vivo数据集成稳定性与数据质量保障及可观测实践

vivo互联网 大数据架构师/ 易龙

vivo

目录

- vivo数据集成平台架构及功能
- vivo数据集成稳定性保障实践
- vivo数据集成链路数据质量保障实践
- vivo数据集成可观测实践

vivo

vivo数据集成平台架构及功能

产品能力地图

Bees，是vivo的一站式数据集成平台，它支持将多场景下多样化、分散的数据源，统一汇聚到大数据存储，是数据流入大数据体系的一座桥梁。

数据量条数
万亿级/日

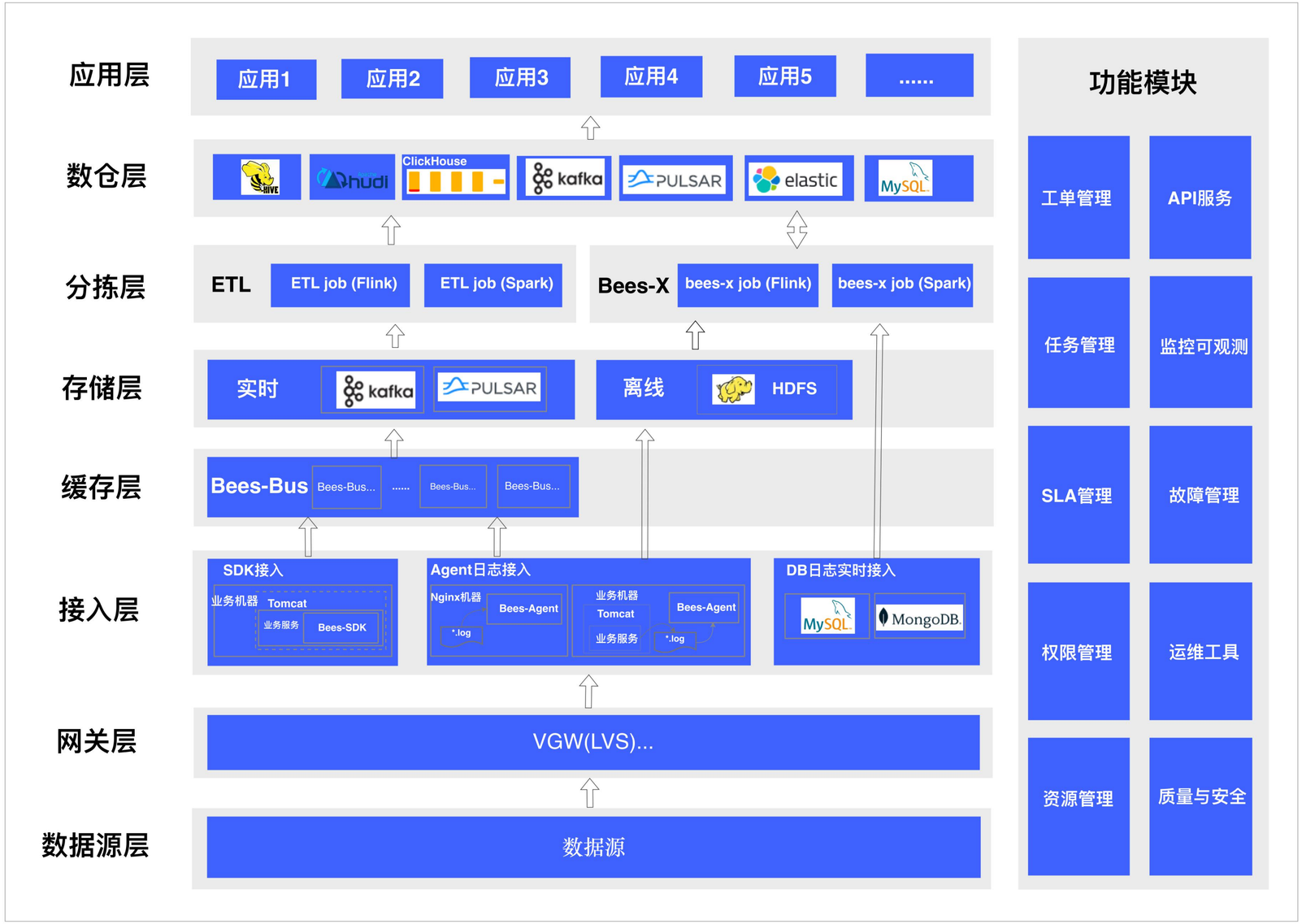
数据量大小
PB级/日

可用性
99.99%

数据完整性
99.999999%

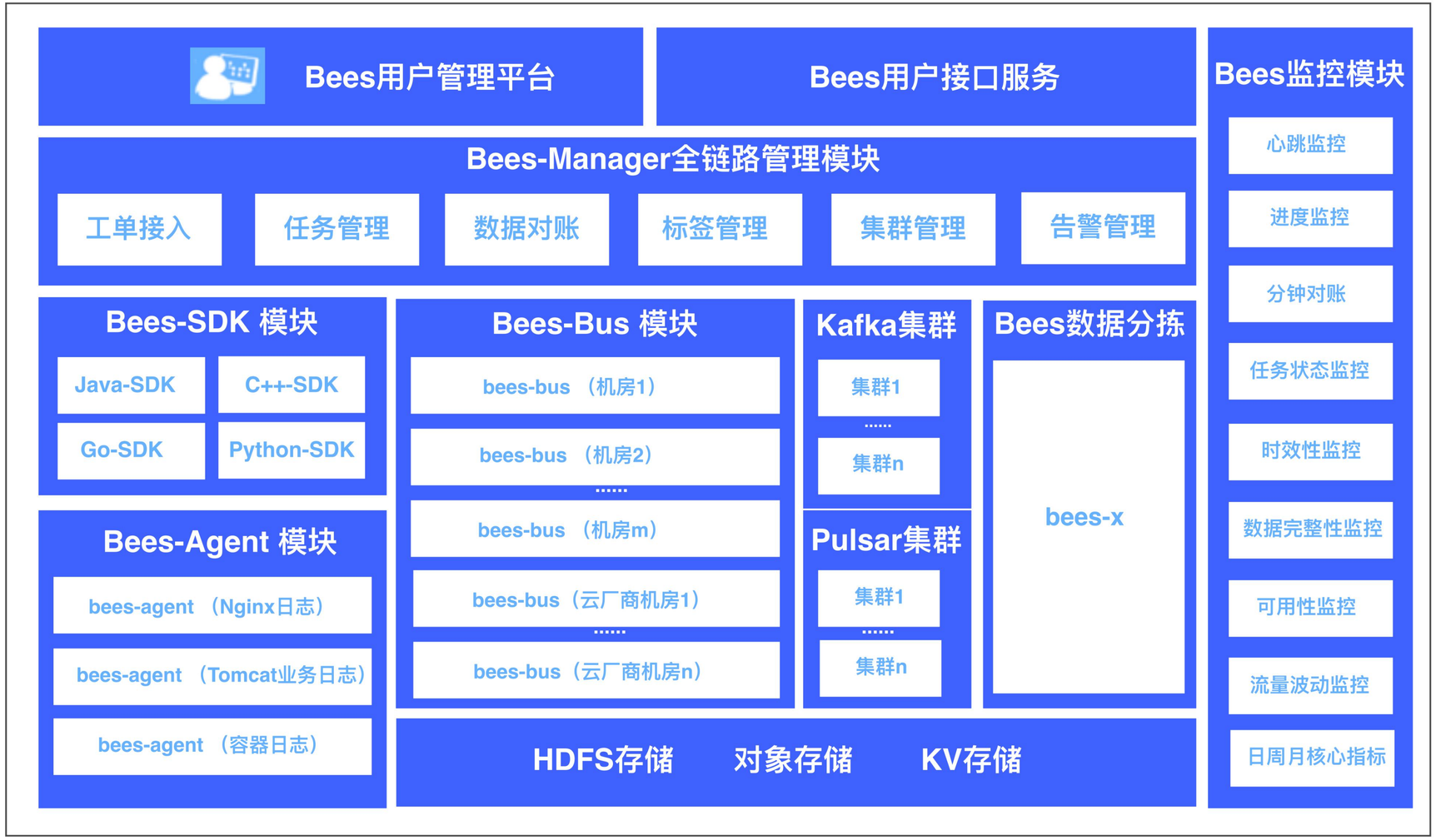
数据时效性
500ms

分层架构图


应用层
应用1
应用2
应用3
应用4
应用5
.....
数仓层
ClickHouse
kafka
PULSAR
elastic
MySQL
分拣层
ETL
ETL job (Flink)
ETL job (Spark)
Bees-X
bees-x job (Flink)
bees-x job (Spark)
存储层
实时
kafka
PULSAR
离线
HDFS
缓存层
Bees-Bus
Bees-Bus...
.....
Bees-Bus...
Bees-Bus...
接入层
SDK接入
业务机器
Tomcat
业务服务
Bees-SDK
Agent日志接入
Nginx机器
Bees-Agent
*.log
业务机器
Tomcat
Bees-Agent
业务服务
*.log
DB日志实时接入
MySQL
MongoDB
网关层
VGW(LVS)...
数据源层
数据源
功能模块
工单管理
API服务
任务管理
监控可观测
SLA管理
故障管理
权限管理
运维工具
资源管理
质量与安全

核心组件架构图

- Bees监控模块
  - 监控、指标展示与告警
- Bees-Manager
  - 工单接入管理
  - 任务管理
  - 采集配置管理中心
  - 用户平台服务【极重要】
- Bees-SDK
  - 数据接入 SDK 工具包
- Bees-Agent
  - 源端日志接入组件
  - 部署在业务机器
  - 影响CPU、内存、文件句柄、IO
- Bees-Bus
  - 数据传输管道服务【极重要】
- Bees-X：实时数据同步服务
  - 支持binlog日志采集
  - mongdb oplog实时采集
  - 支持其他异构数据源数据同步

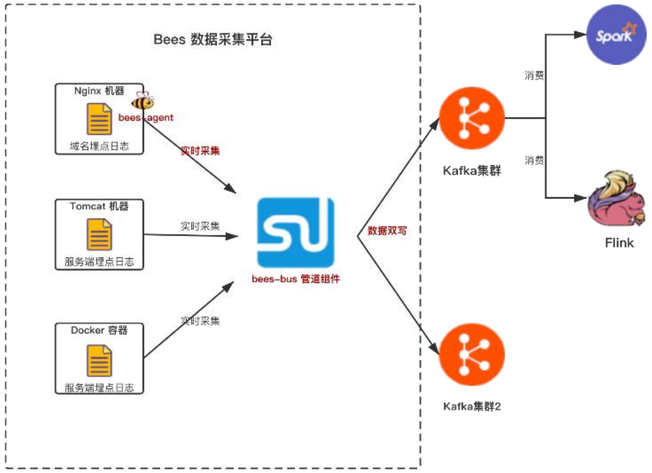


核心功能介绍

vivo

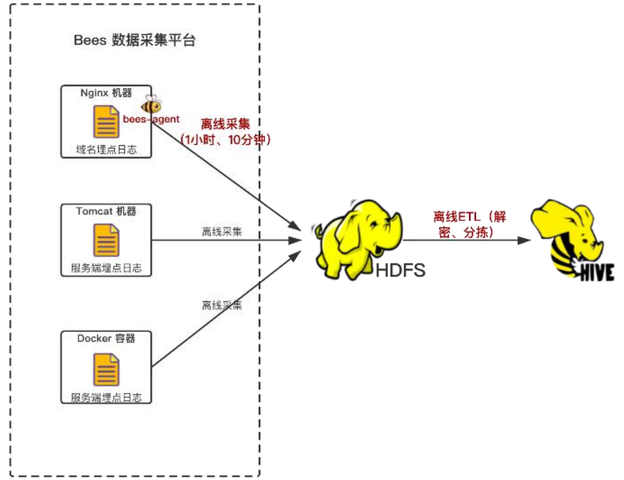
实时日志接入

- Nginx/Tomcat/埋点日志
- 支持容器服务日志
- 传输到Kafka（500ms内）
- 支持过滤
- 支持同时写多Kafka
- 业务隔离



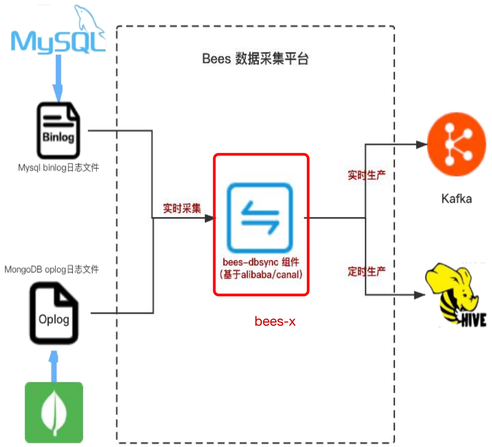
离线日志接入

- Nginx/Tomcat/埋点日志
- 支持容器服务日志
- 按小时粒度批传输
- 按10分钟粒度批传输
- 支持限速



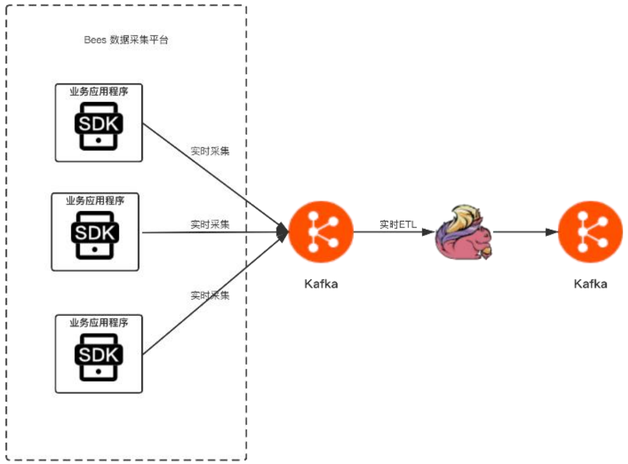
DB全增量日志实时接入

- 支持接入到 Kafka、Pulsar
- 支持接入到 Hive、CK等
- 对主库无性能影响
- 保障秒级别时延
- 支持指定点位进行数据续传



SDK数据接入

- 业务数据无需落地日志
- 更低的时延（毫秒级）
- 支持 Avro、Thrift 协议
- 支持 Java、C++ 语言



核心问题及挑战

核心问题维度

- 链路稳定性
- 链路数据质量
- 链路可观测性

痛点问题

- 被动接收告警，问题定位恢复慢
- 散点式救火，运维成本高
- 数据产出时效性波动大
- 告警多而杂，处理成本高

核心挑战

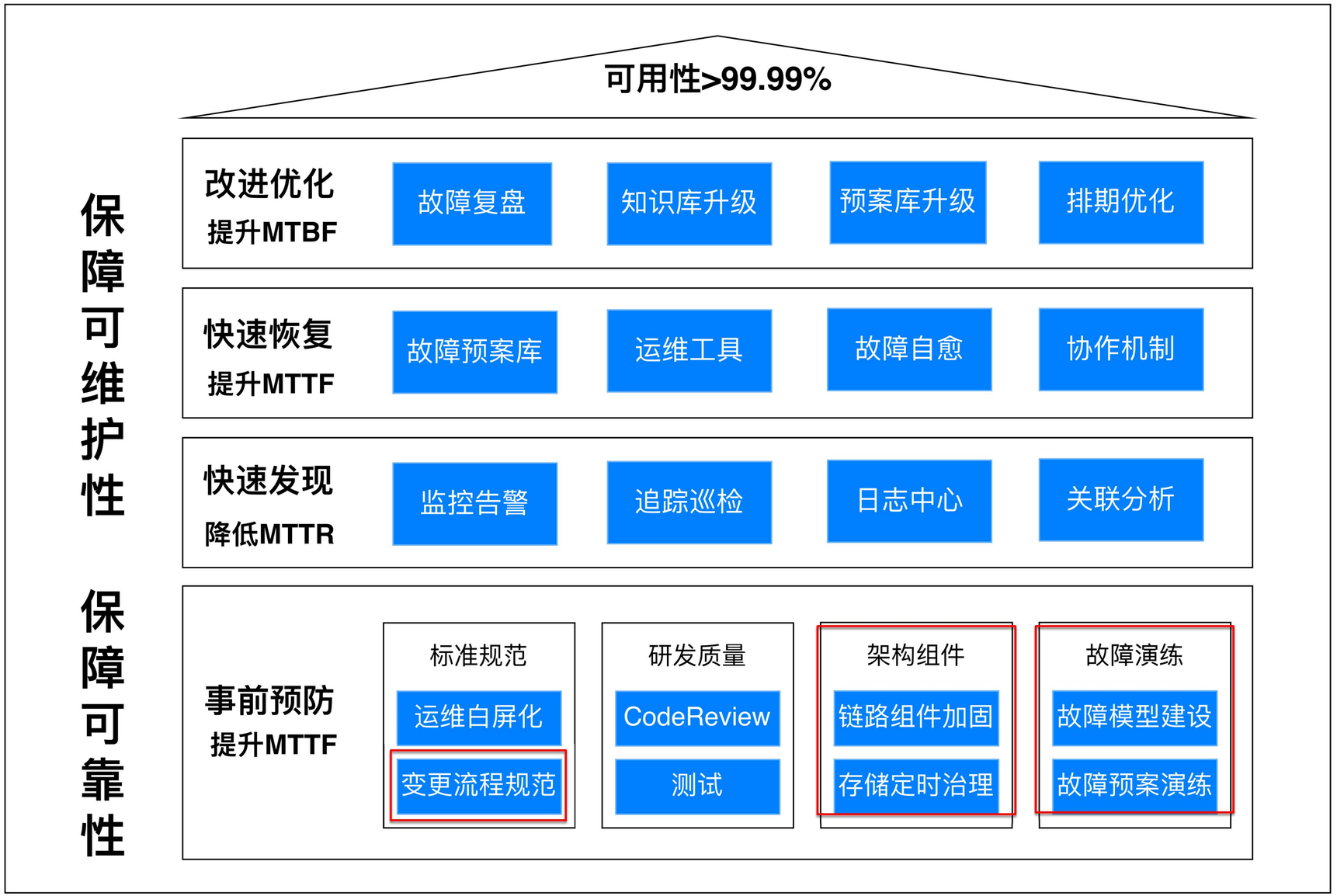
- 如何从**根本上长效的**保障稳定性
- 如何从全**链路视角**保障数据时效性
- 如何**有效准确的**告警并**快速恢复**

vivo数据集成稳定性保障实践

稳定性保障整体方案

vivo

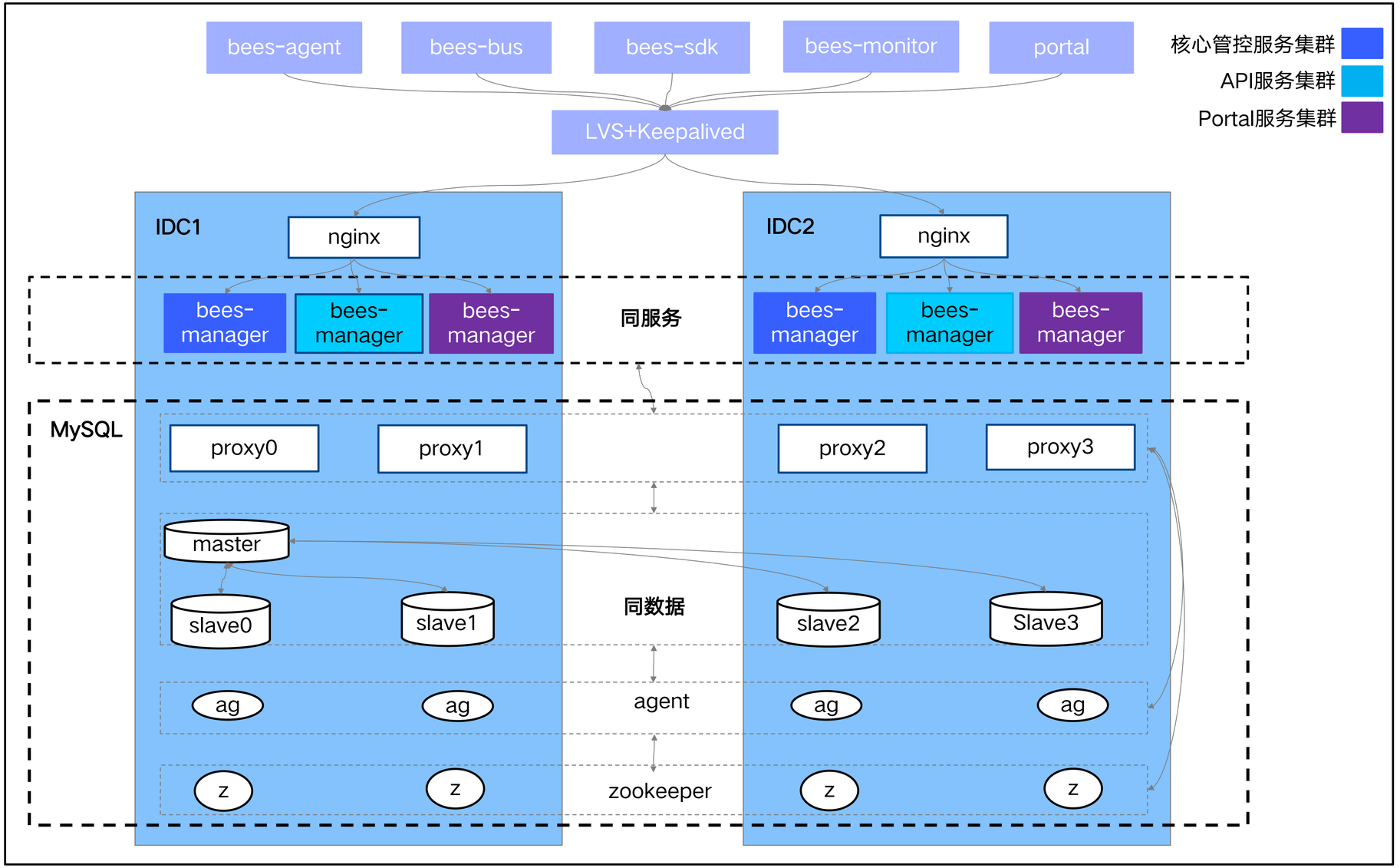
MTBF:（Mean Time Between Failures），平均故障间隔时间
MTTF:（Mean Time To Failure），平均无故障时间
MTTR:（Mean Time To Repair），平均修复时间



架构组件：核心服务&存储 多活高可用

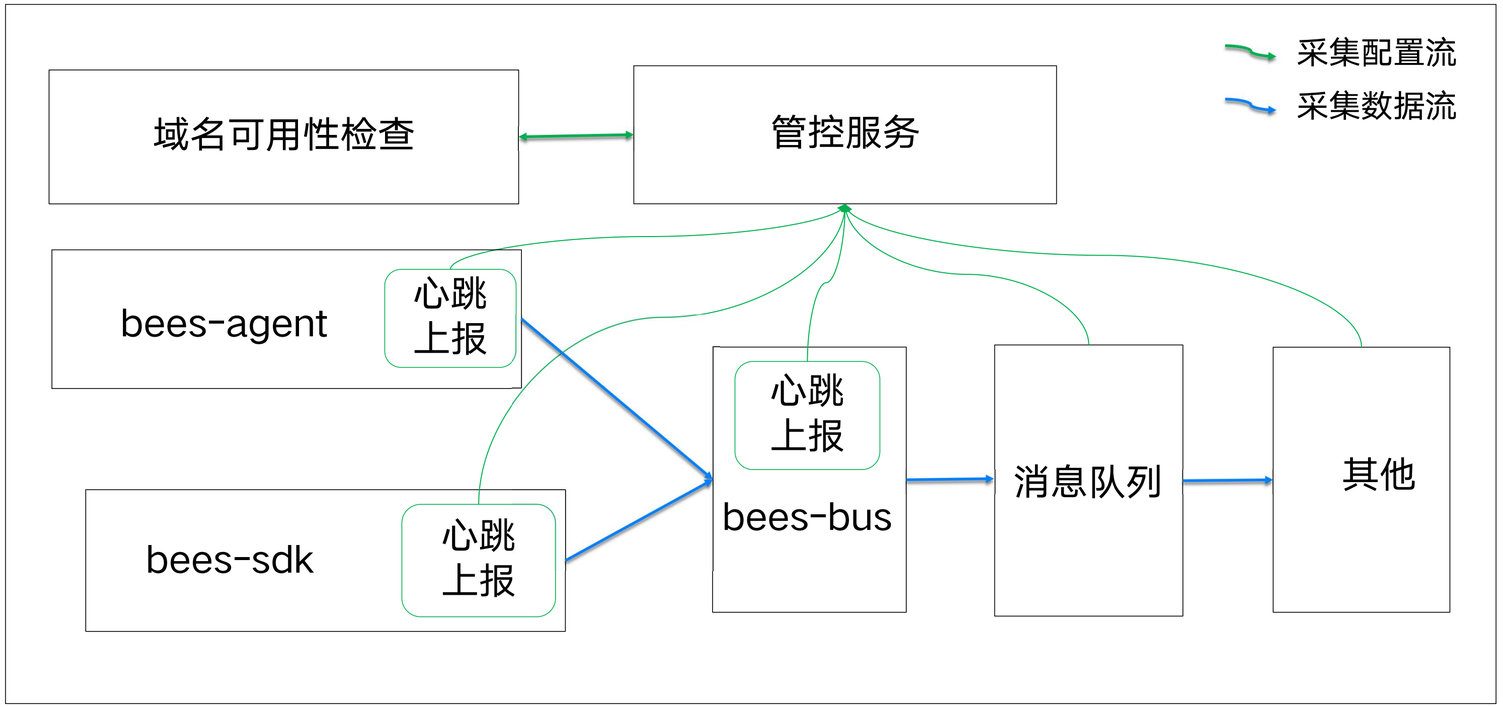
vivo

- 核心服务多活高可用
  - 服务拆分多节点部署
  - 跨机房容灾
- 存储多活高可用
  - 跨机房容灾
  - Proxy，无中心集群，支持高可用
  - Agent，基于Raft选主，支持高可用
  - 节点均支持动态扩缩容
  - Proxy配置基于Zookeeper进行同步，保障一致性



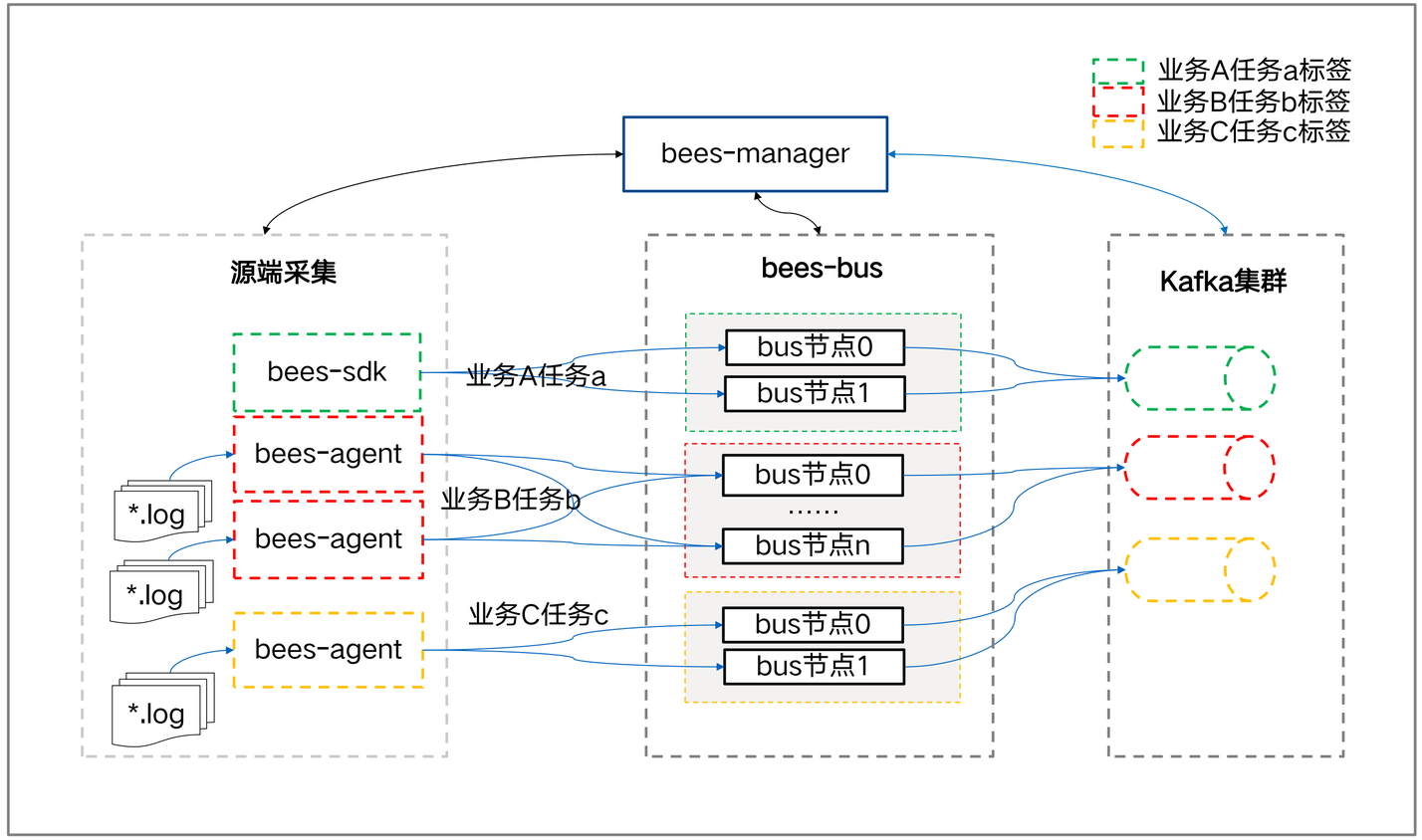
架构组件：核心组件支持健康检查

- 链路核心组件心跳上报
- 异常及时发现，追数补数



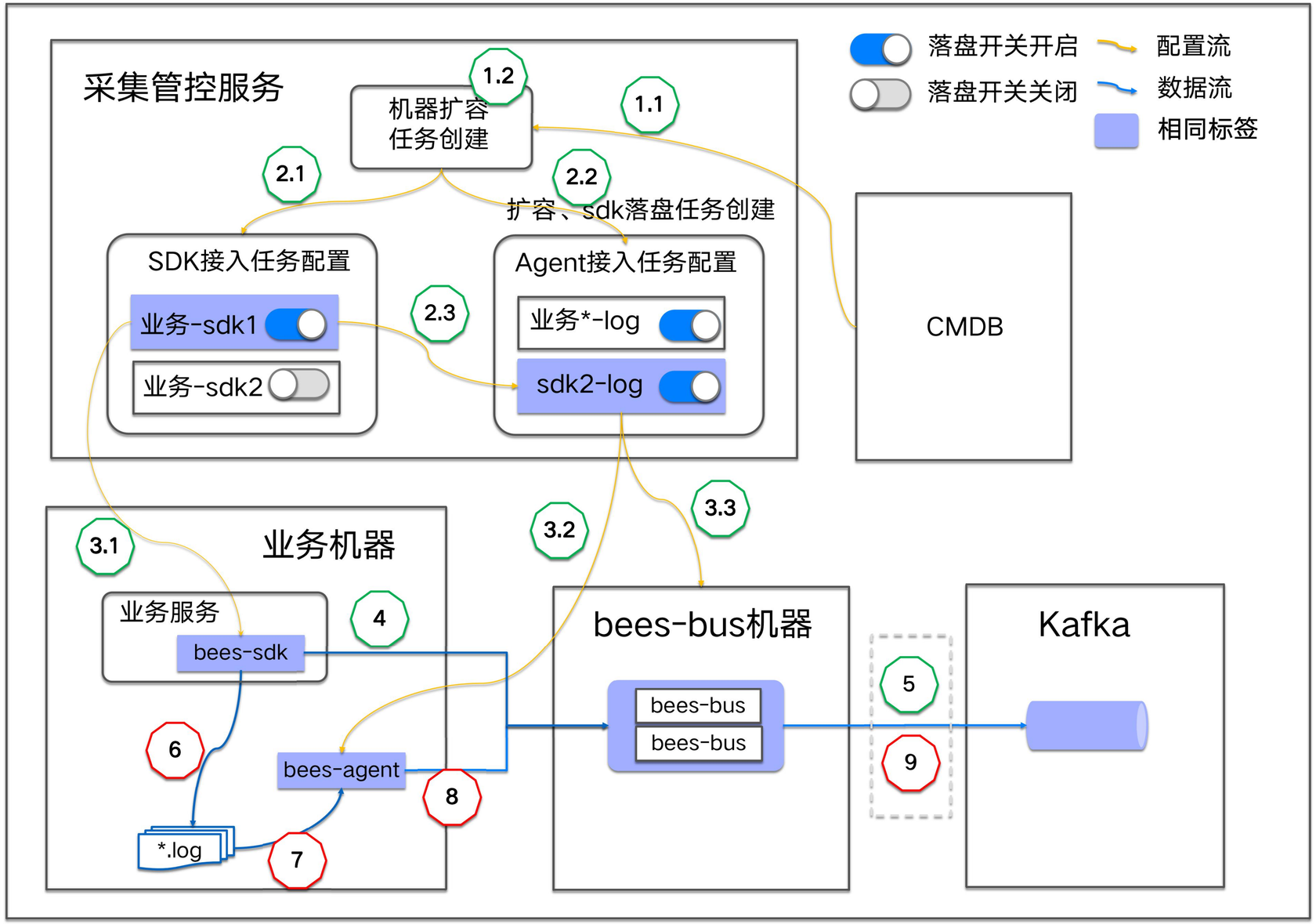
架构组件：物理标签隔离机制

- 标签统一通过bees-manger管理
- 不同业务任务分配不同标签
- 按标签和bees-bus建立连接
- bees-bus使用大内存物理机器
- 同一台bus机器负责一个业务
- bees-bus备机池，及时扩容



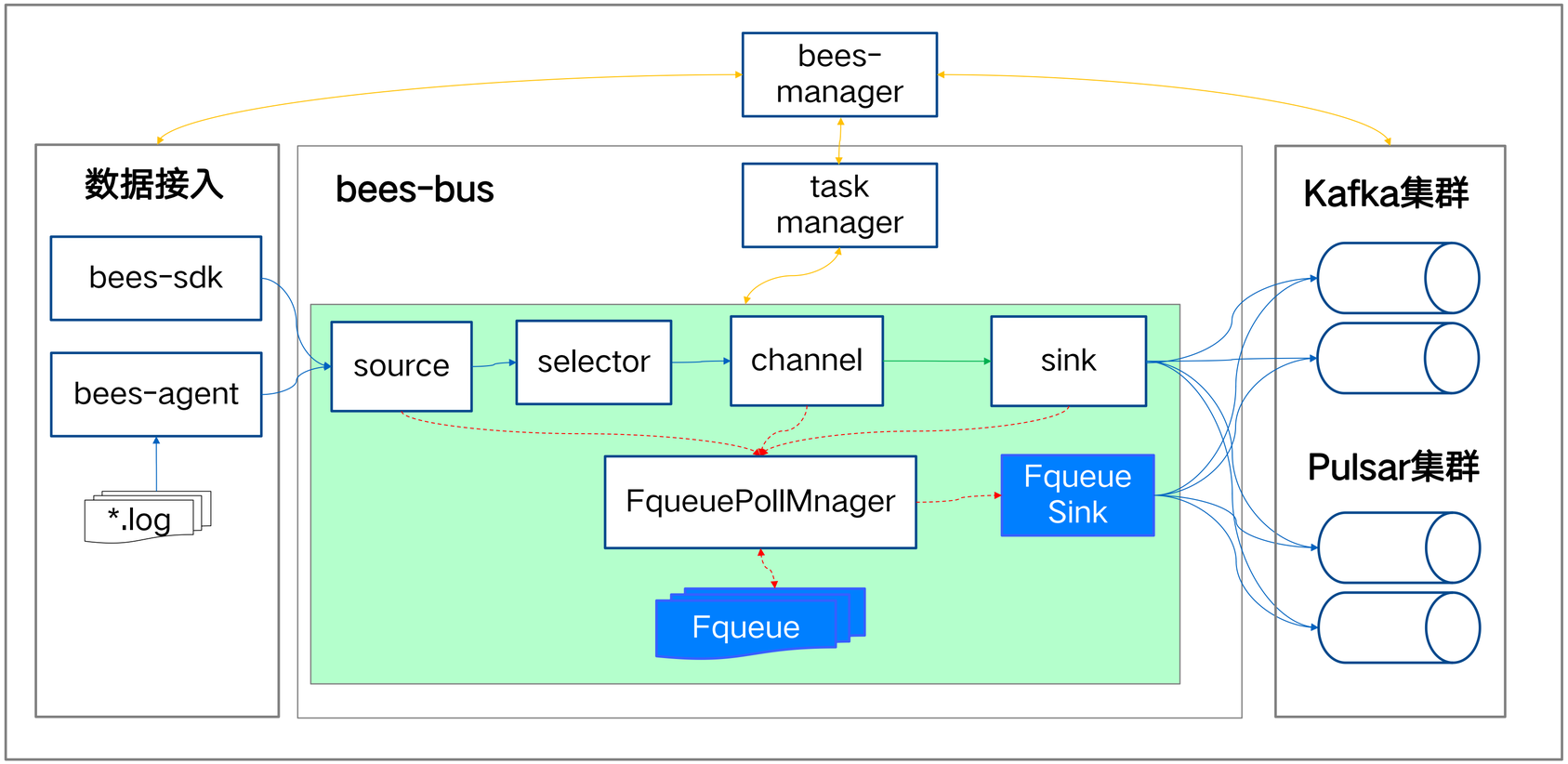
架构组件-实时链路容灾： SDK落盘重发机制

- 平台化配置管控
- 配置动态感知
- 支持落多目录多文件



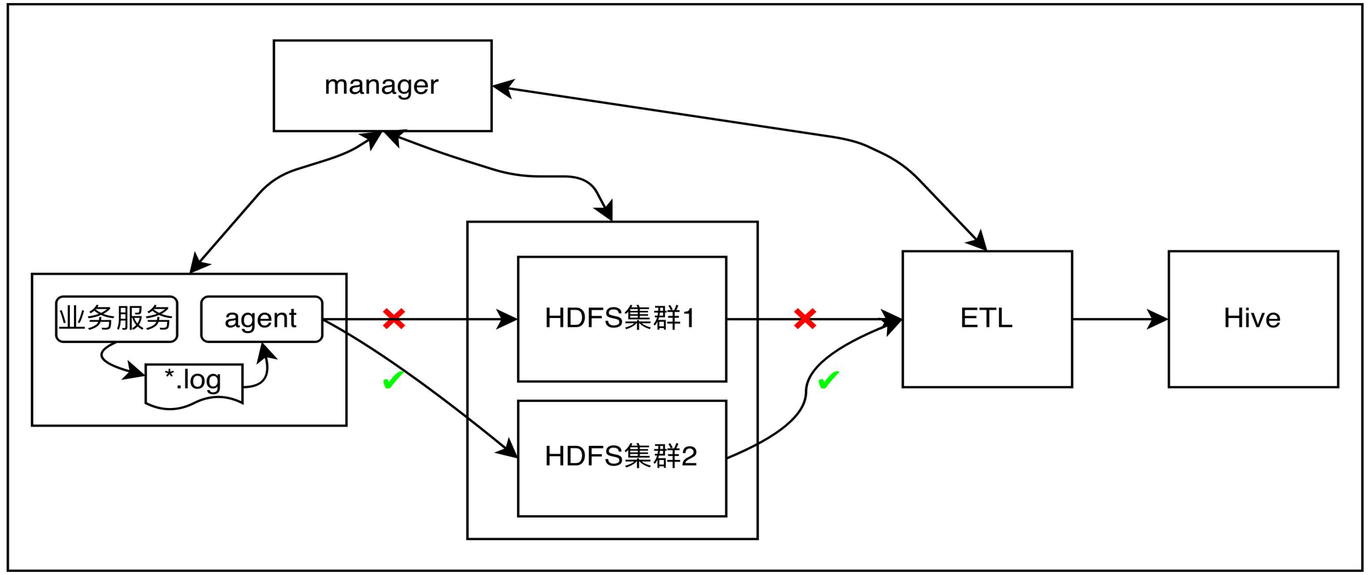
架构组件-实时链路容灾：数据反压缓存动态落盘重发

- 上下游联动，及时感知异常
- 全链路流量波动监控
- 及时数据反压告警
- 引入Fqueue落盘
  - 支持顺序写落盘
  - 支持落单盘和多盘
  - 独立FqueueSink隔离发送

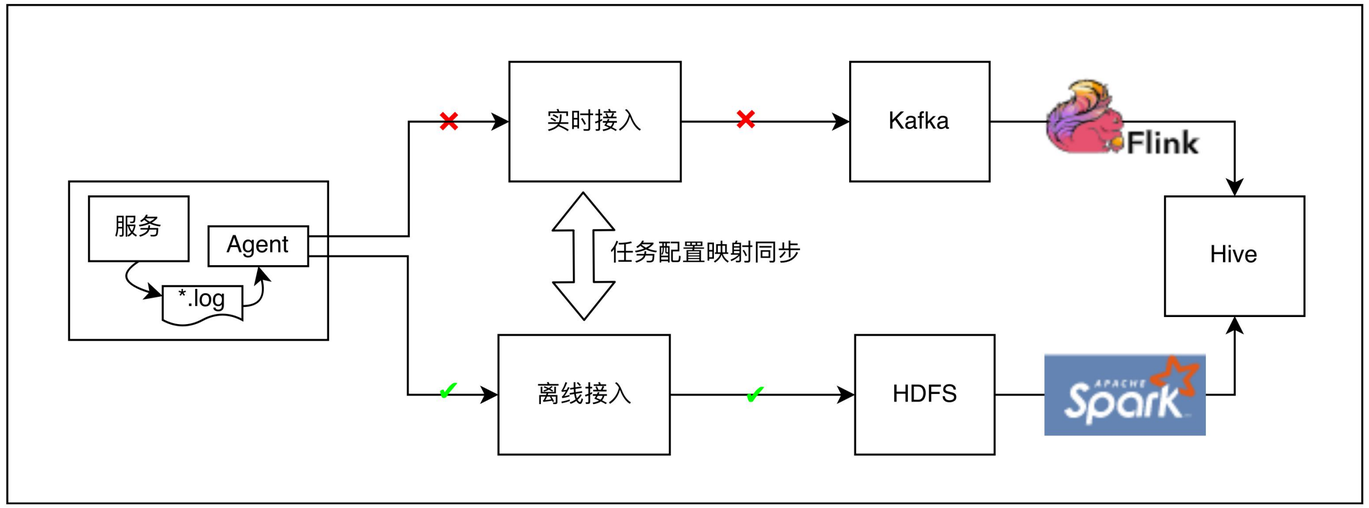


架构组件：离线链路写HDFS主备切换 & 双链路容灾快速切换

- 离线HDFS集群容灾能力
- 上下游联动
- 分钟级切换耗时



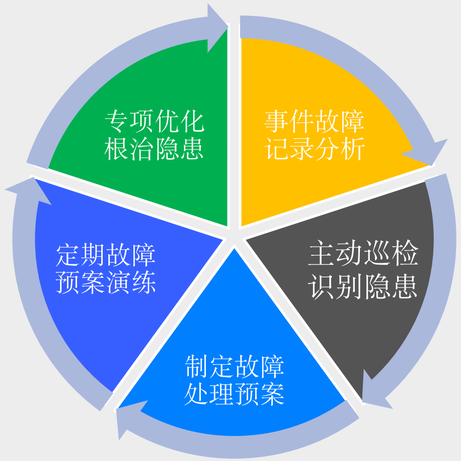
- 核心SLA业务
- 容灾触发切换
- 分钟级切换耗时



链路故障演练

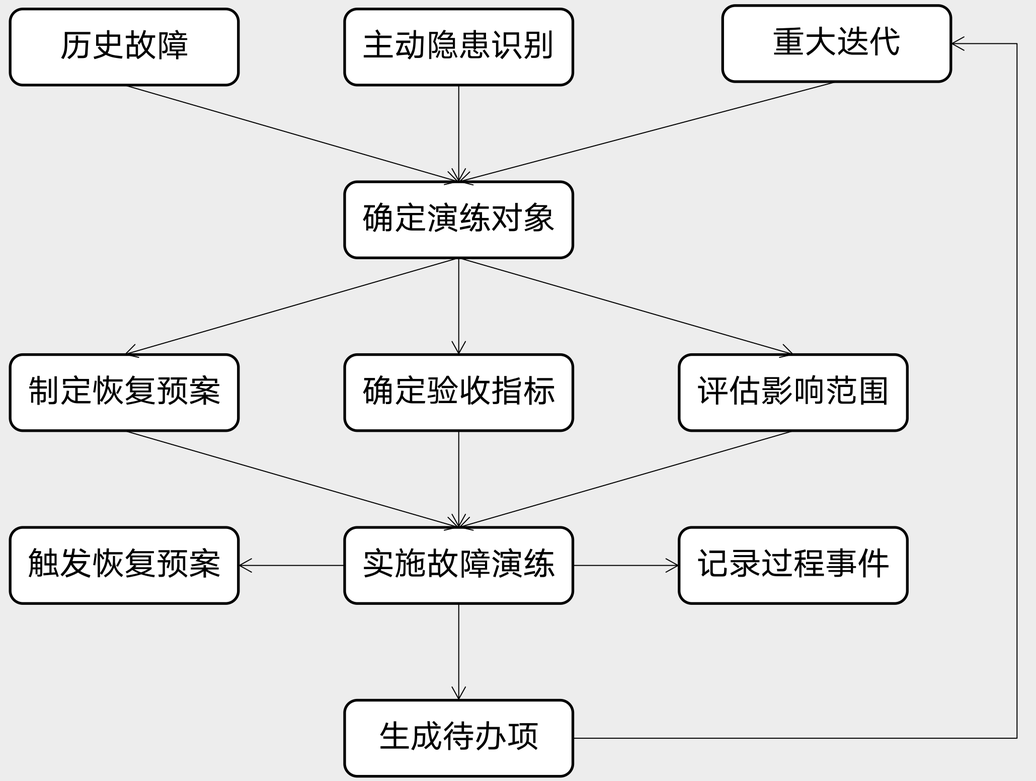
vivo

思路



平台

故障演练步骤



稳定性保障：规范变更发布流程

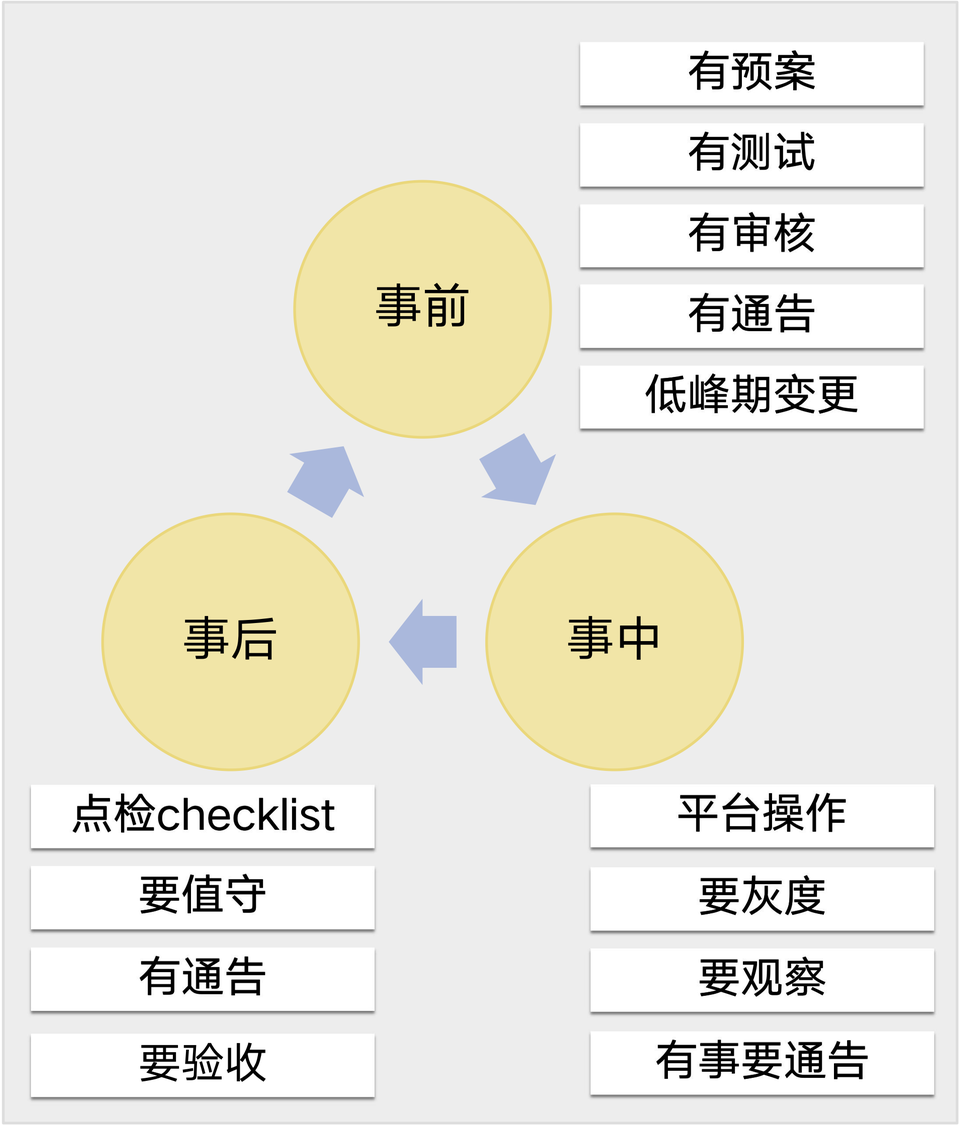
vivo

基础准备

变更类型梳理
- 常规配置
- 大版本升级
- 常规DB变更
- 紧急

- 变更步骤梳理
- 通知范围
- 监控告警
- 值班规范
- 点检checklist
- 协作流程
- 风险应对措施
- 回滚方案

变更发布原则



梯度灰度关键步骤

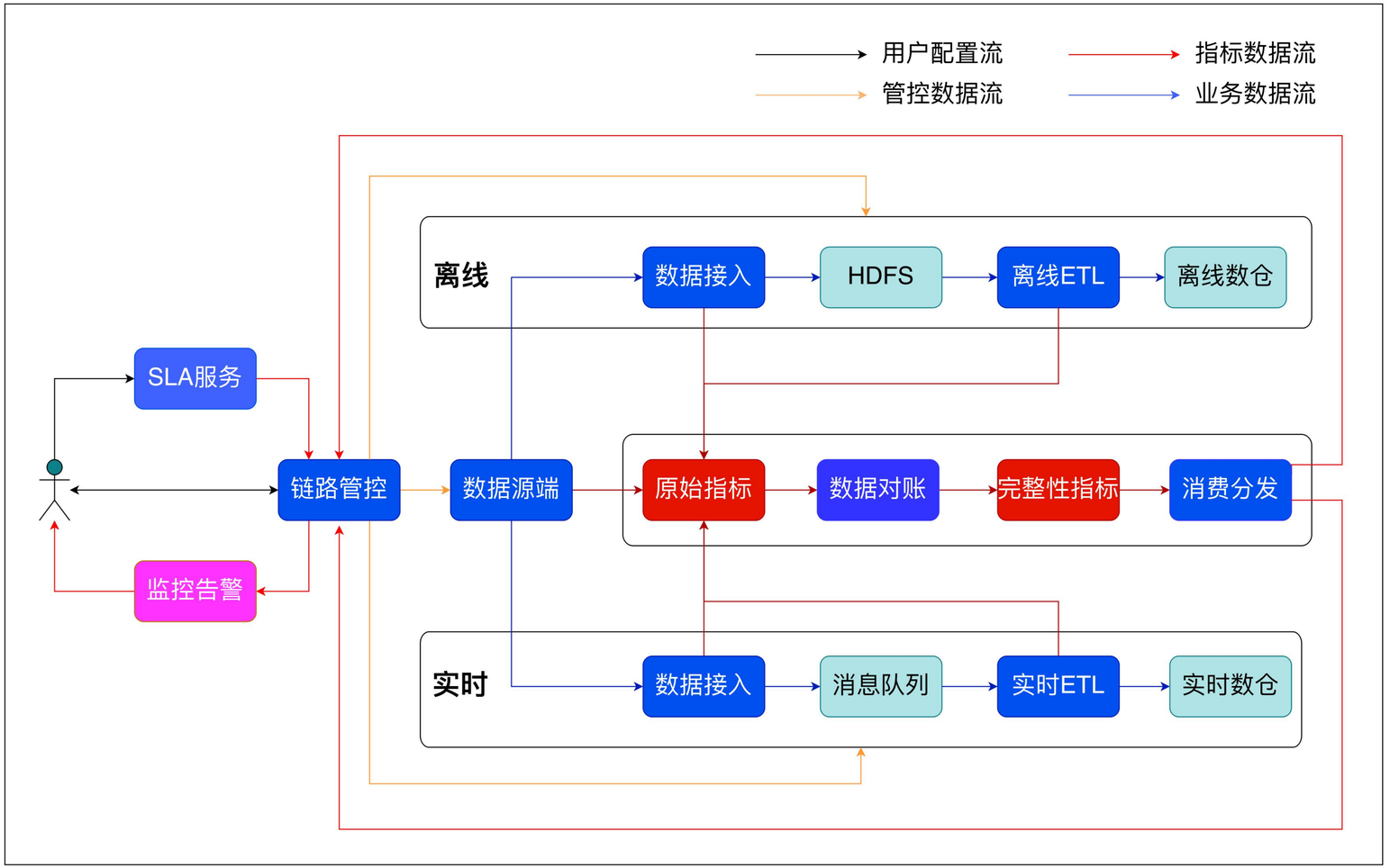
- 明确版本功能
- 明确灰度指标
- 确定灰度管控策略
- 稳定性验证（种子用户）
- 低频用户推广
- 核心用户推广
- 高频用户推广
- 全网用户推广

vivo数据集成
链路数据质量保障实践

链路数据质量-数据完整性

支持双链路数据对账、链路关键卡点校验、发现异常并追数补数，保障SLA业务数据完整性要求

- 支持核心SLA业务离线实时双接入
- 支持多种数据对账方式
  - 离线全链路对账
  - 实时全链路对账
  - 核心业务双链路对账
- SLA动态保障
- 全链路数据完整性卡点校验
- 备份重接、追数补数



链路数据质量-数据时效性：整体思路

vivo

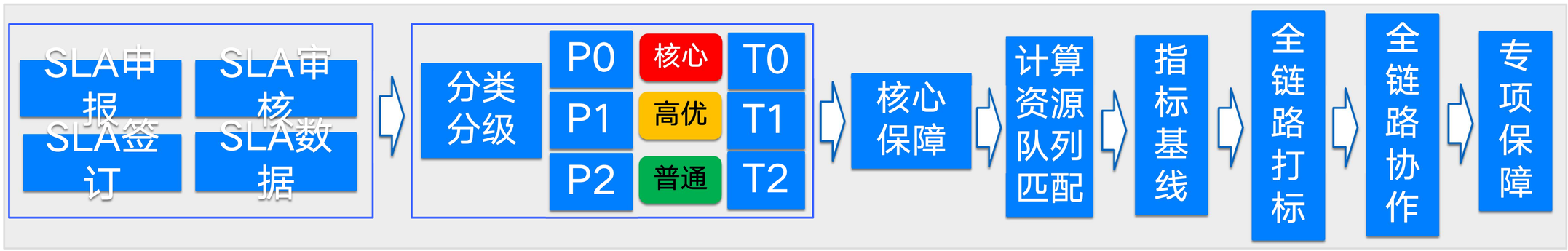
从全链路视角，结合SLA，制定整体措施，保障数据及时产出

优先级：P0>P1>P2
SLA时间：T0<T1<T2

措施

数据上报	网络/服务端	接入传输	ETL	数仓
➢ 实时上报 ➢ 退避重发 ➢ 断点续传	➢ 流量监控 ➢ 波动告警 ➢ CDN可用性 ➢ Nginx监控 ➢ 网络设备监控	➢ Inotify感知 ➢ 轮询发现 ➢ 重采、补采 ➢ 延迟告警 ➢ 断点续传 ➢ 动态扩容	➢ 异常监控 ➢ 容灾切换 ➢ 延时告警 ➢ 任务重跑 ➢ 断点续传	➢ 资源预测 ➢ 北极星指标 ➢ 血缘依赖 ➢ 根因分析 ➢ 动态调度

SLA保障



链路数据质量-数据时效性：Agent及时感知

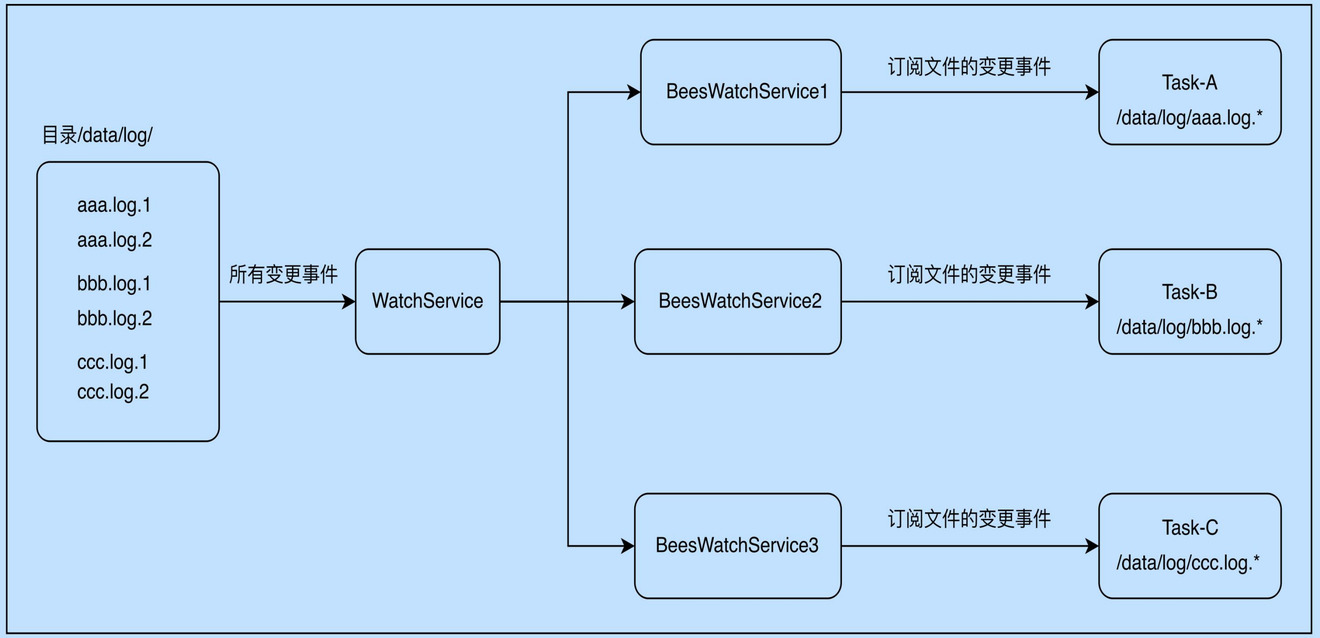
Agent 采用定时轮巡+inotify 组合，时效性达毫秒级

引入inotify时，若直接用jdk的watch service会有**两个问题**：

- 问题1： 日志量TPS很高时，inotify通知频率快，每次agent处理事件，感知的增量日志少，batchSize低，浪费cpu；
- 问题2： JDK提供的WatchService只支持监听目录，不支持监听到具体的文本文件，直接用会出现多任务重复监听单目录，重复越多，cpu浪费越多；

解决方案：

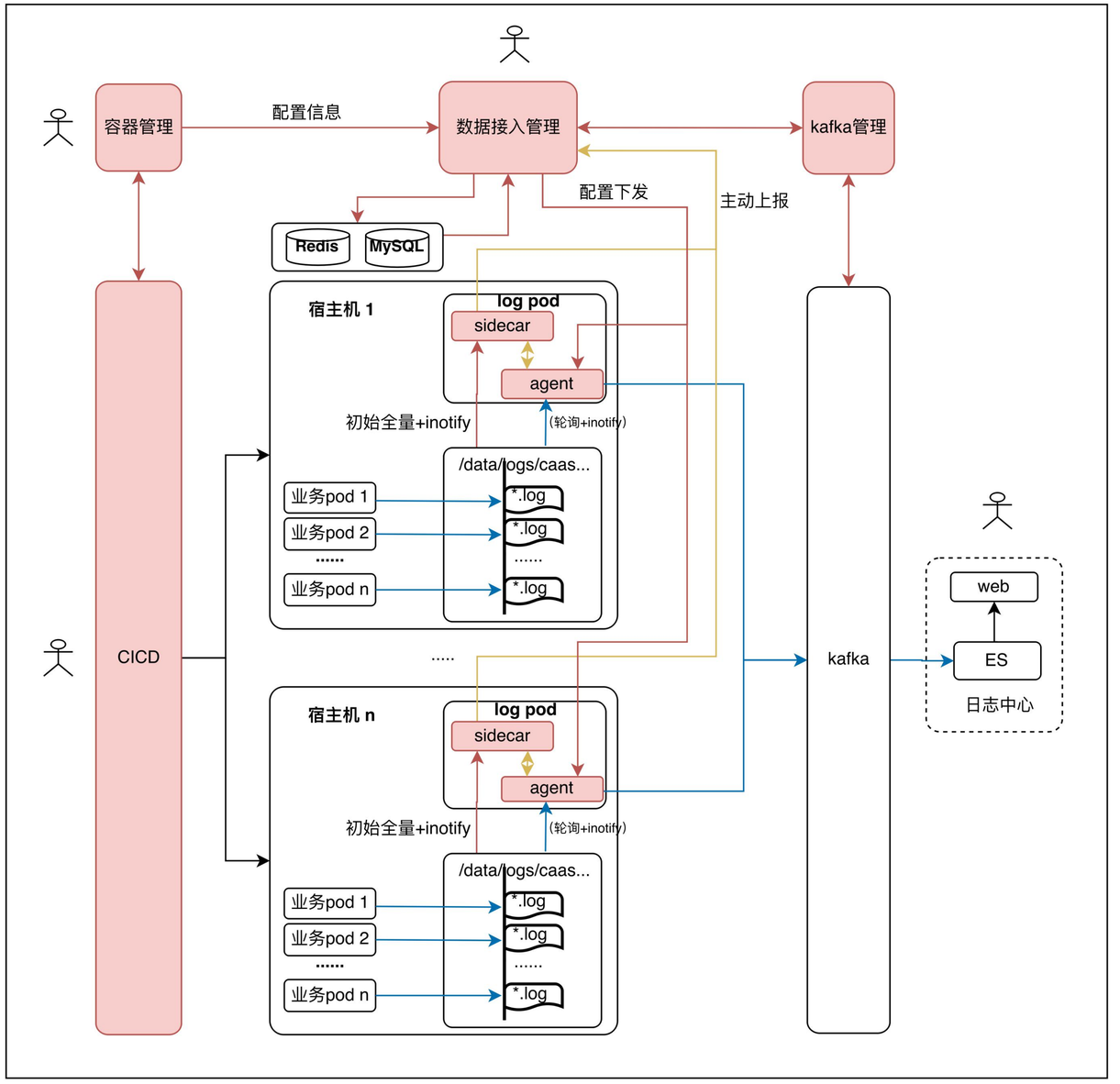
- **问题1方案**：读到日志文件末尾，等待5ms，当batchSize满或 超过5ms 时进行发送；使用该方案，CPU使用率从单核10~20%下降至单核3~6%，下降幅度10%左右；
- **问题2方案**：在WatchService基础上，封装一层支持监听到文件级别的BeesWatchService，实现文件级别监听，每个任务只订阅自己需要的文件，屏蔽了不相干文件的变更事件；



链路数据质量-数据时效性：云原生场景实时日志感知接入

vivo

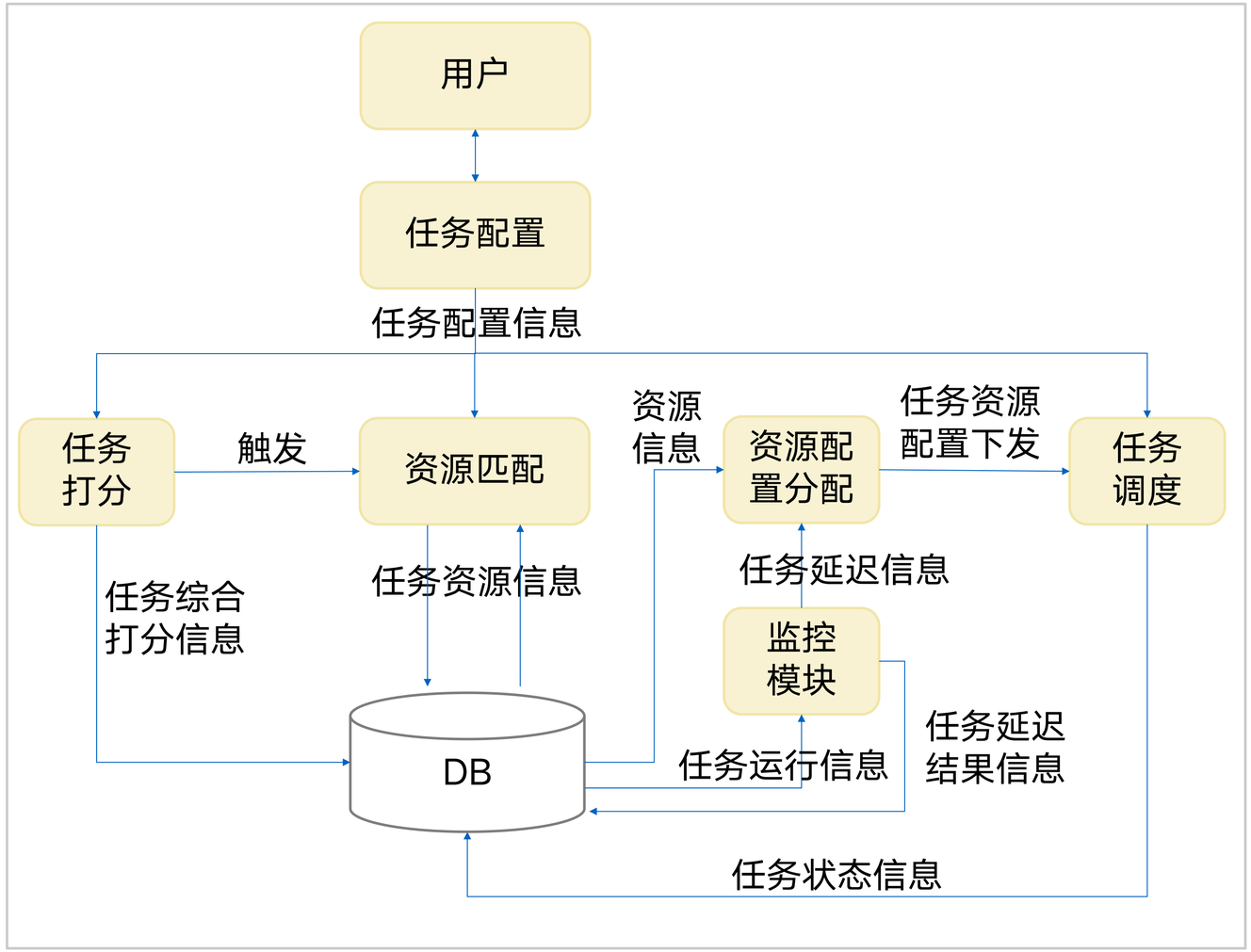
- 毫秒级内容变化事件的感知
- 毫秒级扩容
- 秒级日志接入

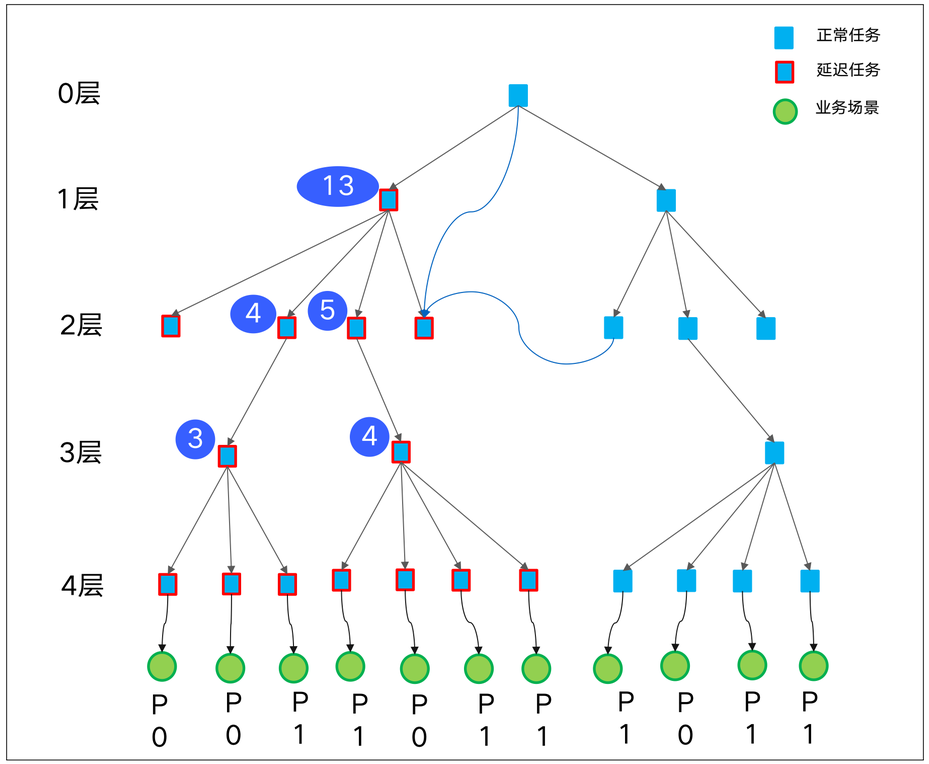


链路数据质量-数据时效性：任务延迟积压场景处理思路

vivo

分析任务视图依赖，识别关键基础任务，触发资源分配调度，及时疏通链路

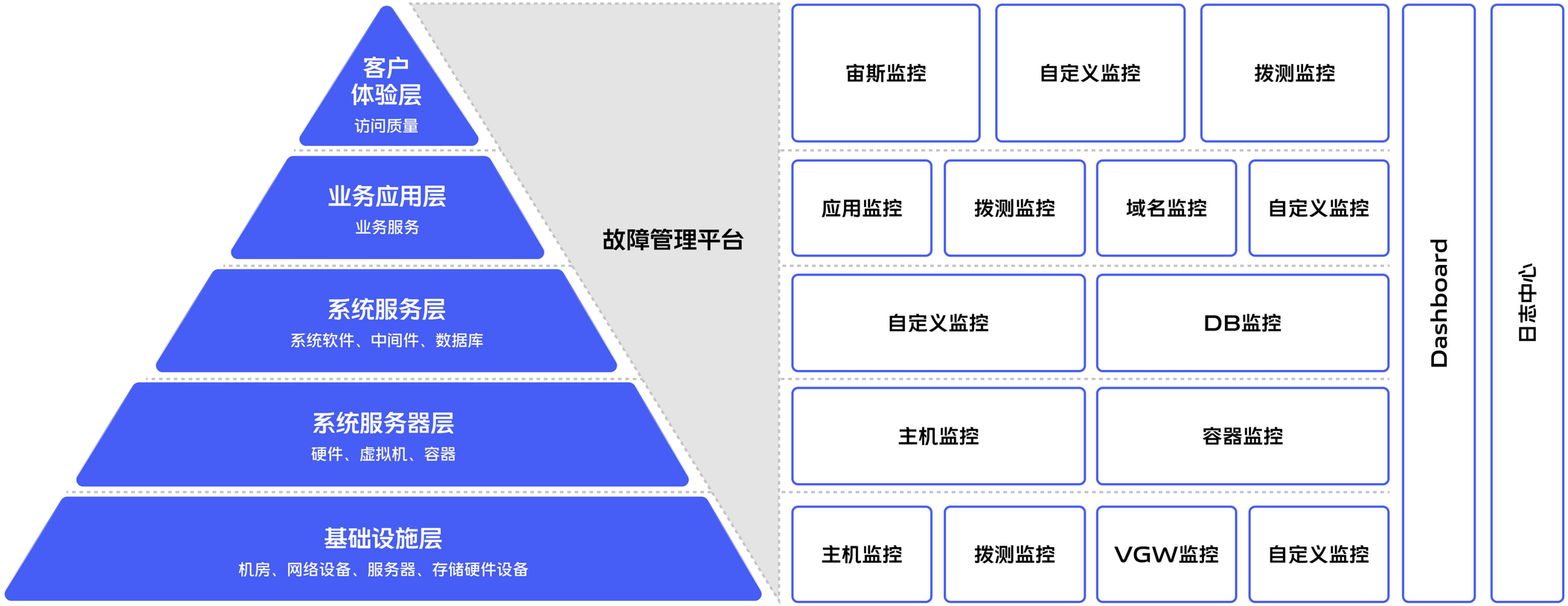




id	task（任务）	task_dependence_score (任务依赖数得分)	scene_importance_score (场景重要性)	total_score（综合得分）	resource (资源)
0	Task1	13	（2*PO+P1）+(PO+3*P1)=3*P0+4*P1	A	a
1	Task5	4	2*PO+P1	B	b
2	Task6	5	PO+3*P1	C	c
3	Task11	3	2*PO+P1	D	d
4	Task12	4	PO+3*P1	E	e

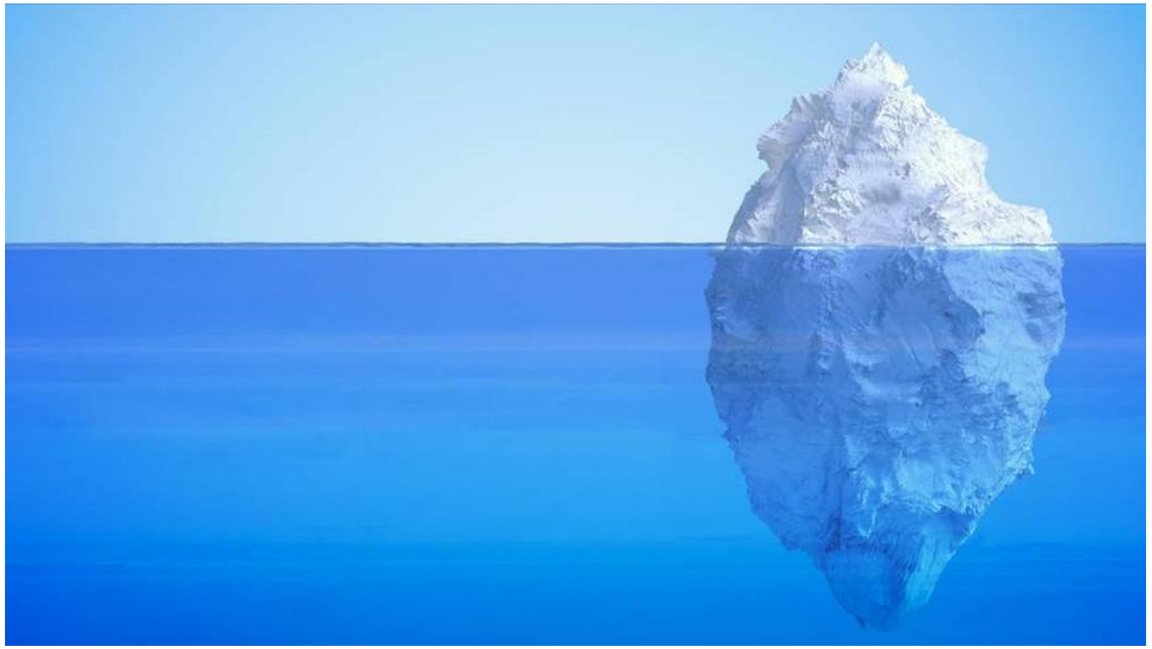
vivo数据集成可观测实践

监控能力矩阵


客户体验层
访问质量
业务应用层
业务服务
系统服务层
系统软件、中间件、数据库
系统服务器层
硬件、虚拟机、容器
基础设施层
机房、网络设备、服务器、存储硬件设备
故障管理平台
宙斯监控
自定义监控
拨测监控
应用监控
拨测监控
域名监控
自定义监控
自定义监控
DB监控
主机监控
容器监控
主机监控
拨测监控
VGW监控
自定义监控
Dashboard
日志中心

从监控到可观测性

”监控是可观测的一种实现手段，但可观测远不止于监控。“



监控

通过收集、分析和使用信息来观察一段时间内的运行进度，并且进行相应的决策管理的过程，监控侧重于观察特定指标。

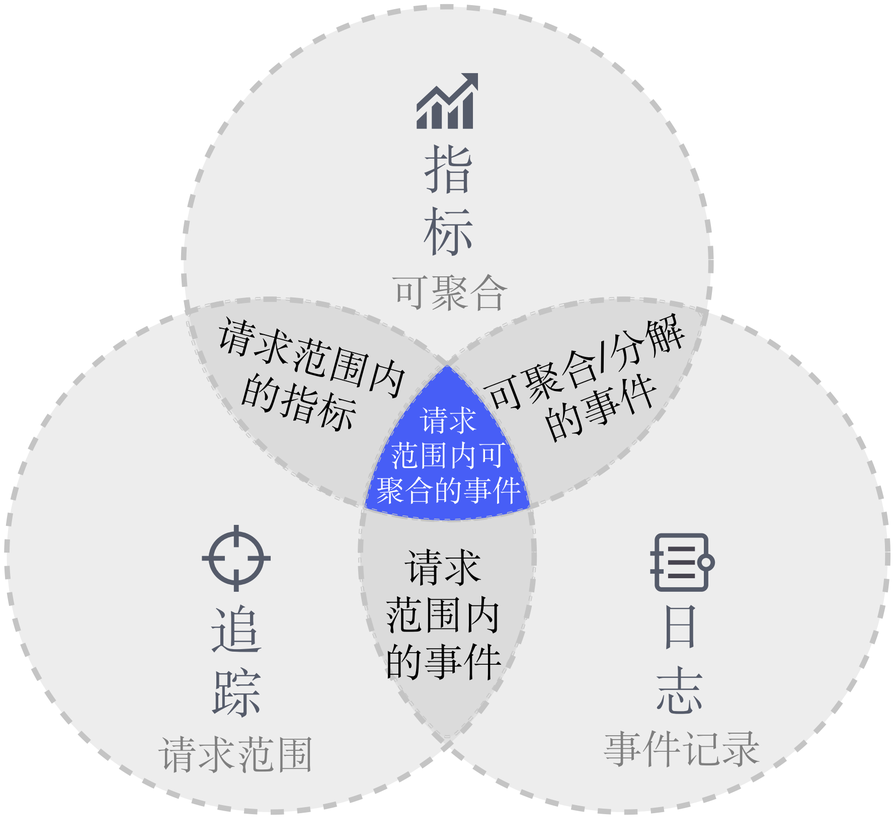
可观测

通过分析系统生成的数据理解推演出系统内部的状态。

可观测体系：统一可观测平台

通过将 指标、日志、追踪 三大支柱信息，按场景进行关联、转化、组合，实现可观测能力

统一可观测平台



一元场景

指标

可聚合的
逻辑计量单元

日志

对离散的不连续的
事件的一种记录

追踪

单次请求范围内的所有
信息、即调用链信息

转化场景

日志 → 指标

通过日志获得指标数据

日志 → 追踪

通过对日志的聚合和转化得到追踪

追踪 → 指标

通过调用链的分析获得
调用范围内的指标

日志 指标 追踪 → 故障

多个源头
产生的故障

二元场景

日志 + 指标

可聚合/分解的事件

日志 + 追踪

一个调用周期内的事件

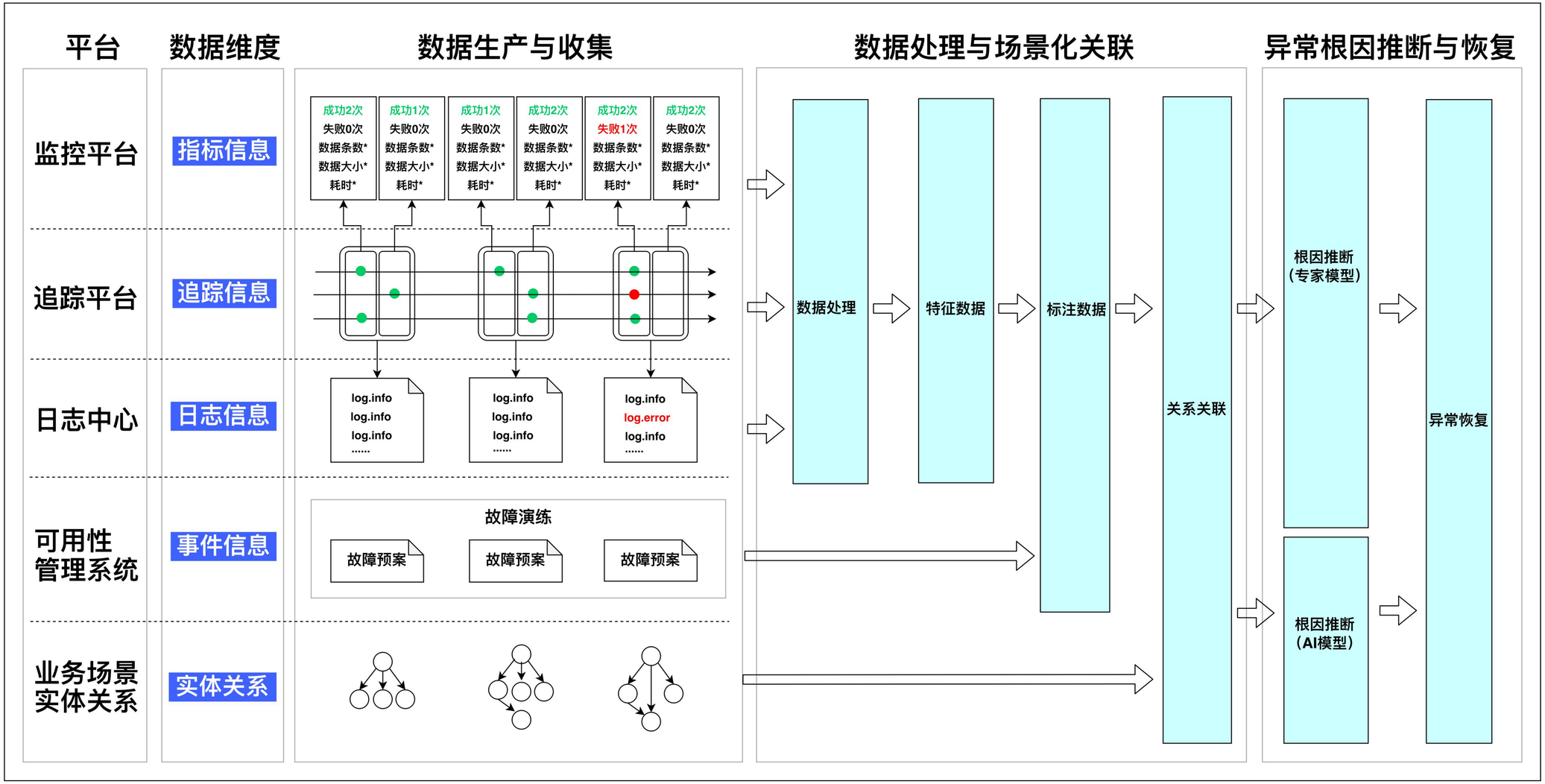
追踪 + 指标

一个调用周期内的指标

可观测体系：落地实践核心思路

vivo

可观测性的价值，不仅仅是观测，更要能助力将服务从异常状态快速恢复到健康状态



vivo数据集成领域可观测落地操作范式及实践举例

操作范式：梳理场景，确定核心影响因子，完善日志、追踪和指标，关联并自动化

场景梳理 → 影响因子确认 → 信息完善 → 信息关联和自动化

稳定性

离线稳定性异常

实时稳定性异常

.....

数据质量

数据完整性异常

数据时效性异常

机器宕机
硬件故障

HDFS容量满

接入agent
组件bug

Kafka异常

网络异常

未落日志/
文件被清空

异常变更

源端流量突增

源端数据
未上报

心跳指标

硬件异常日志

HDFS容量指
标

HDFS请求
异常日志

Kafka请求
异常日志

网络异常日志

文件访问
异常日志

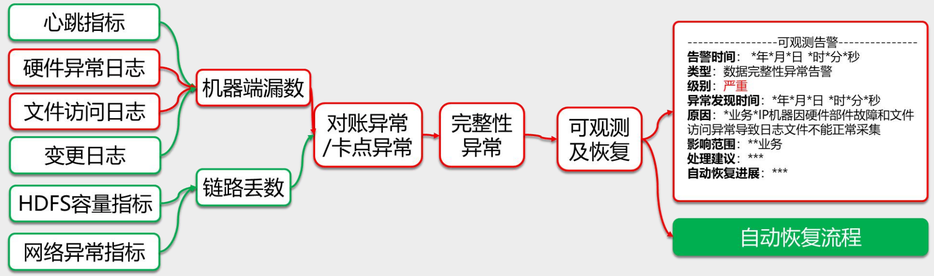
变更异常日志

流量波动指标

数据对账
异常指标

卡点异常指标

数据完整性异常场景可观测实践举例

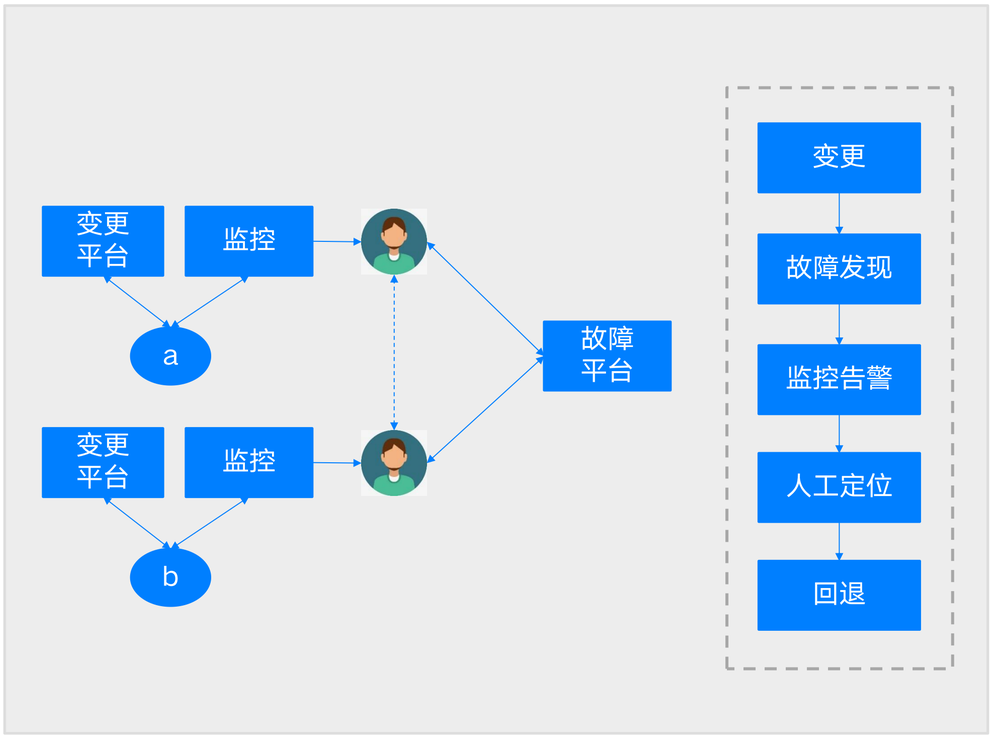


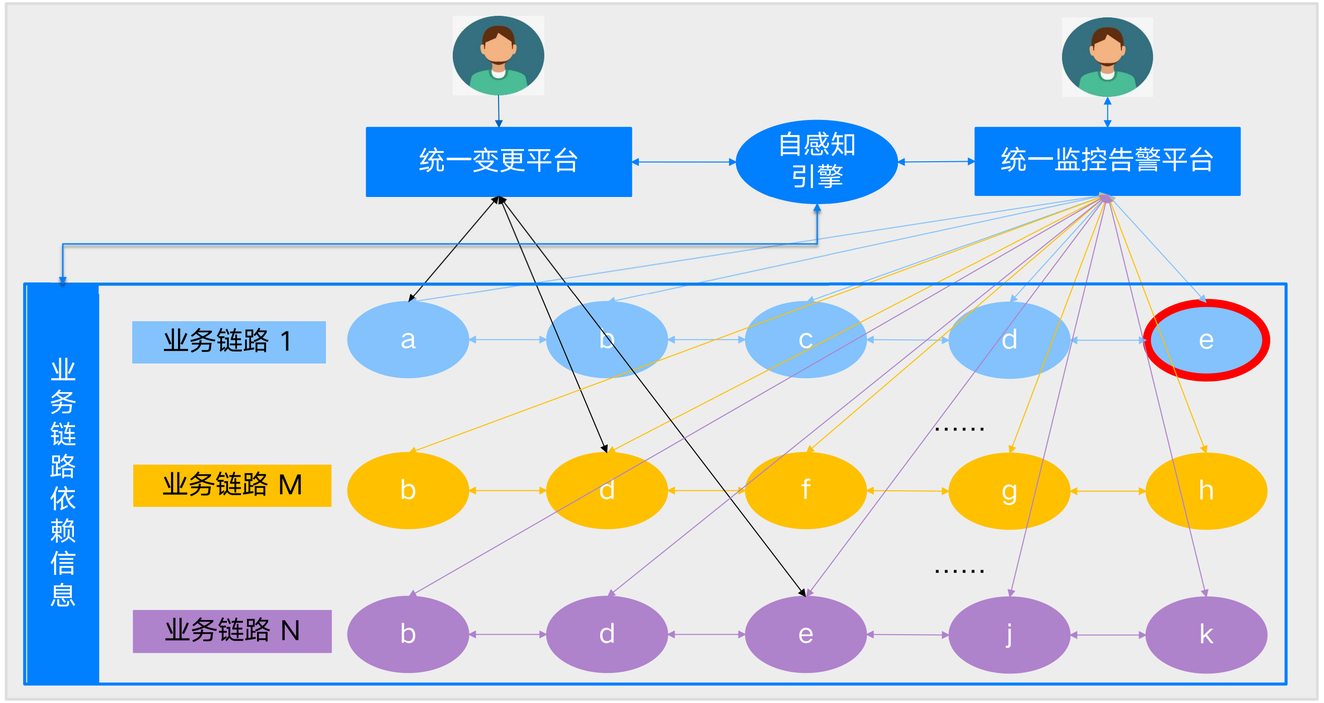
信息	标准化	信息层级	类型收敛	告警	恢复流程
心跳指标	0	L0	机器端漏数	0	-
硬件日志	1	L0	机器端漏数	0	硬件故障恢复流程
文件访问日志	1	L0	机器端漏数	0	文件访问恢复流程
变更异常日志	0	L0	机器端漏数	0	-
HDFS容量指标	0	L0	链路丢数	0	-
网络异常指标	0	L0	链路丢数	0	-
对账异常	1	L1	完整性异常	0	-
卡点异常	1	L1	完整性异常	0	-
完整性异常	1	L2	完整性异常	1	追数补数任务恢复流程

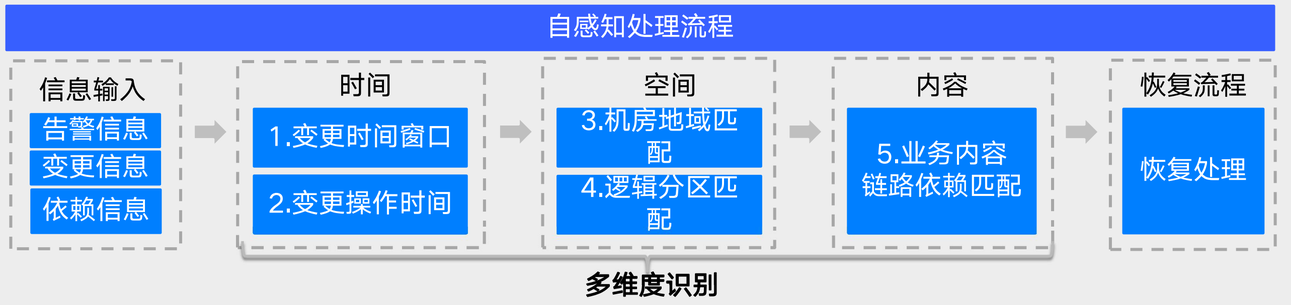
可观测实践：变更故障自感知思路

vivo

将变更信息、业务链路依赖和监控信息关联，进行多维度识别，实现变更故障自感知







未来规划

未来规划

- 提升可观测性的场景覆盖度
- 可观测平台能力建设
  - 根因分析（专家模型 -> 智能模型）
  - 增强 可观测 对接 自动恢复 的能力

想一想，我该如何把这些技术应用在工作实践中？

THANKS