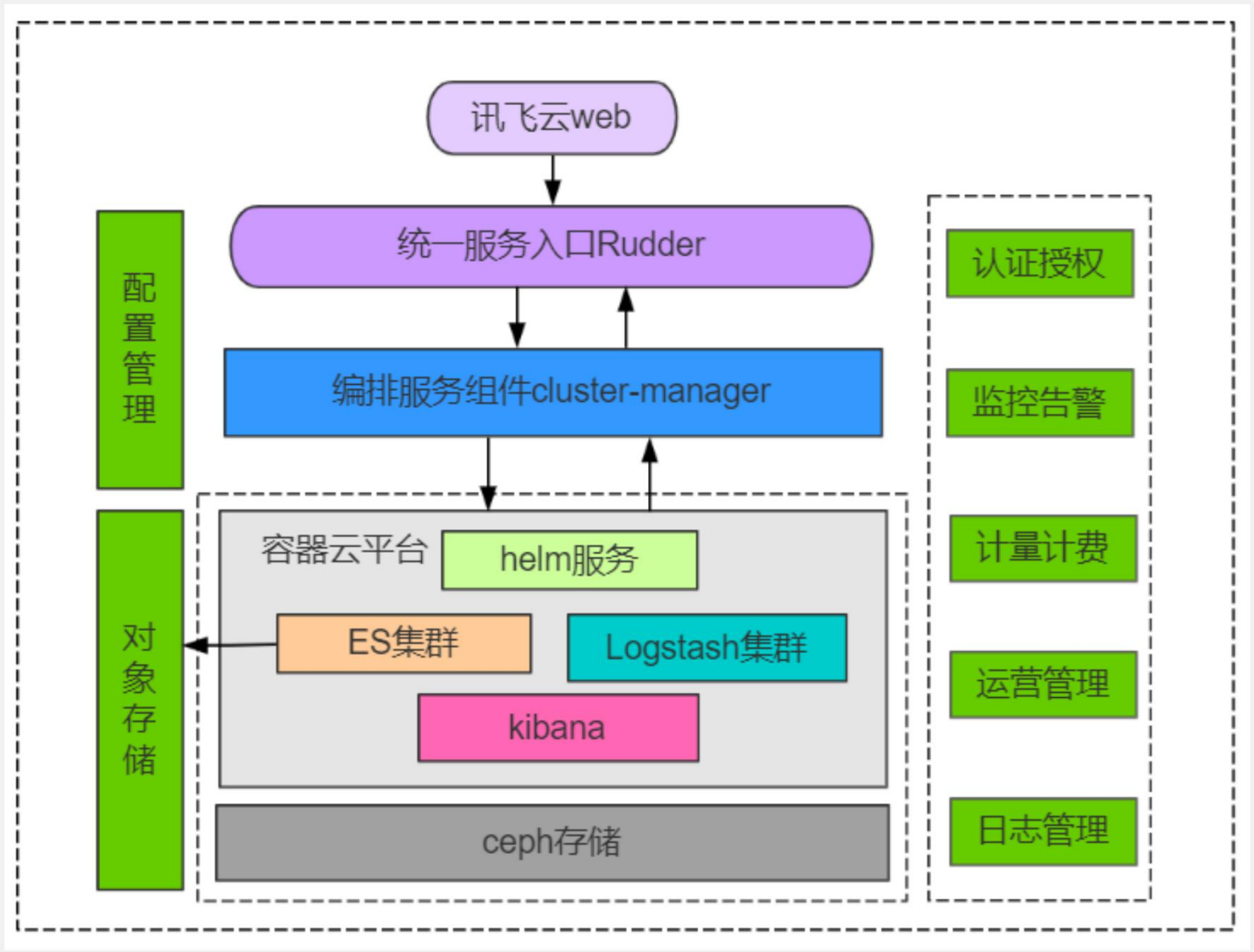
- 底层：k8s、docker、ceph存储、集团云PowerDNS
- 上层编排服务：SpringBoot、MyBatis、MySQL、Vue
- 上下层交互：http、Helm
- 多可用区部署

总体架构

- 集群基础配置管理
- 镜像管理
- Helm包管理。

- 数据备份和恢复

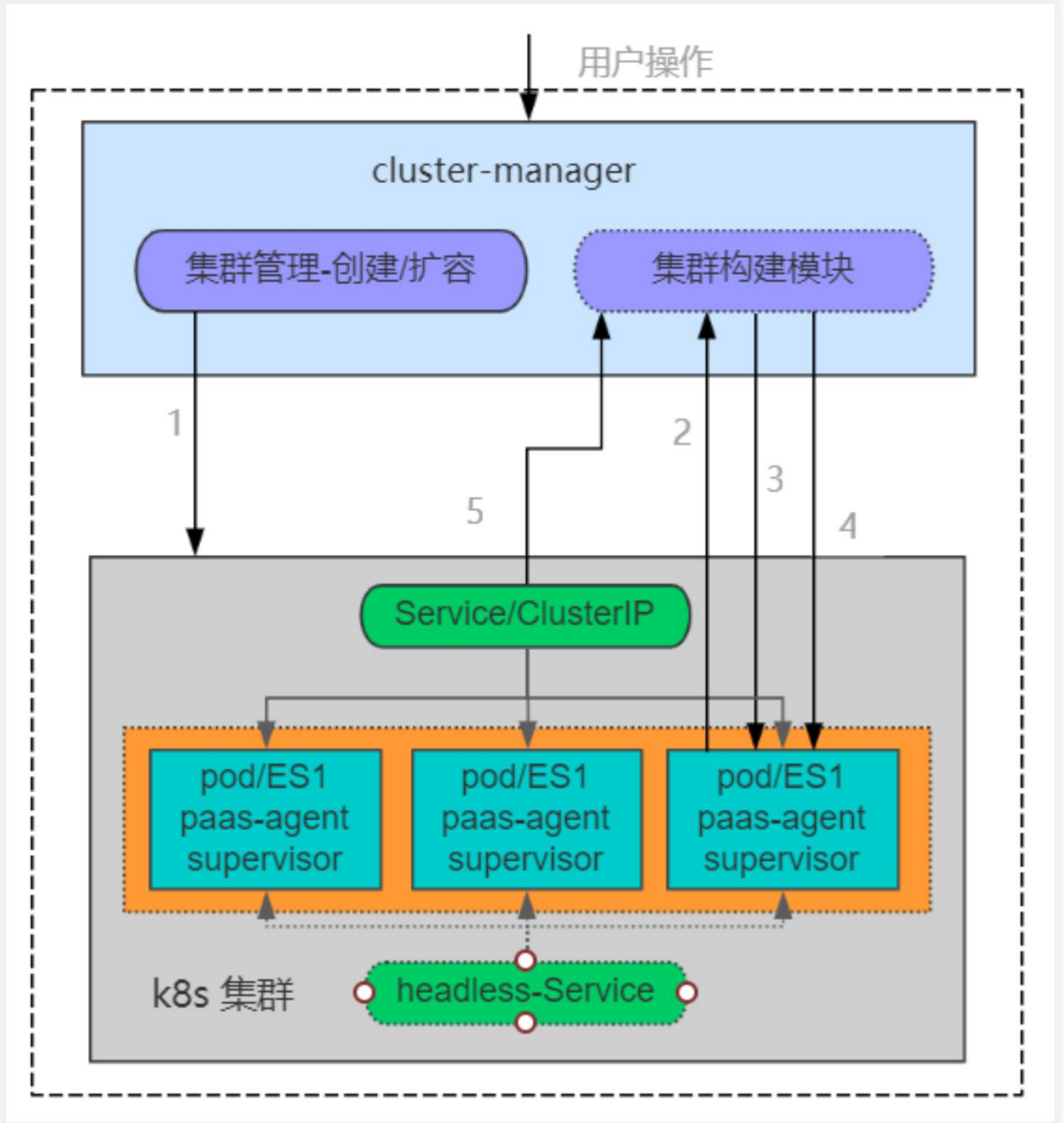
- ES/ELK集群构建和管理



编排服务核心流程

- 标准化集群构建流程，适用整个PaaS产品。
- 准备：镜像构建（ELK代码、paas-agent、supervisor）
- 准备：ES/ELK HelmChart包。
- 1.cpu/内存/存储大小/镜像/helm包地址。
- 2.探活paas-agent服务可用性。
- 3.构建和向paas-agent下发配置文件(脚本)。
- 4.启动ES/Logstash/Kibana进程。
- 5.探活集群服务可用性。
- 并行或串行完成以上流程。
- 其中logstash采用单Service和单Pod模式。

  需要客户端支持负载均衡。



编排服务标准化优缺点

优点

- 适用于PaaS平台所有产品
- 统一管理产品不同版本产品配置
- 新产品接入快
- 集群搭建过程可控
- k8s层yaml资源文件简单
- 升级兼容性较好

缺点

- 无法更好应用更多k8s层面的元素
- 如健康检查，configmap
- 无法指定顺序pod

HelmChart包核心yaml

namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: {{ .Values.metadata.namespace }}
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
  name: default
  namespace: {{ .Values.metadata.namespace }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: paas-role
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
---
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: {{ .Release.Name }}-es-pdb
  namespace: {{ .Values.metadata.namespace }}
spec:
  maxUnavailable : 50%
  selector:
    matchLabels:
      name: es-pdb
```

es-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.config.esServiceName }}
  namespace: {{ .Values.metadata.namespace }}
  labels:
    app: elasticsearch
    release: {{ .Release.Name }}
spec:
  type: ClusterIP
  selector:
    pod-es: pod-es{{ .Values.config.id }}
  ports:
  - name: es-transport
    port: 9300
    protocol: TCP
    targetPort: 9300
  - name: es-http
    port: 9200
    protocol: TCP
    targetPort: 9200
---
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.config.esServiceName }}-in
  namespace: {{ .Values.metadata.namespace }}
  labels:
    app: elasticsearch
    release: {{ .Release.Name }}
spec:
  clusterIP: None
  selector:
    pod-es: pod-es{{ .Values.config.id }}
```

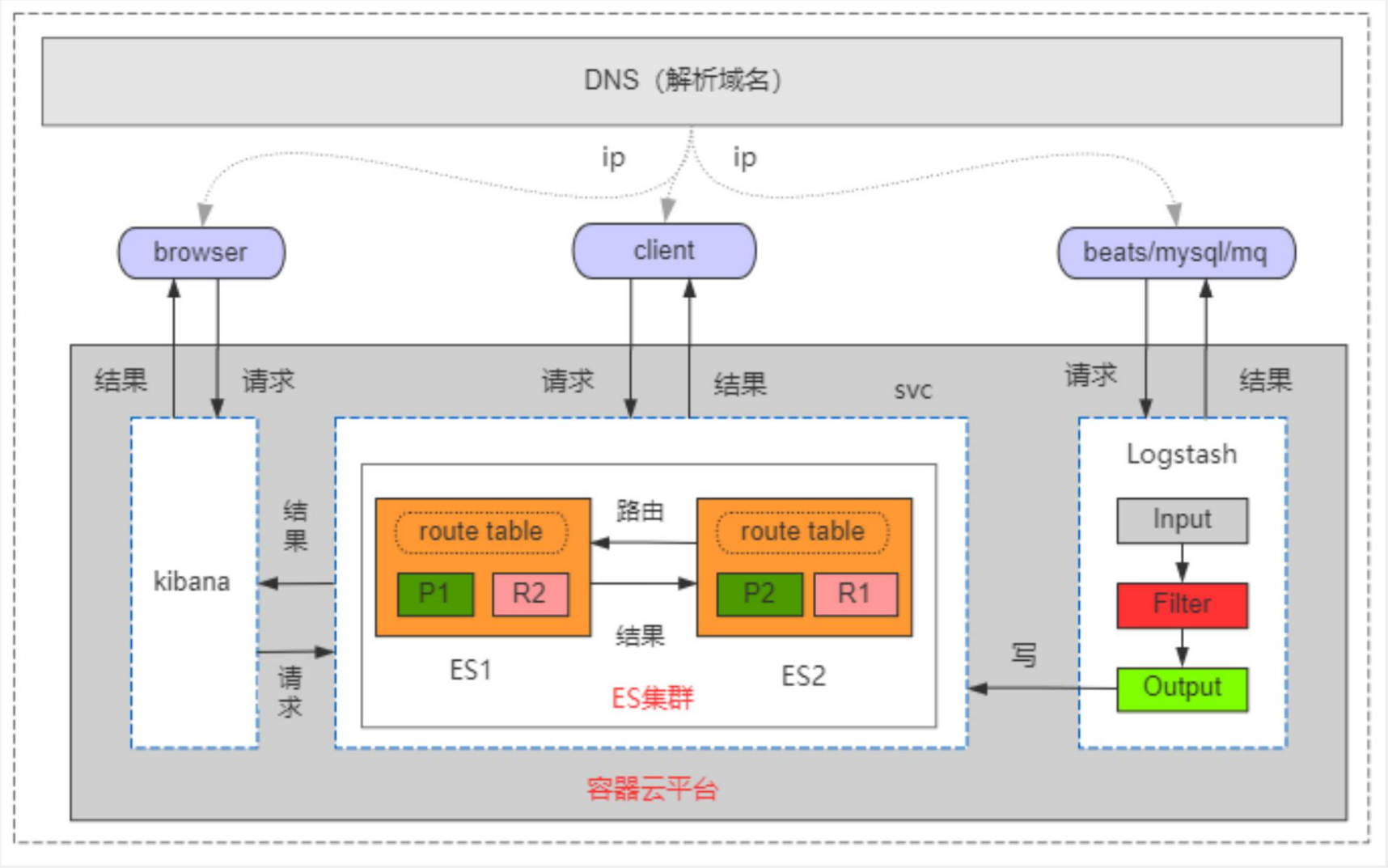
HelmChart包核心yaml

es-sts.yaml

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  annotations:
    "initializer.kubernetes.io/lxcfs": "true"
  name: {{ .Values.resources.es.stsName }}
  namespace: {{ .Values.metadata.namespace }}
  labels:
    app: elasticsearch
    release: {{ .Release.Name }}
spec:
  serviceName: {{ .Values.config.esServiceName }}-in
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      partition: 0
  replicas: {{ .Values.resources.es.replicas }}
  template:
    metadata:
      labels:
        pod-es: pod-es{{ .Values.config.id }}
        name: es-pdb
    spec:
      terminationGracePeriodSeconds: 30
      initContainers:
      - name: "chown"
        command: ["/bin/bash", "-c", "chown -R genitus:genitus
{{ .Values.config.esHomeDir }}"]
        image:
"{{ .Values.resources.es.image.name }}:{{ .Values.resources.es.image.
tag }}"
        imagePullPolicy:
"{{ .Values.resources.es.image.imagePullPolicy }}"
        securityContext:
          runAsUser: 0
        volumeMounts:
        - mountPath: {{ .Values.config.esHomeDir }}/out
containers:
    - name: es
      env:
      - name: POD_IP
        valueFrom:
          fieldRef:
            fieldPath: status.podIP
      -  name: ES_PROCESSORS
        value: "{{ .Values.resources.es.limits.cpu }}"
      - name: NODE_MASTER
        value: "true"
      - name: NODE_DATA
        value: "true"
      - name: NODE_INGEST
        value: "true"
      command: ["/usr/bin/supervisord", "-c",
"/home/genitus/supervisor.conf"]
      image:
"{{ .Values.resources.es.image.name }}:{{ .Values.resources.es.image.t
ag }}"
      resources:
        limits:
          cpu: {{ .Values.resources.es.limits.cpu }}
          memory: {{ .Values.resources.es.limits.memory }}
          ephemeral-storage: 10G
        requests:
          cpu: {{ .Values.resources.es.requests.cpu }}
          memory: {{ .Values.resources.es.requests.memory }}
      imagePullPolicy:
"{{ .Values.resources.es.image.imagePullPolicy }}"
      ports:
      - containerPort: 9200
        name: http
      - containerPort: 9300
        name: transport
      - containerPort: 10080
        name: agent
 volumeMounts:
      - name: es-data
        mountPath:
{{ .Values.config.esHomeDir }}/out
volumeClaimTemplates:
  - metadata:
      name: es-data
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage:
{{ .Values.resources.es.storage.capacity }}
      storageClassName:
{{ .Values.resources.es.storage.type }}
```

应用如何访问集群

- 域名访问，集群创建成功后生成域名。



遇到的问题

- **问题1：** ES跑在k8s中和相对于物理机中性能如何？

表 5.1 Elasticsearch 在物理机、Docker、K8s 环境下分别外挂 HDD 和商业存储测试结果汇总表

	HDD		商业存储	
	写（docs/s）	读（ops/s）	写（docs/s）	读（ops/s）
物理机	42370.4	50.0512	46400.6	50.0481
Docker	42350	50.051	47720	50.0518
K8s 容器	42733.5	50.0518	42462.7	50.048
Docker 性能变化	-0.048%	-0.0004%	2.765%	0.007%
K8s 容器性能变化	0.857%	0.001%	-8.487%	-0.0002%

备注： 1.表中性能变化是对比物理机的，“-”表示下降

2. 由于测试环境问题，导致 K8s 容器上外挂商业存储测试结果波动较大，有约 10%的差异，上表 K8s 容器外挂商业存储数据是测试五次取平均值的结果，环境问题后续跟进中。

遇到的问题

- **问题2**：ES使用Ceph存储相对于本地磁盘性能如何？

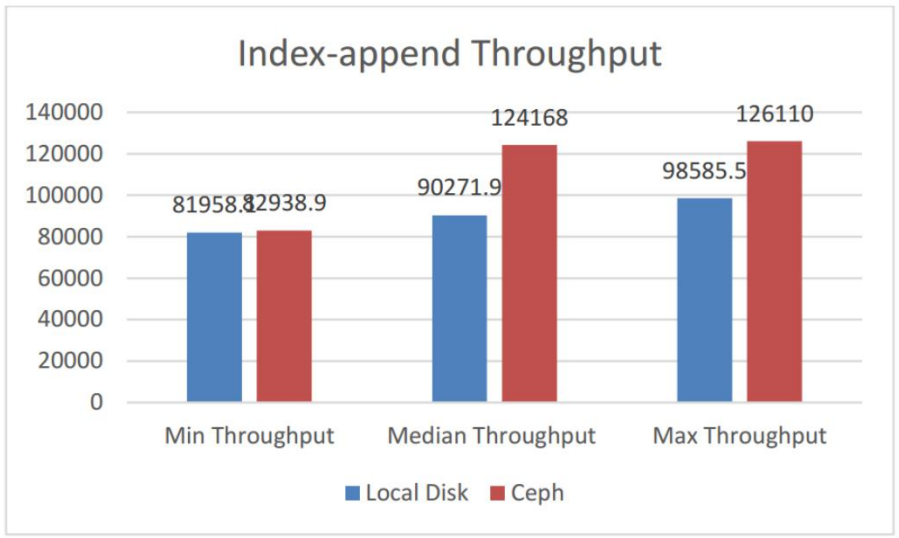


图 4-1 index-append 吞吐量本地磁盘与 Ceph 测试对比

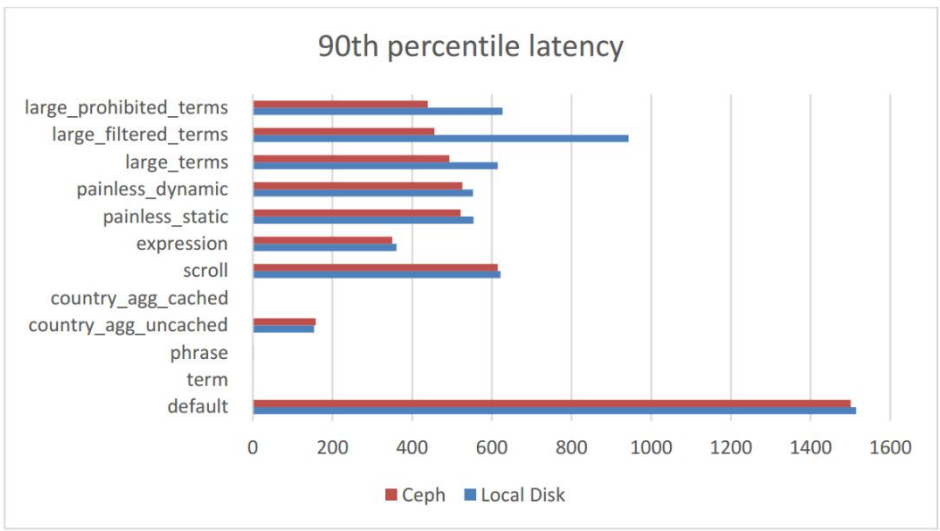


图 4-4 各查询操作 90th percentile latency 本地磁盘与 Ceph 测试对比

在本次基于 Rally 的性能测试中，由以上章节的测试数据和测试分析可以得出以下结论：

1） Ceph 比本地磁盘写吞吐量高，中值吞吐量高出 37.5%。

2） Ceph 在写入处理速度上比本地磁盘快，90th percentile 比本地磁盘快 29.8%，相应地，写入延迟亦比本地磁盘低。

3） 在查询吞吐量方面，Ceph 与本地磁盘几乎相同。

4） Ceph 在一般的查询处理上比本地磁盘略快，而大文件查询处理速度明显比本地磁盘更快（10%以上），相应地，查询延迟亦比本地磁盘低。

遇到的问题

- **问题3**：k8s中默认使用root挂载数据卷，ES非root启动进程无法读写数据目录？

  解决：增加initContainer，使用runAsUser: 0修改挂载的数据目录权限为ES可读写。

- **问题4**：k8s中ES podIp会随pod漂移而变化，ES配置文件无法使用该IP做服务发现？

  解决：增加Headless Service，使用其名称做ES节点之前服务发现地址。

  discovery.zen.ping.unicast.hosts: es-svc-in

- **问题5**：容器云平台不支持cpu无效，导致ES获取到是宿主机cpu核数，影响性能？

  解决：ES配置文件中配置processors: pod cpu核数。提升:写40%，部分读25%。

  后续平台通过lxcfs提供隔离。

- **问题6**：ES6.3.1和6.7.0数据本备份无法直接对接集团云对象存储（ s3 ）？

  解决：1）修改s3插件源码，重写endpoint获取，签名协议。

  2）jvm.options中添加-Des.allow_insecure_settings=true。

遇到的问题

- **问题7**：中文分词词库更新需要重启实例？

  解决：开发一个独立中文词库获取服务，在ES集群构建时，将服务地址下发到IKAnalyzer.cfg.xml中热更新配置中。

- **问题8**：logstash单Service多实例架构，大部分场景影响处理性能？

  解决：改为单Service单实例架构，依据：大部分场景为为filebeat写logstash模式。

- **问题9**：logstash JVM使用率随着数据写入一直上升，达到90%后才GC，波动明显？

  解决：1）改为G1GC垃圾回收器。

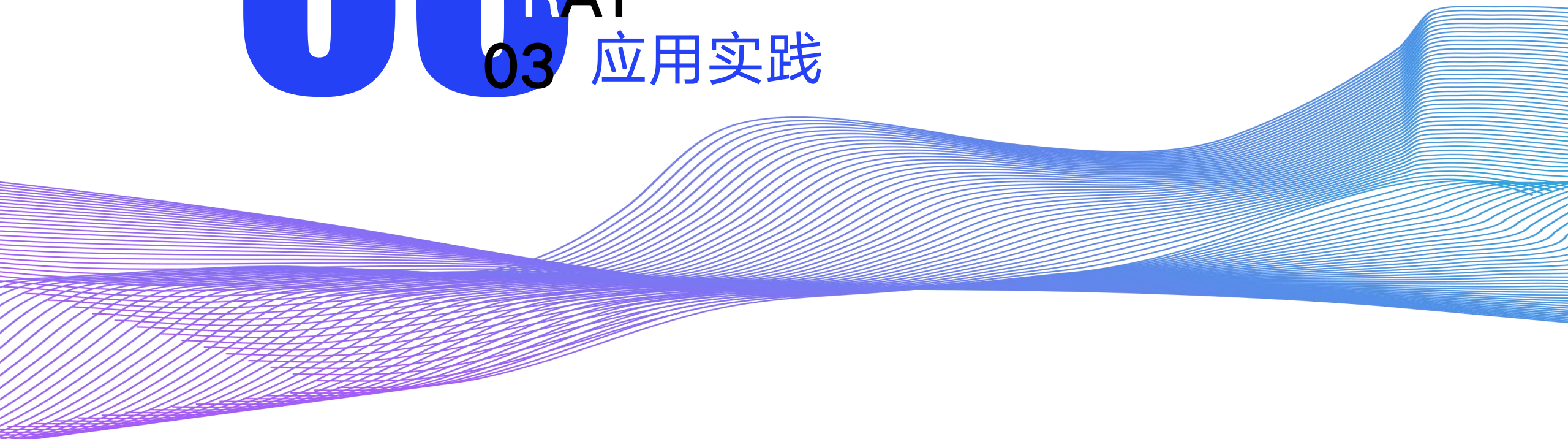
  2）调整默认值 pipeline.batch.size: cpu核数*125，pipeline.workers: cpu核数。

  3）经测试对比性能提升30%-50%。

03

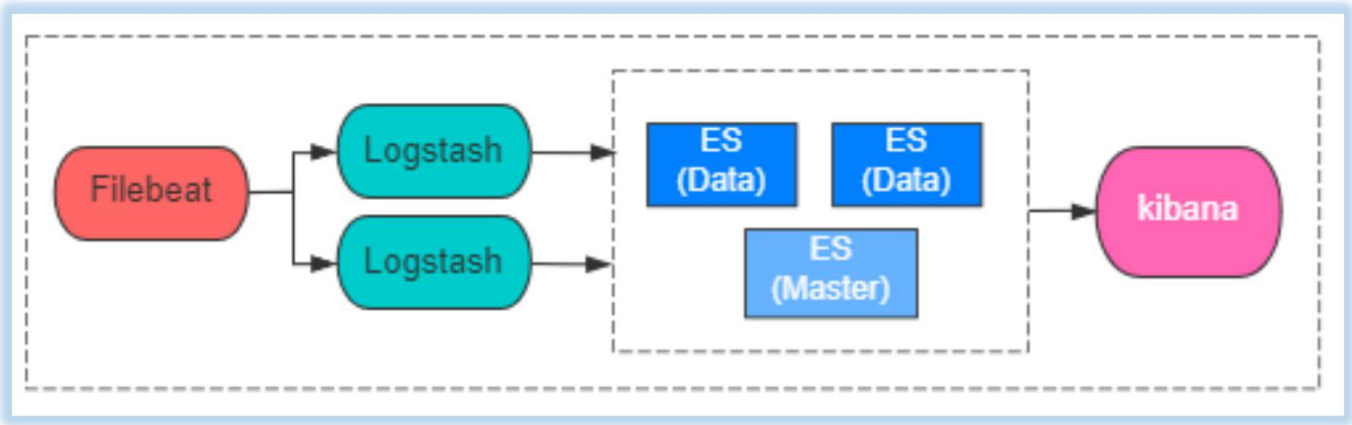
PRAT

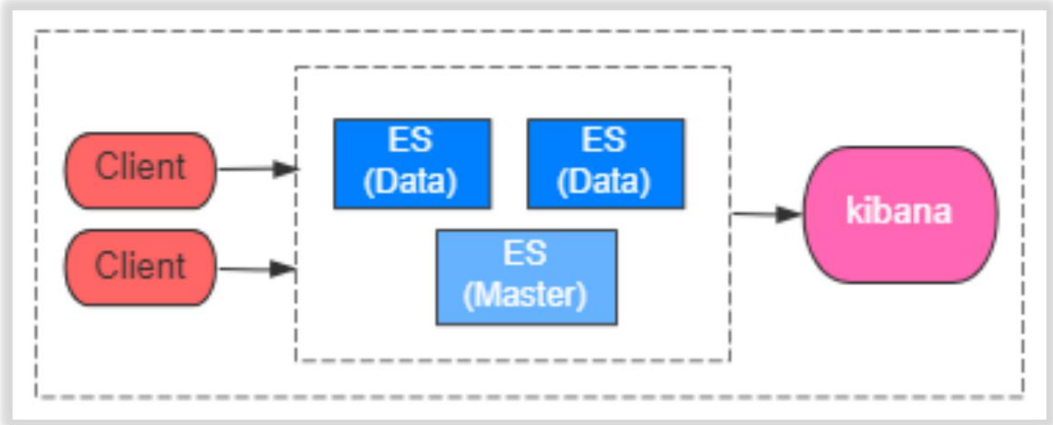
03 应用实践

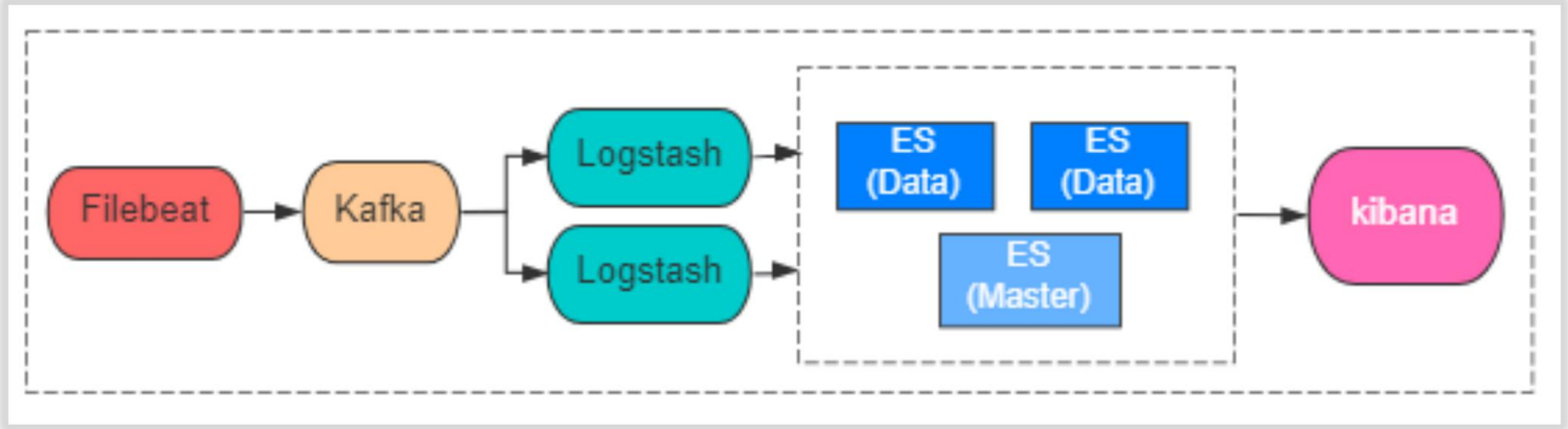


日志搜索与分析

- 日志输入：Filebeat/Syslog/Kafka/SDK/UDP/TCP/Mongo/其他平台

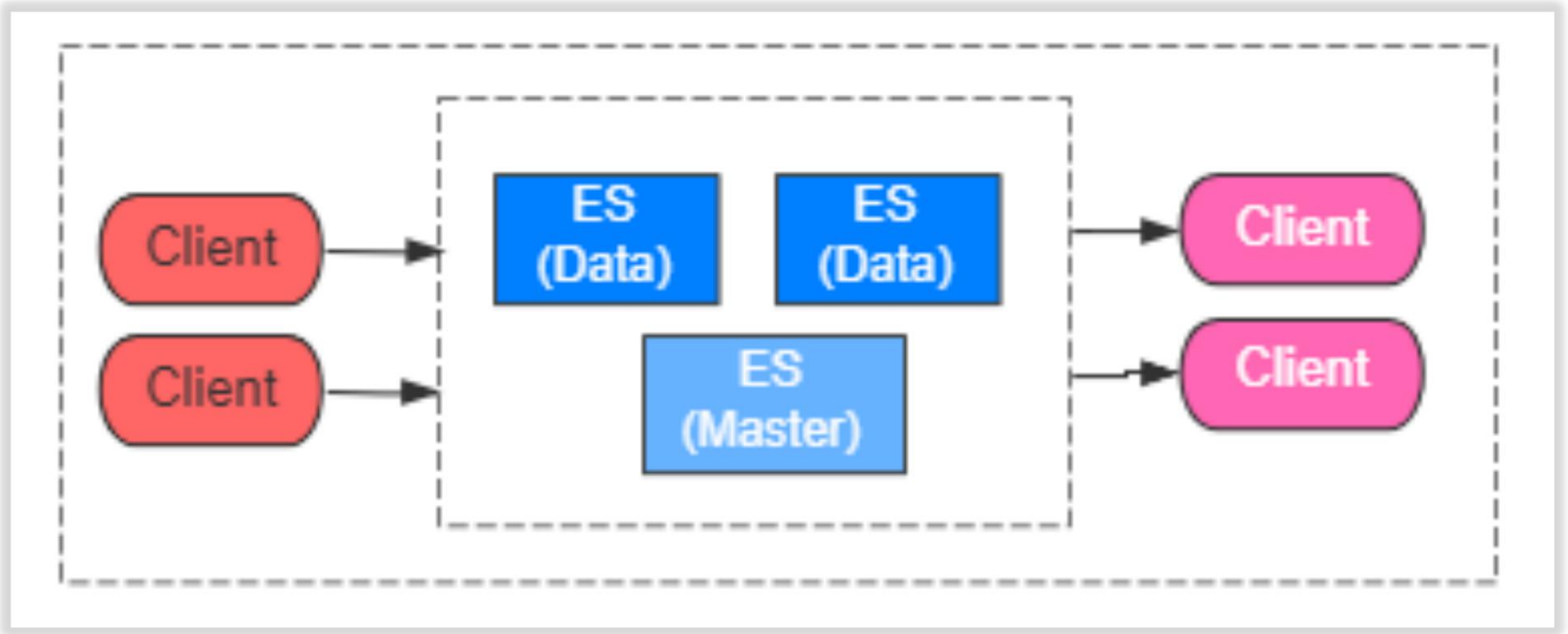






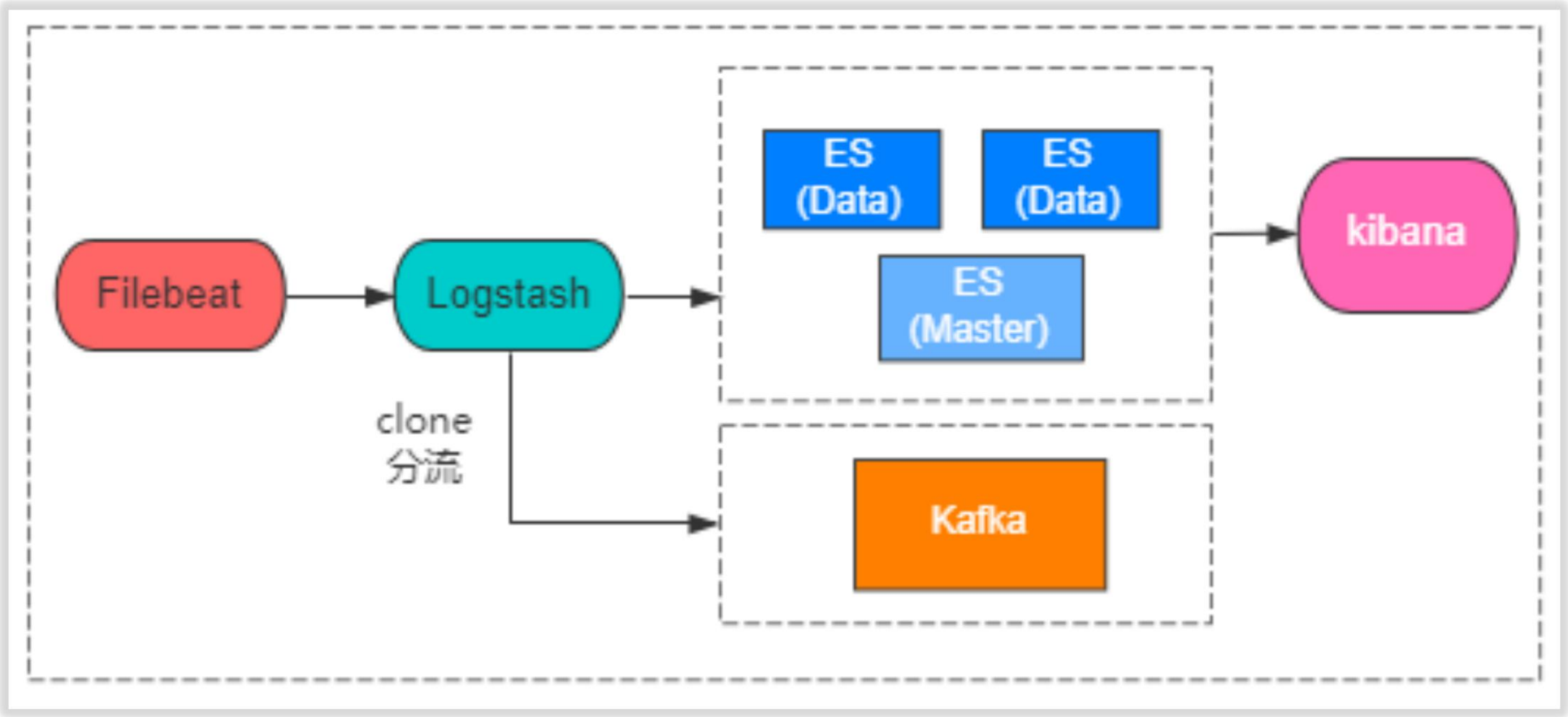
业务搜索

- 数据输入：SDK/MySQL/Mongo/其他平台/业务系统



清洗后数据分流

- Logstash filter clone插件进行事件拷贝



用户场景

- 成本预算有限，申请集群规模小，期望较好的读写性能
- 小索引较多，分片较多，集群cpu、JVM告警，但用户不扩容
- 不做集群规模评估，申请cpu、内存和存储比例不协调
- 不做数据生命周期管理，存储告警不处理，触发自动保护，无法写入
- 不做索引字段类型定义和约束

集群规格参考列表

规格	最大节点数	单节点最大磁盘（查询）	单节点最大磁盘（日志）	单节点最大磁盘（通常）
2C 4G	8	40 GB	200 GB	120 GB
2C 8G	8	80 GB	400 GB	200 GB
4C 8G	16	80 GB	400 GB	200 GB
4C 16G	16	160 GB	800 GB	512 GB
8C 16G	32	160 GB	800 GB	512 GB
8C 32G	32	320 GB	1600 GB	1000 GB
16C 32G	50	320 GB	1600 GB	1000 GB
16C 64G	50	640 GB	2000 GB	2000 GB

说明：

- 目前生产讯飞云ES集群申请页面中支持的最大节点数是50个。
- 写多读少的场景建议选择cpu:内存 = 1:4规格，写少读多的场景建议选择cpu:内存 = 1:2规格。

实践典型问题

- **问题1**：以上问题？

  解决：协助用户做集群规模评估和索引数据建模。后台手动处理。

- **问题2**：集群red，分片unassigned ？

  解决：GET _cluster/allocation/explain

  POST _cluster/reroute?retry_failed=true

  指定分片reroute。

- **问题3**：bulk写入拒绝，返回429？

  解决：写入拒绝告警，提醒用户降低并发写入或扩容集群。

- **问题4**：手动段合并POST /index/_forcemerge导致索引无法写入？

  解决：段合并，先写入新segment，然后删除老的segment，会消耗大量的资源和磁盘空间，执行前预估磁盘空间充足。

实践典型问题

- **问题5**：LockObtainFailedException？

原因：在主节点重启时，有线程正在对某个shard做bulk或者scroll等长时间的写入操作。等主节结点重新加入集群的时候，由于shard lock没有释放，master无法allocate这个shard。

解决：1） POST _cluster/reroute?retry_failed=true

2）找到这个锁住的线程，并释放掉

3）从master日志中找到锁住shards所在的ES实例，重启实例，释放内存锁。

- **问题6**：Scroll查询导致FullGC？

原因：scroll search context持有lucene的searcher不释放，导致段合并后一些leafreader，fst内存数据结构常驻JVM，长期运行触发FullGC。

解决：GET /_nodes/stats/indices/search

DELETE /_search/scroll/_all

实践典型问题

- **问题7**：GET _cluster/allocation/explain，出现"allocate_explanation": "cannot allocate because all found copies of the shard are either stale or corrupt"?

  原因：由于有节点离线或文件描述符耗尽，导致副本损坏，无法用副本恢复主分片。

  解决：1）GET _cat/allocation?v查看是哪个分片不正常

  2）GET _shard_stores?pretty查看不正常分片存储在那个节点

  3）将陈旧的分片分配为主分片allocate_stale_primary ，此时会丢失部分数据。

- **问题8**：ES 安装pinyin分词插件写入性能明显下降？

  原因： pinyin分词插件bug导致cpu耗尽，如输入较长连续数字字符串，进入深度递归。

  解决： https://github.com/medcl/elasticsearch-analysis-pinyin/issues/234 。

- **问题9**：集群总数据量较少，读写量较少，但CPU和JVM利用率较高？

  原因：集群中小索引较多，导致小shards和segment较多，需要维护较大内存数据结构。

  解决：手动merge segment，cpu使用下降，old gc次数和延迟明显下降。

04

PRAT 04

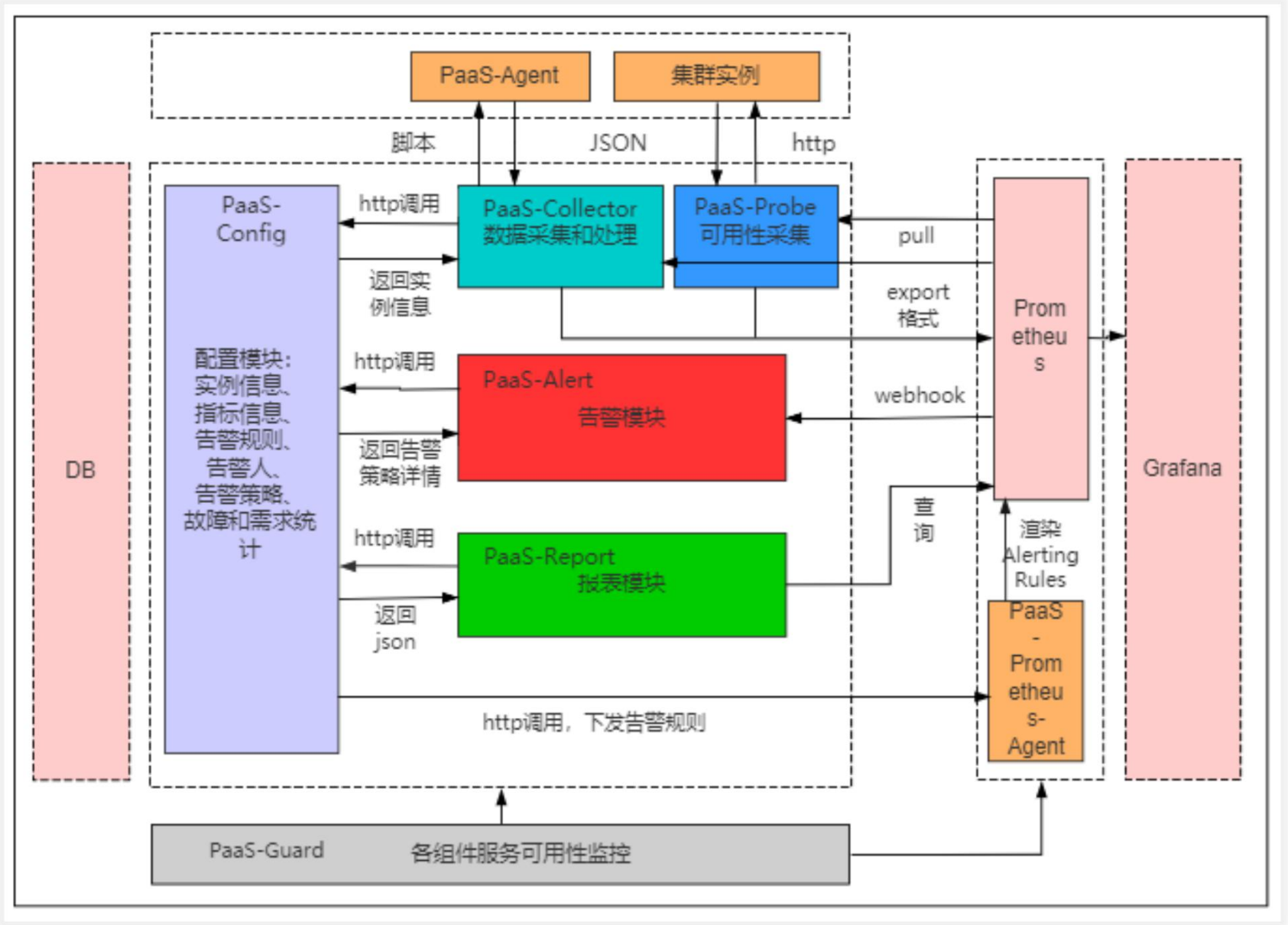
有效运维

有效的监控和告警指标定义

维度	指标	告警值
公共	CPU、内存、节点可用性、集群可用性	90%、90%、0、0
ES集群	集群yelow状态、集群red状态	2、3
ES集群	总索引请求率、总索引延迟、总搜索请求率、总搜索延迟	-
ES节点	存储使用率、JVM使用率、写入拒绝率	85%、85%、0
ES节点	索引请求率、索引延迟、搜索请求率、搜索延迟、Segment总数	-
ES节点	OldGC平均次数、OldGC延迟、YoungGC平均次数、YoungGC延迟	-
Logstash节点	JVM使用率、事件接受率、事件发送率、Queue使用率	85%、0、0、50%
Logstash节点	Queue事件数、事件延迟	-

服务保障系统

- 多可用区互监控
- 基础信息多可用区同步
- 集群信息自动同步
- 默认告警策略自动添加
- 默认告警规则自动下发
- 告警策略规则自由定义
- 支持短信、邮件和微信
- 秒级延迟
- 月、周、天报表



05

PRAT 05

未来规划

认证授权

1、提高数据安全性

业务数据告警

1、助力业务故障感知

数据迁移工具

1、快速数据迁移
2、助力业务上云

跨机房容灾

1、提高集群可用性

谢谢!

Q&A

科大讯飞 iFLYTEK | elastic 中文社区