# 腾讯云 Oceanus 对 Flink 云原生演进的实践与思考

董伟柯 腾讯 资深高级工程师

DataFunSummit # 2023

腾讯云 DataFun.

# 目录
# CONTENT

# 01

# 为什么选择拥抱云原生

# 大数据产品演进趋势

腾讯云 DataFun.

**分析实时化趋势**

时效性、资源分配速度、稳定性要求高

**技术多样化融合**

开源技术的百花齐放 部署方式繁杂

## 技术演进

**上云后的安全挑战**

开源组件时时爆出漏洞，需做好隔离

**低成本实现高算力**

存算分离等技术进一步降低成本

# 性能追求 永无止境

**服务混合部署，CPU 和内存超用，争抢严重**

组件相互影响，超用导致服务依次被 OOM Kill

**无法满足多样的网络和磁盘需求，性能差**

不同组件对磁盘和网络的要求不同

标准化虚拟机难以满足定制化需求

**业务存在峰谷，整体利用率低**

高峰期资源不足，低谷期存在浪费

**容器环境，CPU 和内存轻松限额**

隔离度好，按需划分资源，保障整体稳定性

**可扩展的资源定义和弹性申请能力**

通过资源定义能力，提供多种规格的磁盘和网络插件

按需申请磁盘和网卡，即时分配

**自动扩缩容，按需调配资源**

通过配置策略，可以按规则扩容和缩容

提升高峰期性能，在低谷期节省成本

# 组件多样 运维困难

**云上、本地环境不统一，上线时各种报错**

依赖不同版本的共享库，容易冲突

本地和线上环境不同，线上问题不易复现

**升级和回退流程繁琐**

每个服务的升级顺序都有差异，运维压力大

每个平台都有自己的脚本，培训成本高

**运维要求高、缺少自愈能力**

每个服务都需要配置监控、保活策略

不易发现异常，很多故障需要人工处理

**定制化镜像，标准运行环境**

每个服务有自己的镜像，互不干扰

联调、测试、线上环境统一，快速部署

**声明式的滚动升级策略**

统一的声明式 API，定义滚动升级的间隔、批次、超时策略

支持随时暂停、恢复、回滚

**高可用、故障自愈**

多种探活、自动重启机制，自动填补缺失的副本

驱逐异常节点，在可用节点重新部署

# 安全加固 不可轻视

腾讯云 DataFun.

**进程相互可见，环境共享**

裸机环境共享，如遇攻击，容易泄密

**基础隔离仍然可能逃逸**

偶现 0Day 漏洞或配置不当导致容器穿透

**容器基础隔离**

文件系统、网络、进程、CPU 内存等资源隔离、用户隔离

**底层虚拟化加固**

通过 Pod 背后的轻量级 VM 技术

保证即使容器穿透，也不会影响宿主机的其他 Pod

kubernetes

*云原生时代的操作系统*

*管理整个运行环境的计算、存储、网络等资源*

*负责资源的编排和调度，扩展性高*

# 02

# 腾讯 Flink 云原生架构演进

DataFunSummit # 2023

腾讯云 DataFun.

# 开源流计算引擎对比

腾讯云 DataFun.


**基于事件的流计算引擎**

- √ 单条事件处理
- √ 毫秒级延迟
- ! 低吞吐
- ! At-least-once
- ! 状态管理：无
- ! 易用性：SQL 支持不佳


**基于数据分片的流计算引擎**

- ! 微批处理
- ! 秒级延迟
- √ 高吞吐
- √ Exactly-once
- √ 状态管理：DataStream
- √ 易用性：支持 SQL
- √ 丰富的高级 API


**流批融合的流计算引擎**

## Flink 真正为流计算而生

- √ 单条事件处理
- √ 亚秒级延迟
- √ 高吞吐
- √ Exactly-once
- √ 状态管理：Stateful Operator
- √ 流控：天然背压
- √ 易用性：支持 SQL
- √ 支持丰富的流语义
- √ 丰富的高级 API

从分治的事件型/数据分片型流计算引擎，进入流批融合计算引擎的技术架构演进

# 从 YARN 到云原生，Flink 的增益

腾讯云 DataFun.

**资源争抢 配比不合理**

大作业相互抢占 CPU 和内存，彼此拖累

标准化硬件配置，难以满足突发需求

大集群存在资源闲置和碎片问题

**稳定性差**

作业失败、OOM 场景会导致线上业务断流

问题感知慢，经常需要人工介入

**使用门槛太高**

大数据组件繁杂，集群部署困难

高性能磁盘等定制需求很难满足

## 资源利用

- **隔离性更佳**：离在线业务互不抢占，提升总体利用率
- **扩缩容更灵活**：弹性资源池，按需申请资源
- **调度策略可调**：根据业务负载，切换调度模式

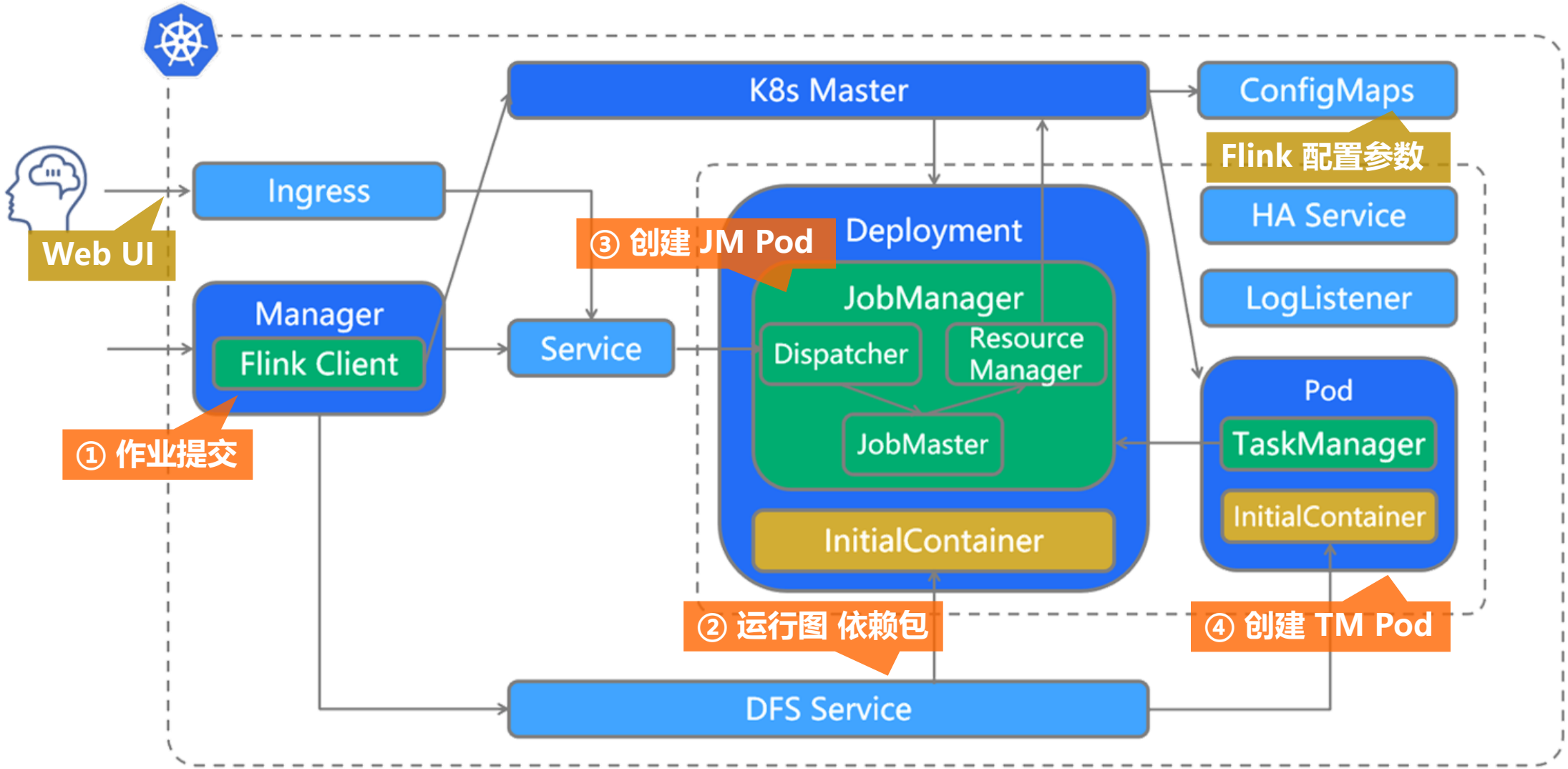
## 运维成本

- **故障自动恢复**：保证高可用，自动跨节点迁移
- **监听推送机制**：第一时间感知组件异常

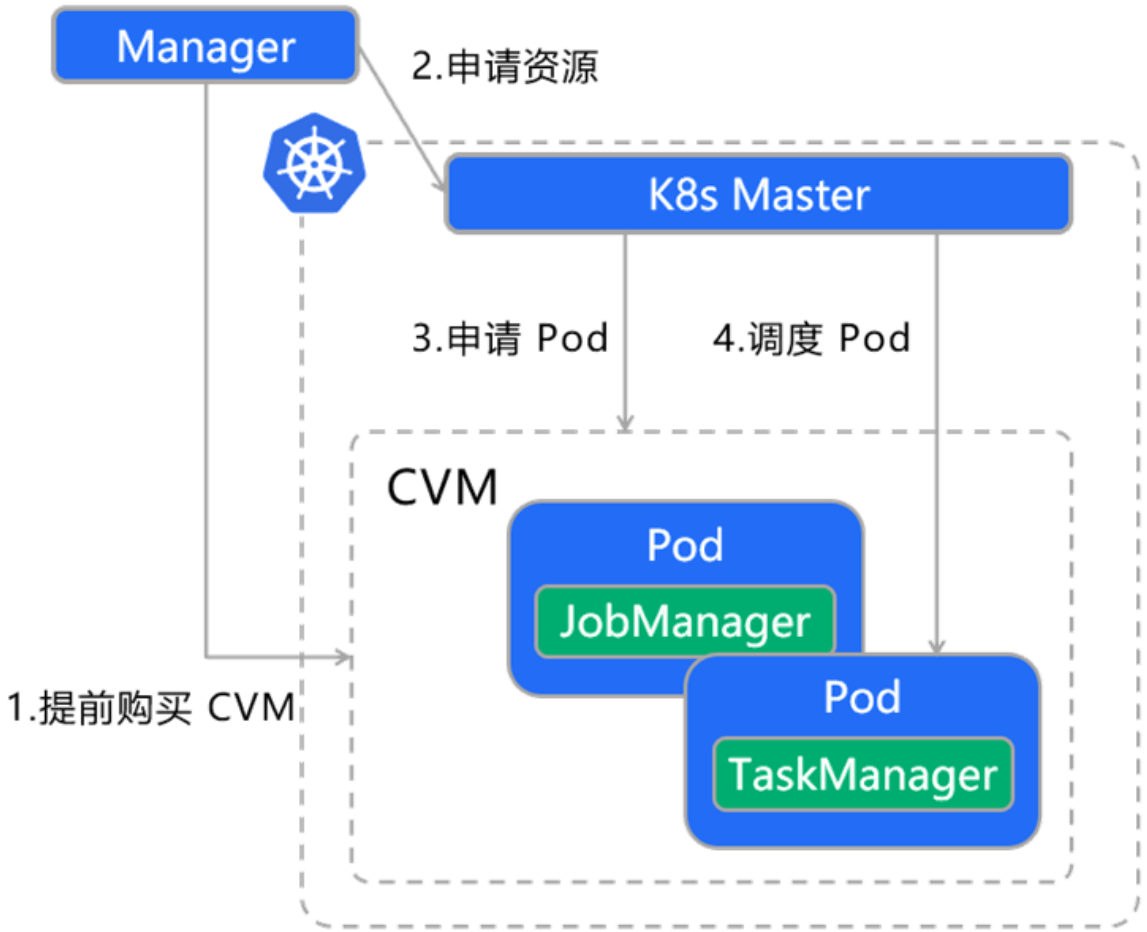
## 开发效率

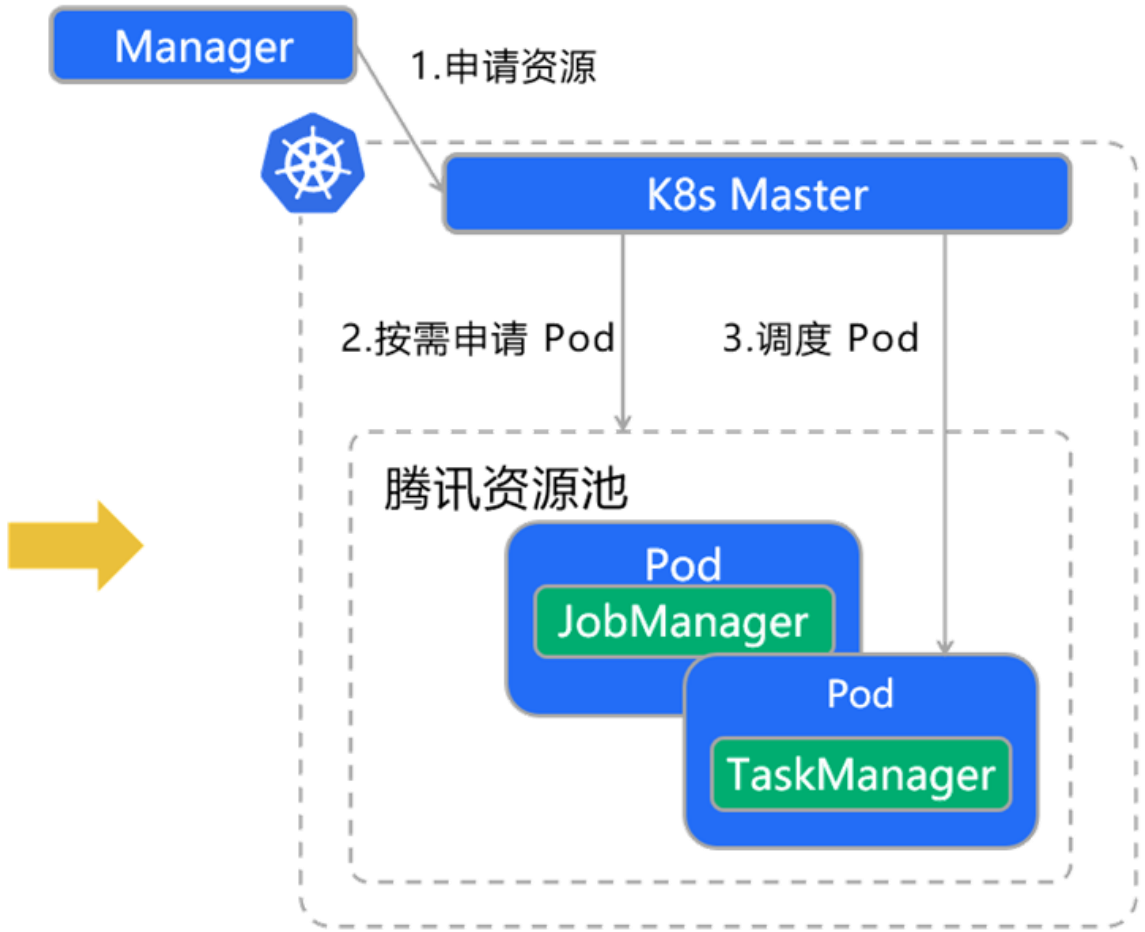
- **强大的 API**：统一的声明式 API，屏蔽细节
- **资源定义能力**：运行时、存储类、网络等资源可插拔，提供多种规格的选项

# Flink on Kubernetes 内核设计



# Serverless 弹性容器平台改造

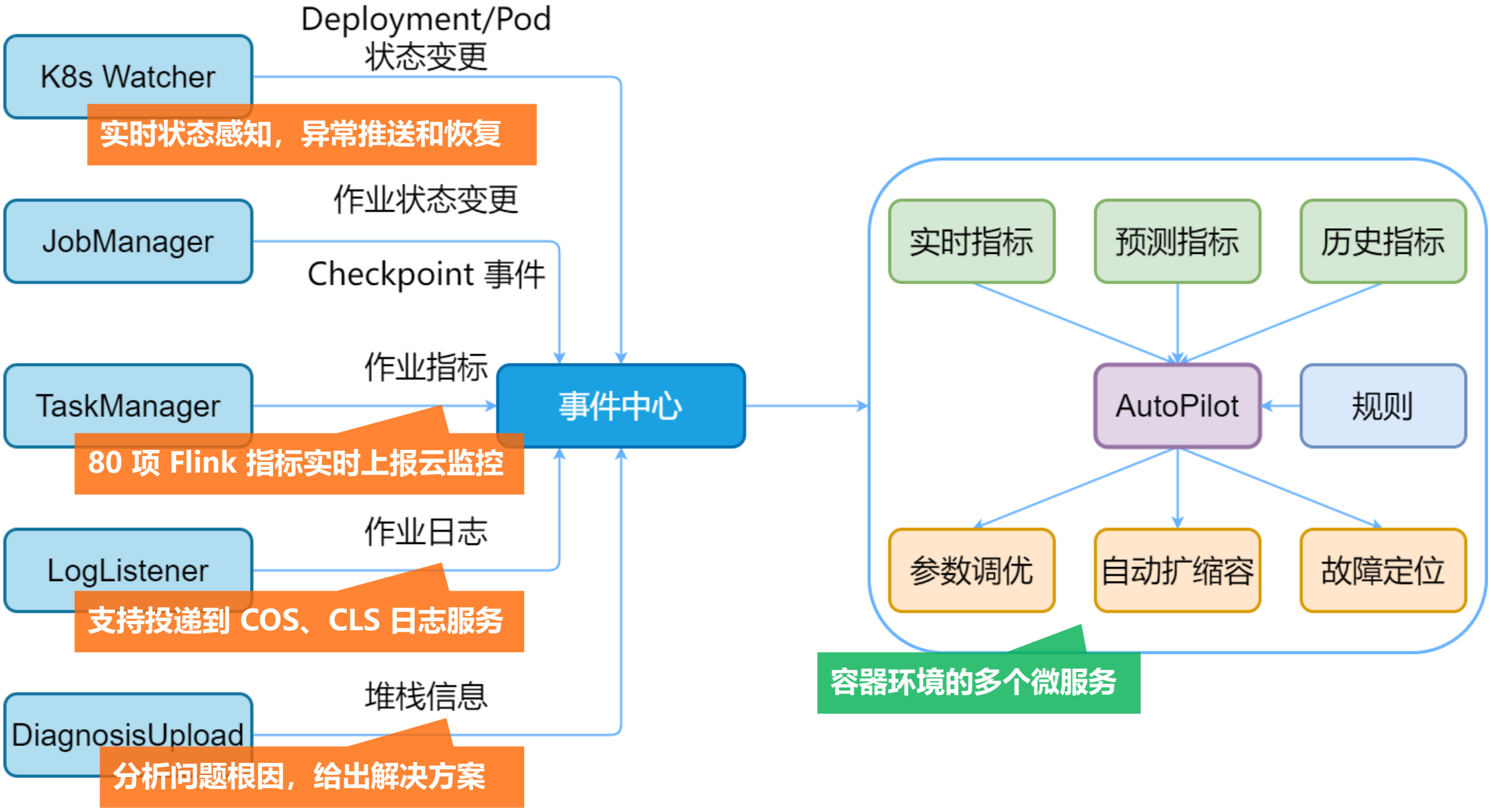


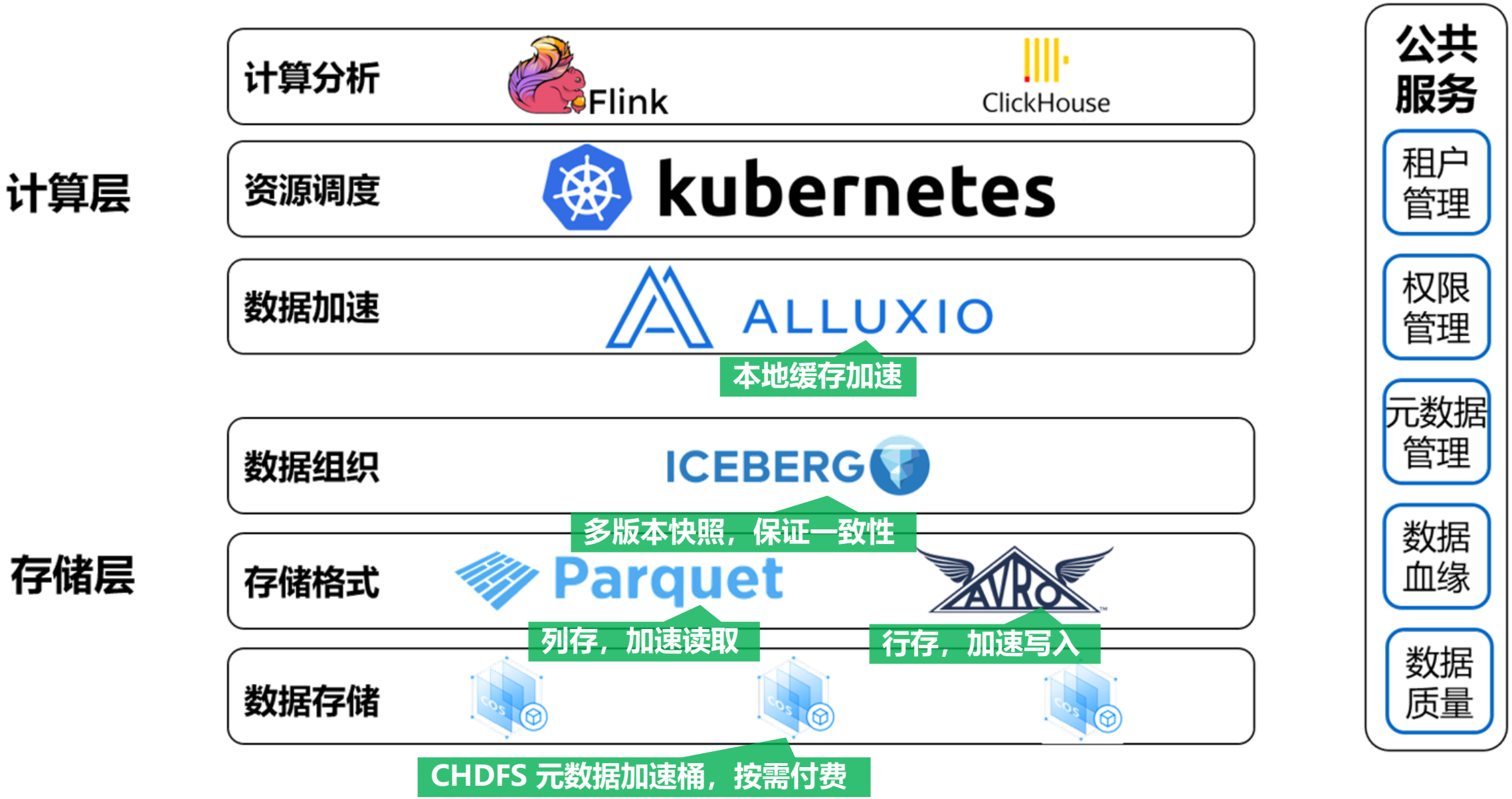

避免碎片和闲置资源造成的浪费

但需要解决调度问题、安全问题

# 云原生下的 Flink 运维体系



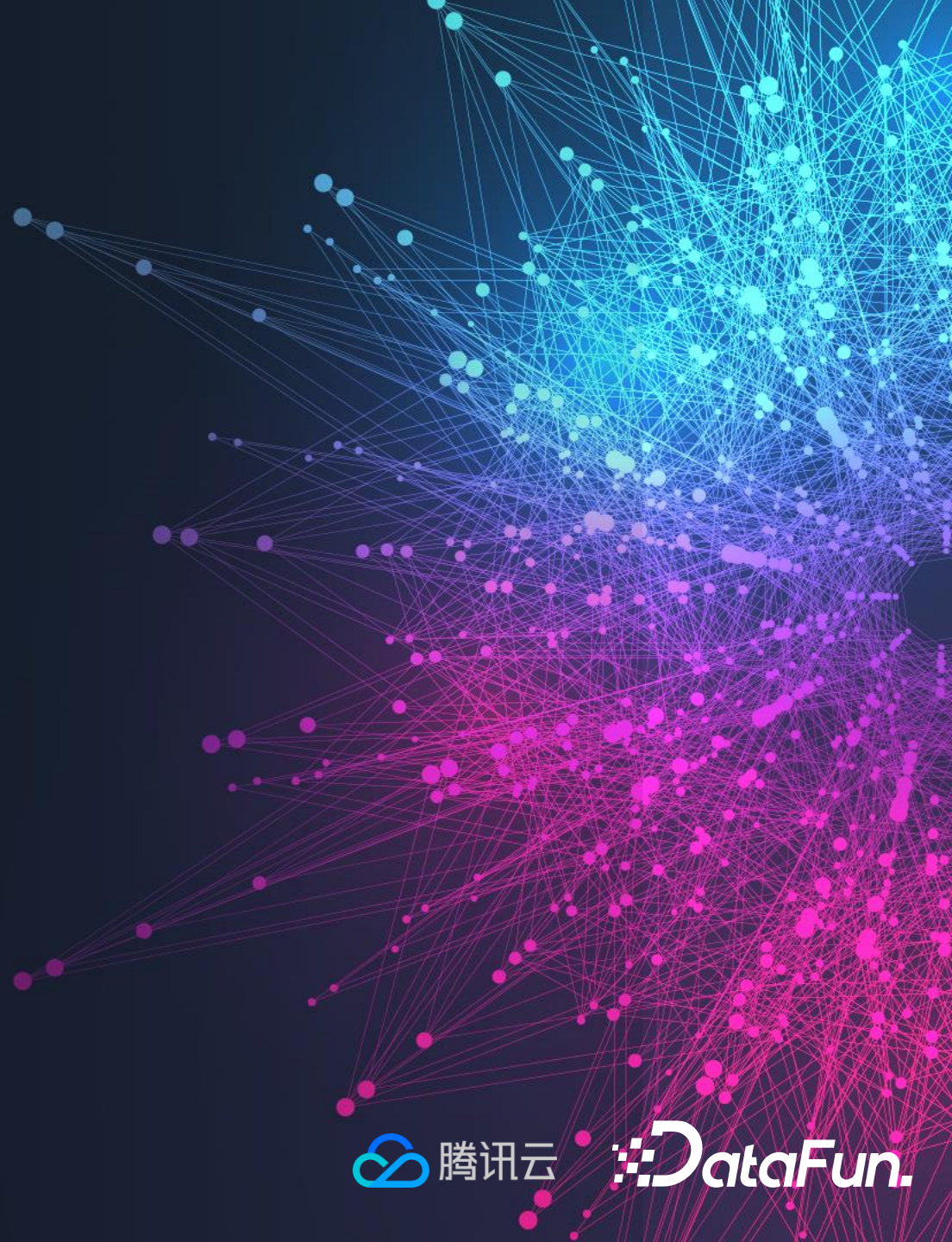
# Flink 云原生数仓架构



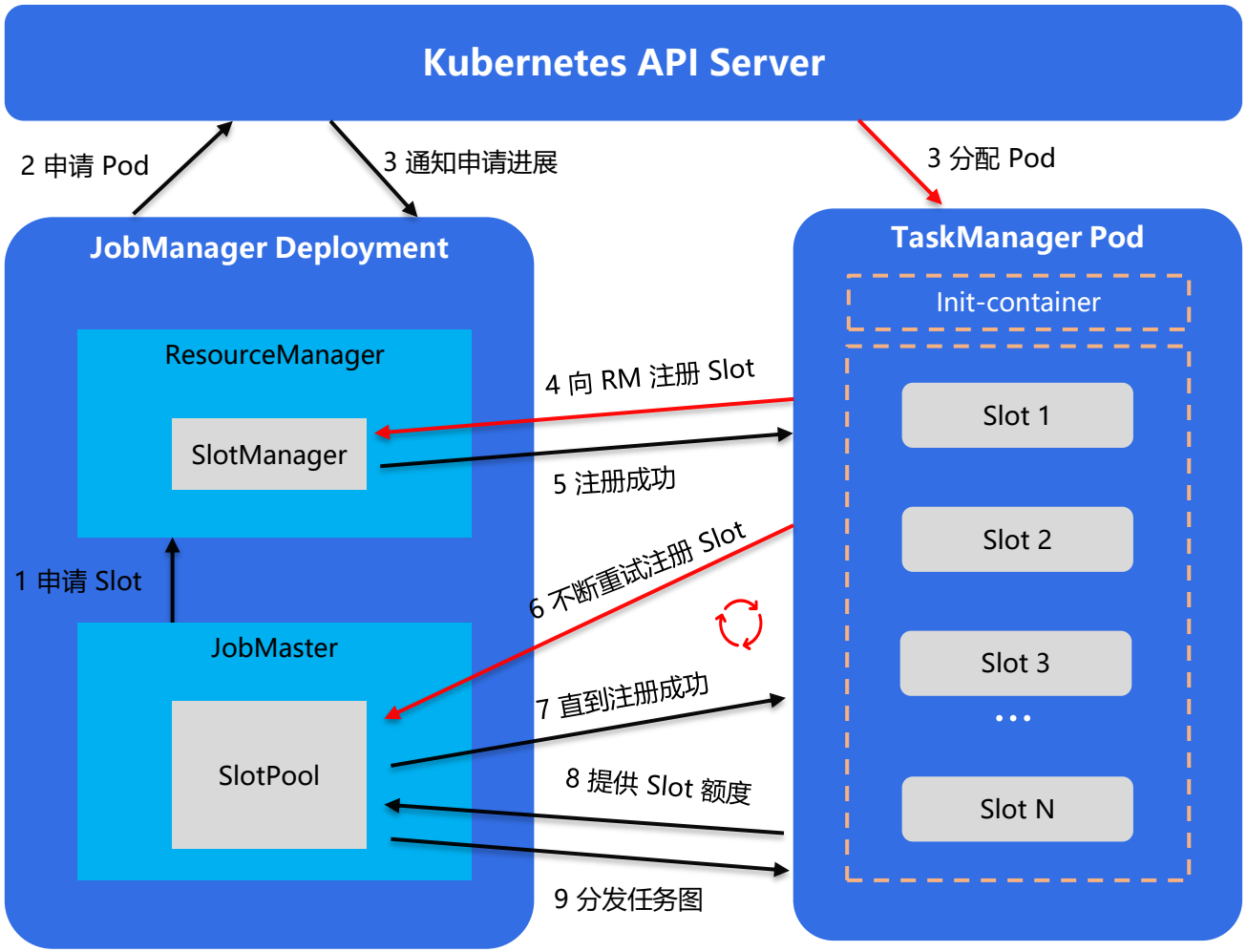
# 03

# 容器环境的深度优化

DataFunSummit # 2023



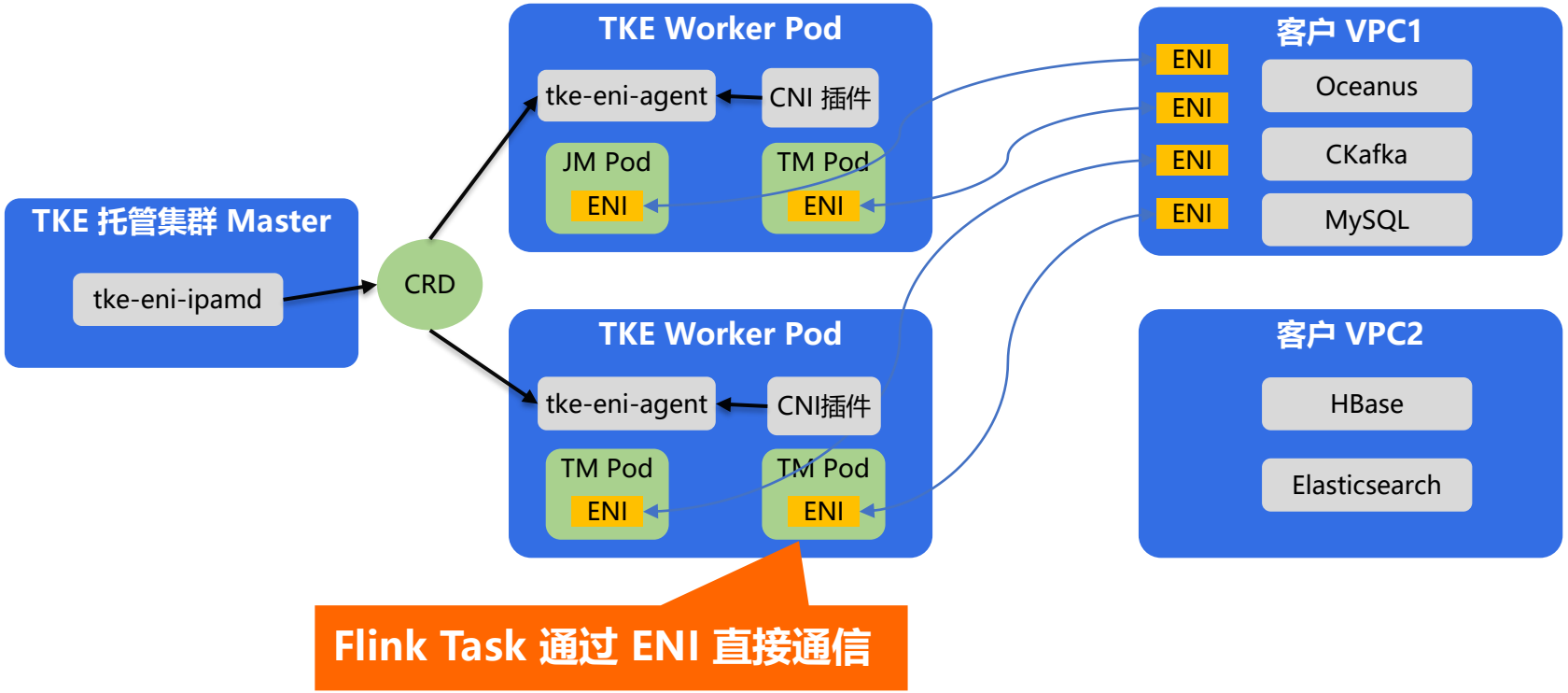
# 【性能】容器环境 Flink 启动速度优化




1. Pod 启动慢
2. Slot 注册慢
3. 资源申请慢

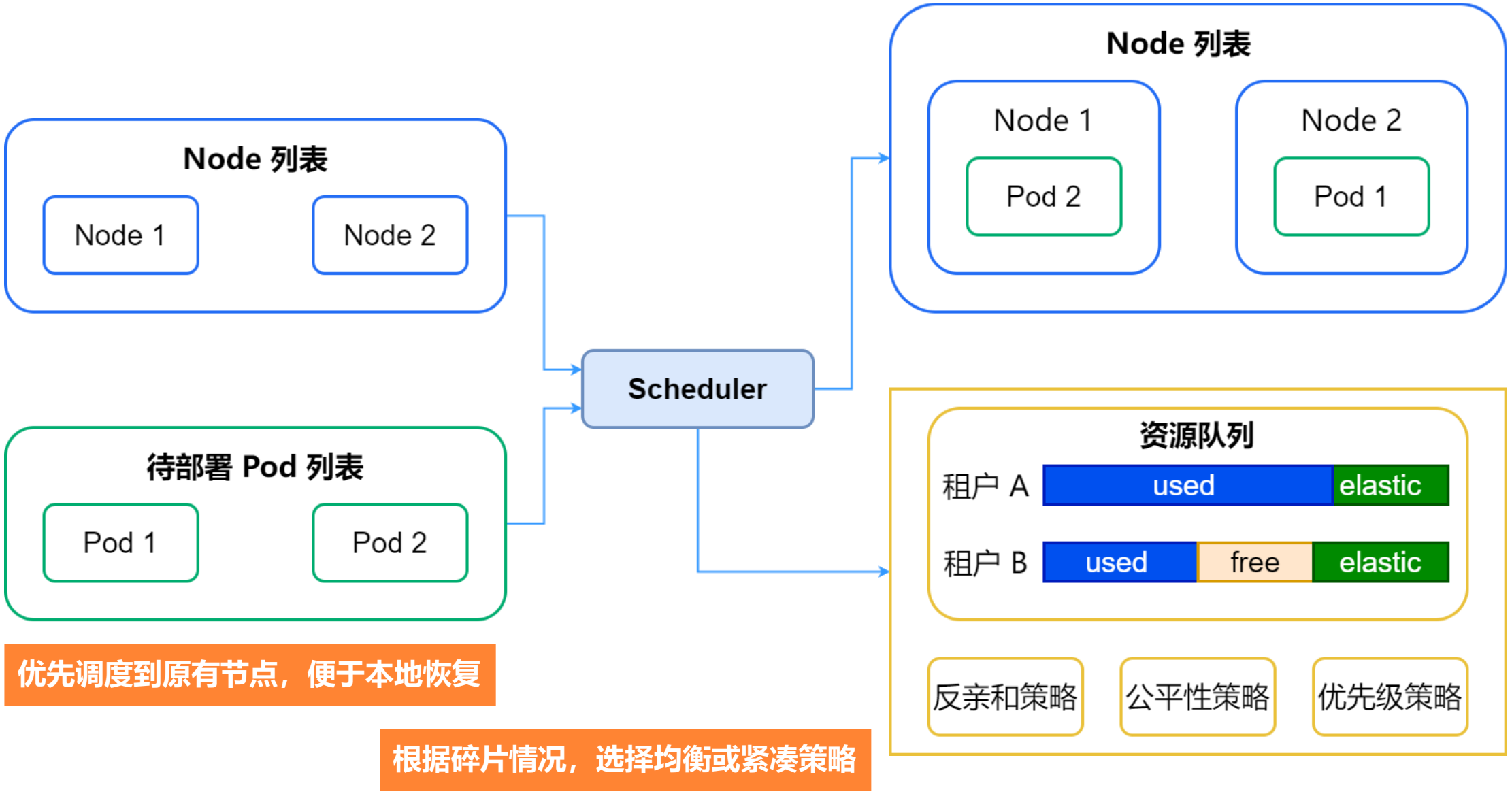
- 定制 Flink 镜像 按需裁剪
- 独享网卡 提升拉取速度
- 用户依赖与 Flink 镜像分离
- 预下载并绑定任务资源
- 精简 RPC 调用
- 弹性堆内存 先启动 后扩展

# 【性能】网络能力拓展




- **平均性能损耗降低到 1.42%，相比优化前提升 40%**
- **隔离性提升**：只有授权的 Flink 作业 Pod 能访问客户资源，同宿主机的其他 Pod 无权访问

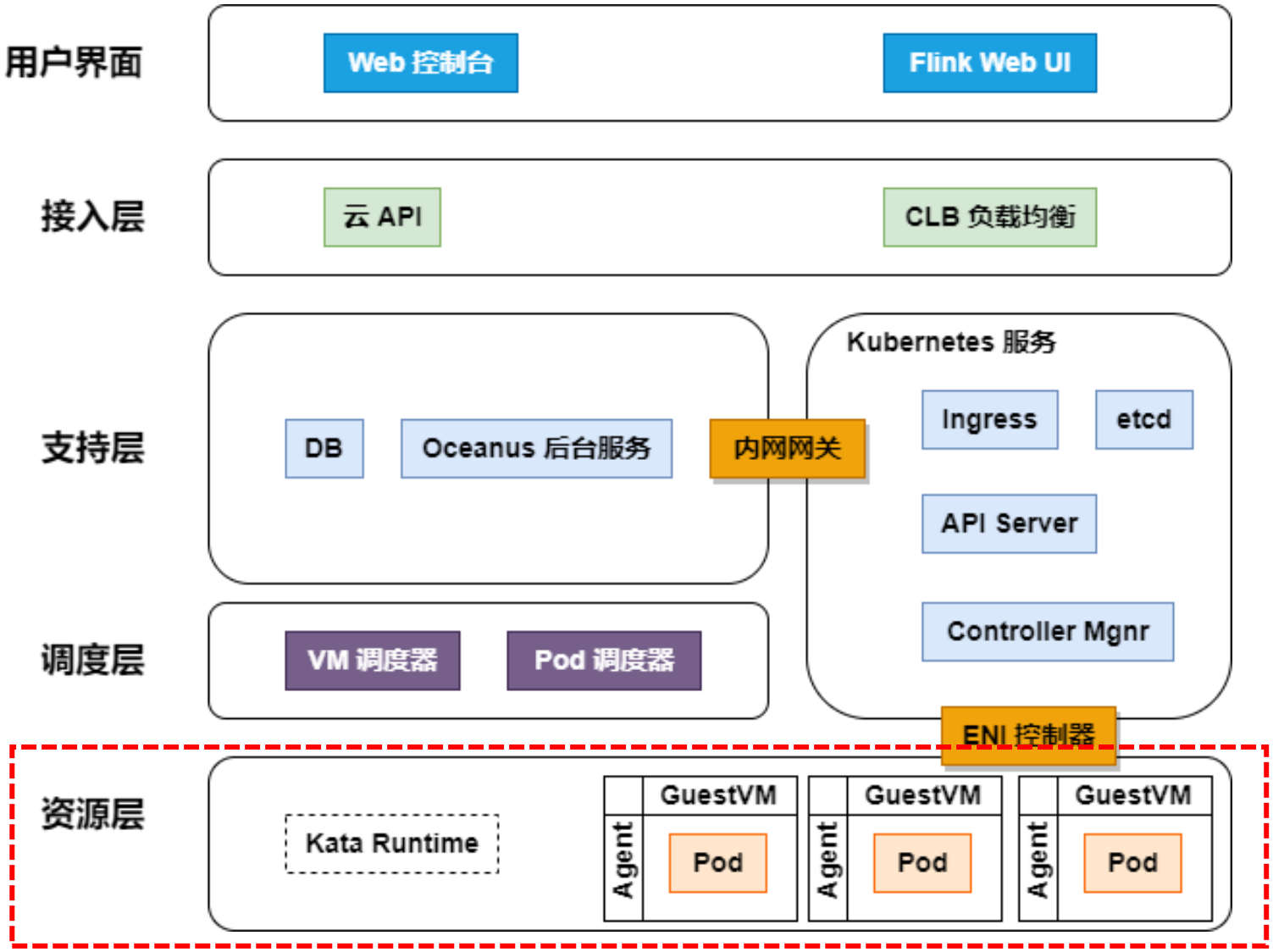
# 【性能】更适合 Flink 的调度器




优先调度到原有节点，便于本地恢复

根据碎片情况，选择均衡或紧凑策略

# 【安全】基于 MicroVM 隔离的 Pod 方案




*基于 KataContainers 的轻量级 CVM*

*通过腾讯 VStation 平台秒级启动*

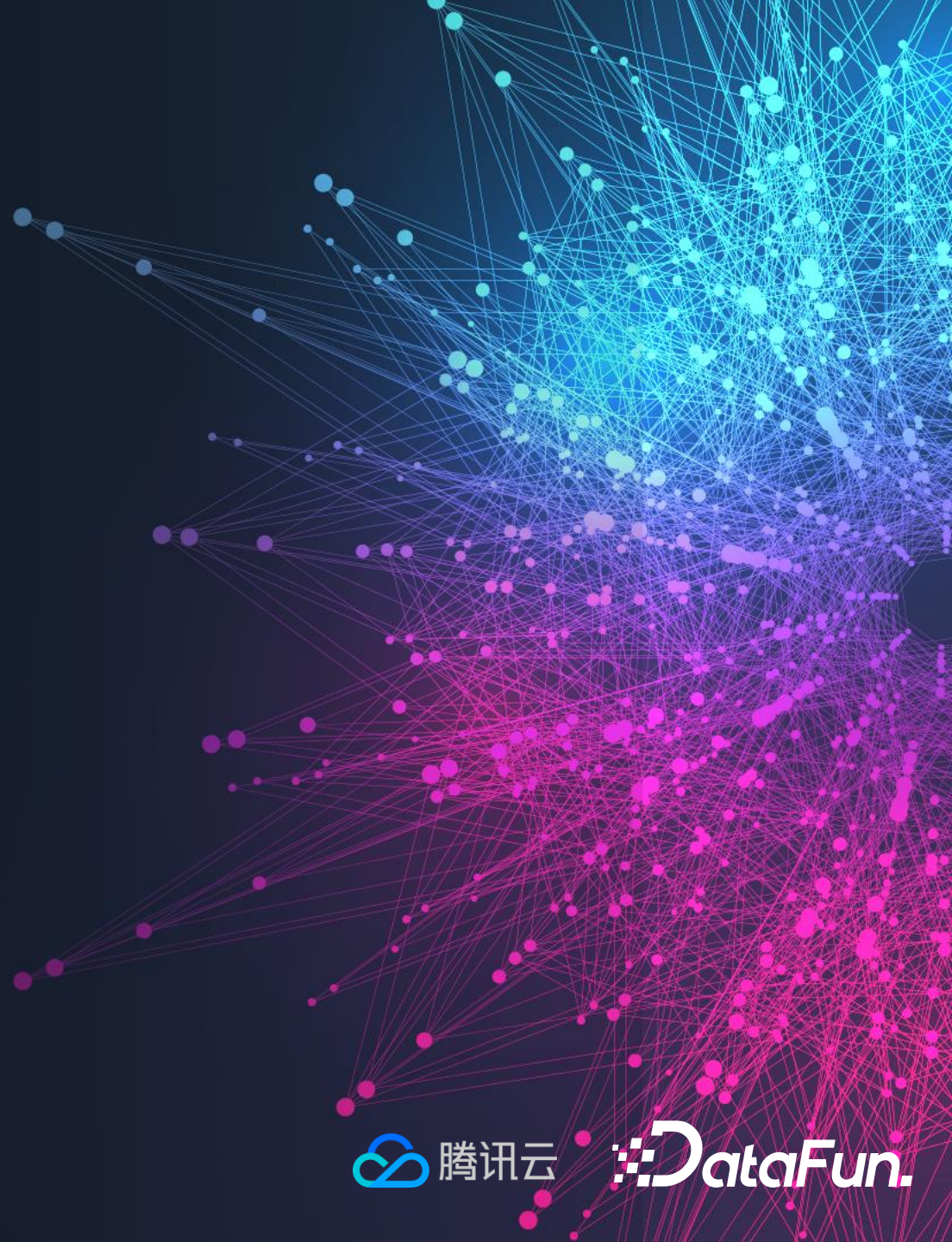
*既有传统 Kubernetes Pod 的体验*

*又有虚拟机级的安全隔离度*

# 04

# 行业实践经验分享

DataFunSummit # 2023



# 监控平台的云原生改造

## 迁移背景

- **YARN 集群资源弹性不足**

  无法按需自动扩容，大数据系统资源的高峰往往具有明显的周期性。

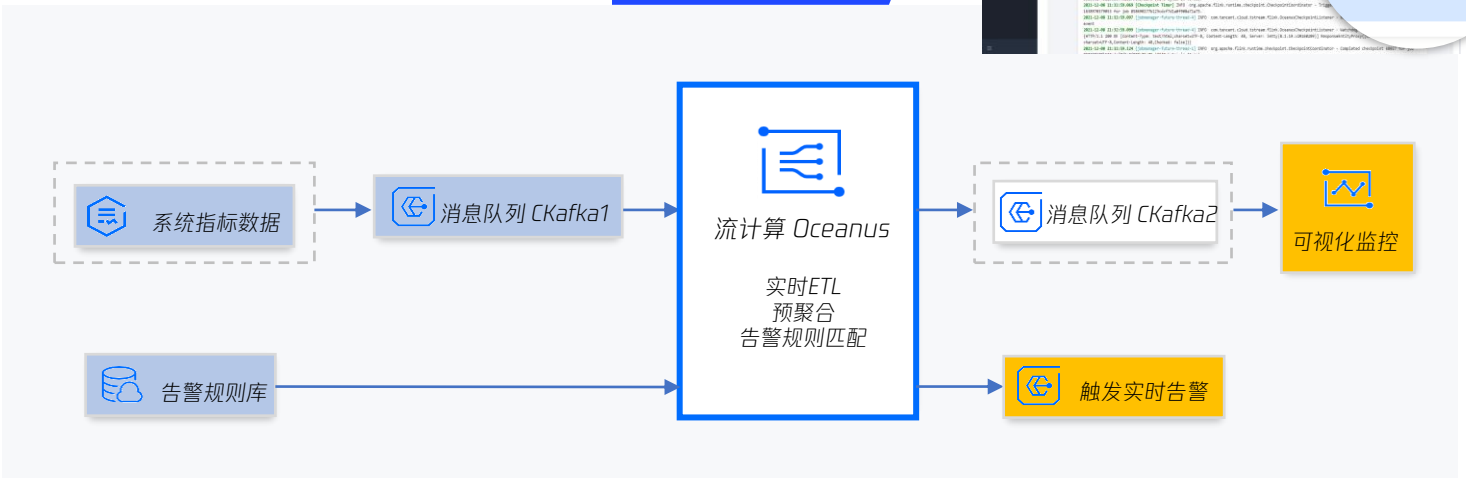
- **资源隔离性差**

  CPU、内存隔离性差，业务间相互影响，导致作业不稳定，文件系统隔离不完善，经发生类包冲突。

- **资源利用率低**

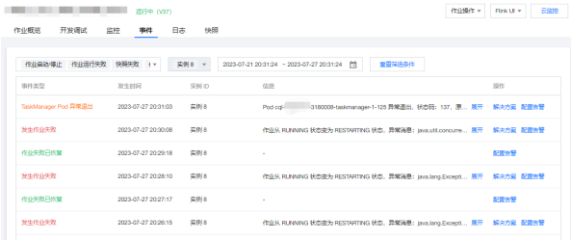
  需预留更多资源保证业务突增时资源充足，作业的 CPU 利用率普遍不高，避免业务间资源竞争影响稳定性。

## 业务架构

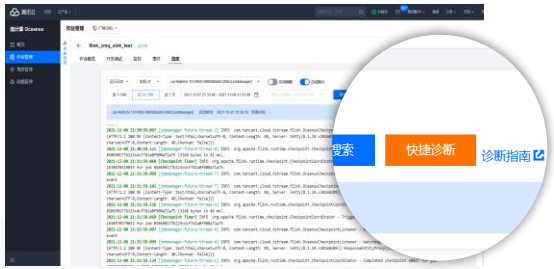



## 云上特性

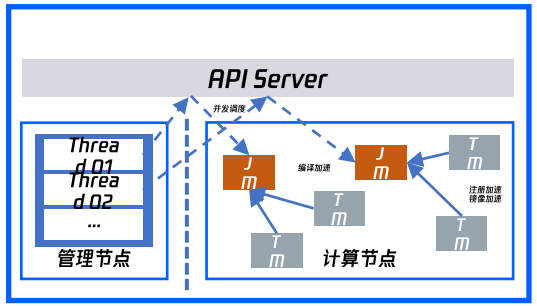
### 故障感知与作业自动扩缩容



### 日志 + 快捷诊断




### 作业高可用 + 调度容灾




### 内核以及上下游性能优化

- JM 高可用优化
- 启动加速
- Sink 端负载感知
- 存算分离

## 迁移效果

- 资源利用率从40.2%提升至58.3%
- 作业的稳定性显著提升
- 资源节省20%+

# 电商平台的云原生迁移

腾讯云 DataFun.

## 迁移背景

- **资源成本高**

  开源版本性能较低，自建集群资源利用率低，资源消耗大。

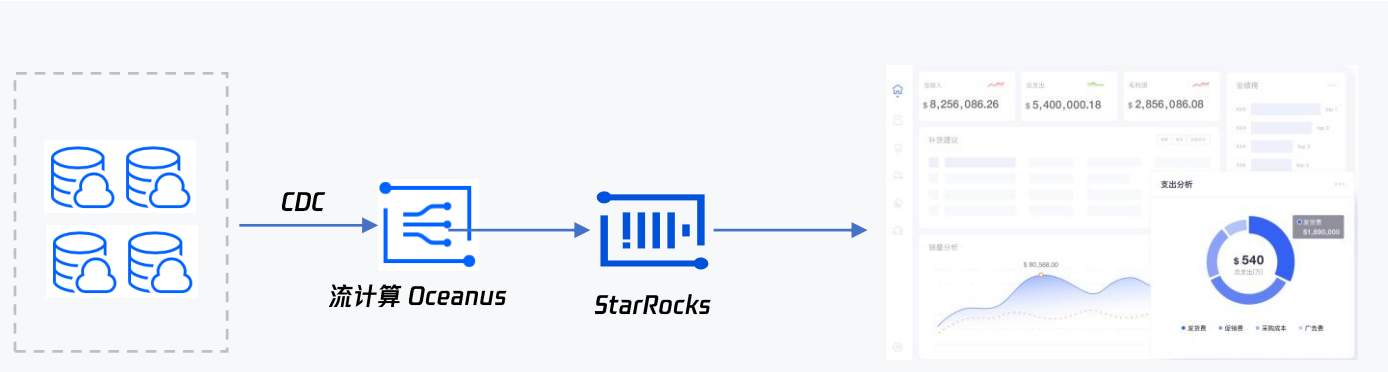
- **运维成本高**

  自建平台开发成本、运维成本高。各类问题难以第一时间发现。

- **需求多样**

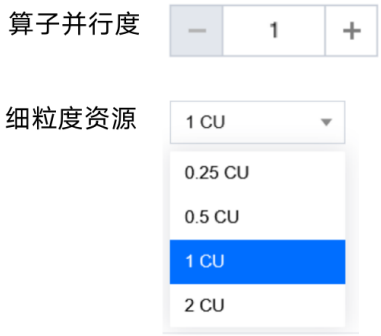
  开源版组件很多功能无法满足实际需求，需快速迭代适配。
  不同业务需要不同版本 Flink，环境多样。

## 业务架构

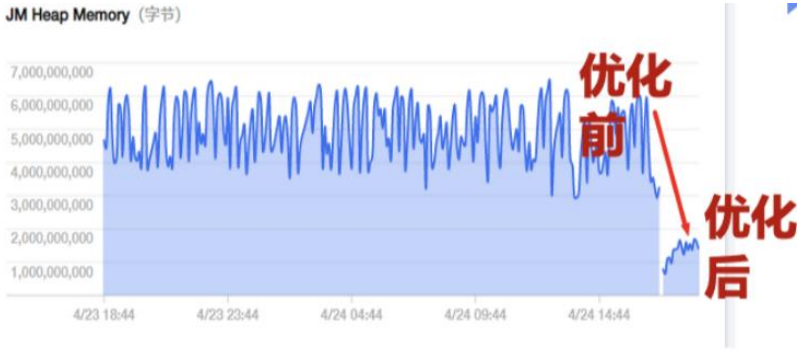



## 云上特性

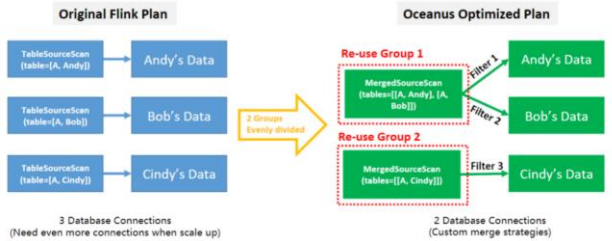
### 细粒度资源




### CDC JM 内存优化




### Source 连接复用




### 整库同步

```
CREATE DATABASE IF NOT EXISTS <target_database>
[COMMENT database_comment]
[WITH (key1=val1, key2=val2, ...)] -- 指明写入目标库的参数
AS DATABASE <source_catalog>.<source_database>     -- source_database 是被同步的源数据库
INCLUDING { ALL TABLES | TABLE 'table_name' }
-- INCLUDING ALL TABLES 表示同步数据库中的所有表
-- INCLUDING TABLE 'table' 表示同步数据库中特定的表，支持正则表达式，如 'order_.*';
-- 同步多张表时，可以写成 INCLUDING TABLE 'tableA|tableB|tableC'的格式
[EXCLUDING TABLE 'table_name']
-- EXCLUDING TABLE 'table' 表示不同步数据库中特定的表，支持正则表达式，如 'order_.*';
-- 排除多张表时，可以写成 EXCLUDING TABLE 'tableA|tableB|tableC'的格式
[/*+ OPTIONS(key1=val1, key2=val2, ... ) */]
--  (可选，指明读取source的参数，如指定source serverId的范围，解析debezium时间戳字段类型等)
```

## 迁移效果

- 资源成本节省 30%+
- 第一时间用上新特性
- 各版本无缝切换，互不影响


感谢观看
腾讯云
DataFun.