# 基于 NRI 实现精细化且可插拔的容器资源管理

曹贺 字节跳动
任强 Intel

Content

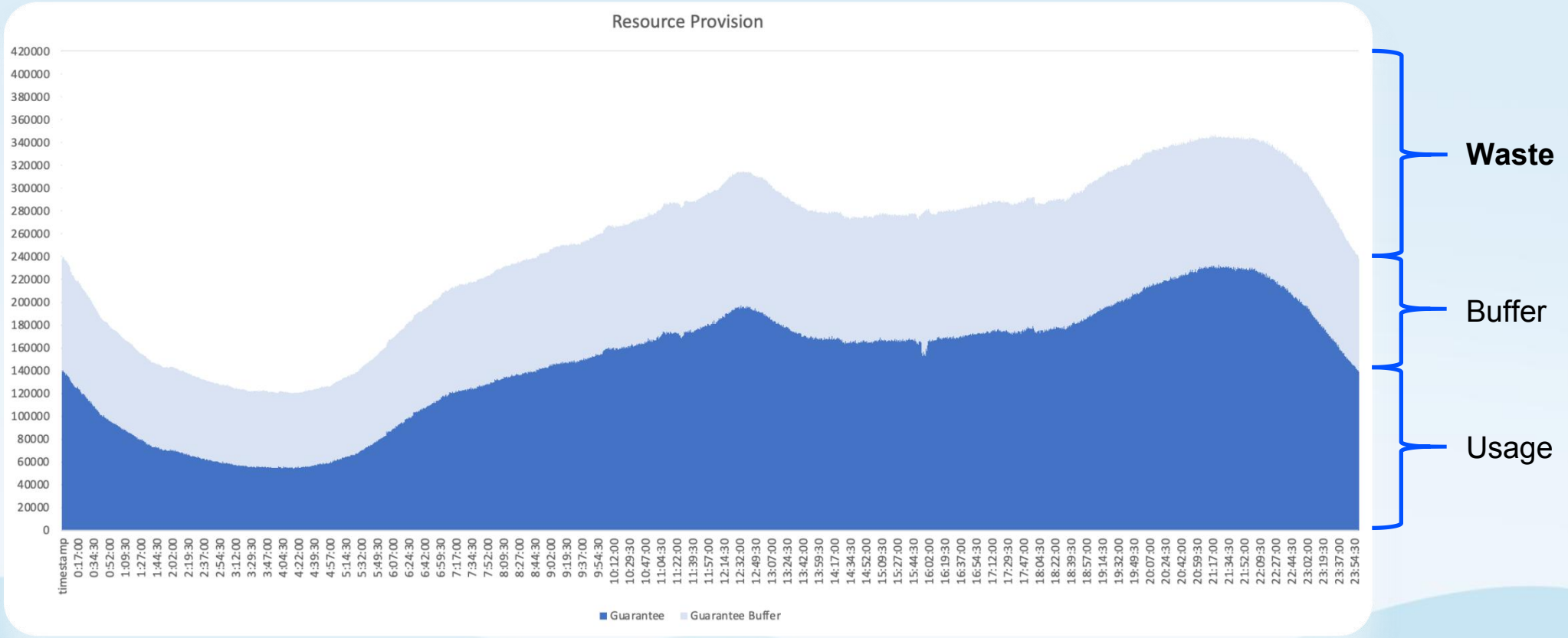
# 目录

# Part 01

# Katalyst 简介

# Kubernetes 资源规划的挑战

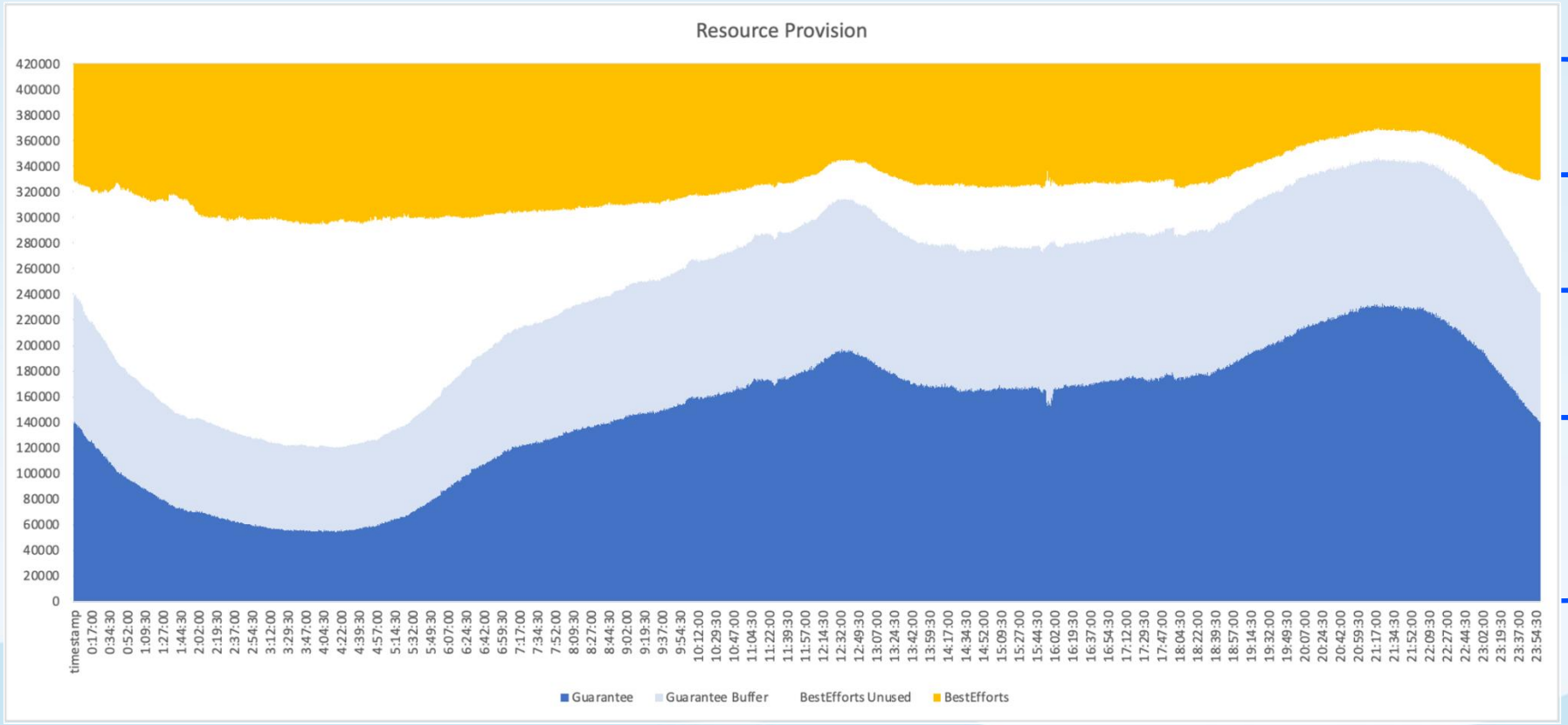
- 在线业务的资源利用率呈现潮汐现象，夜间的资源利用率非常低
- 用户倾向于过度申请资源以保证服务的稳定性，造成资源浪费



# 在离线混部

在线业务和离线作业对资源的使用模式天然是互补的:

- 在线业务重 CPU 和 RPC 时延
- 离线作业重内存和吞吐

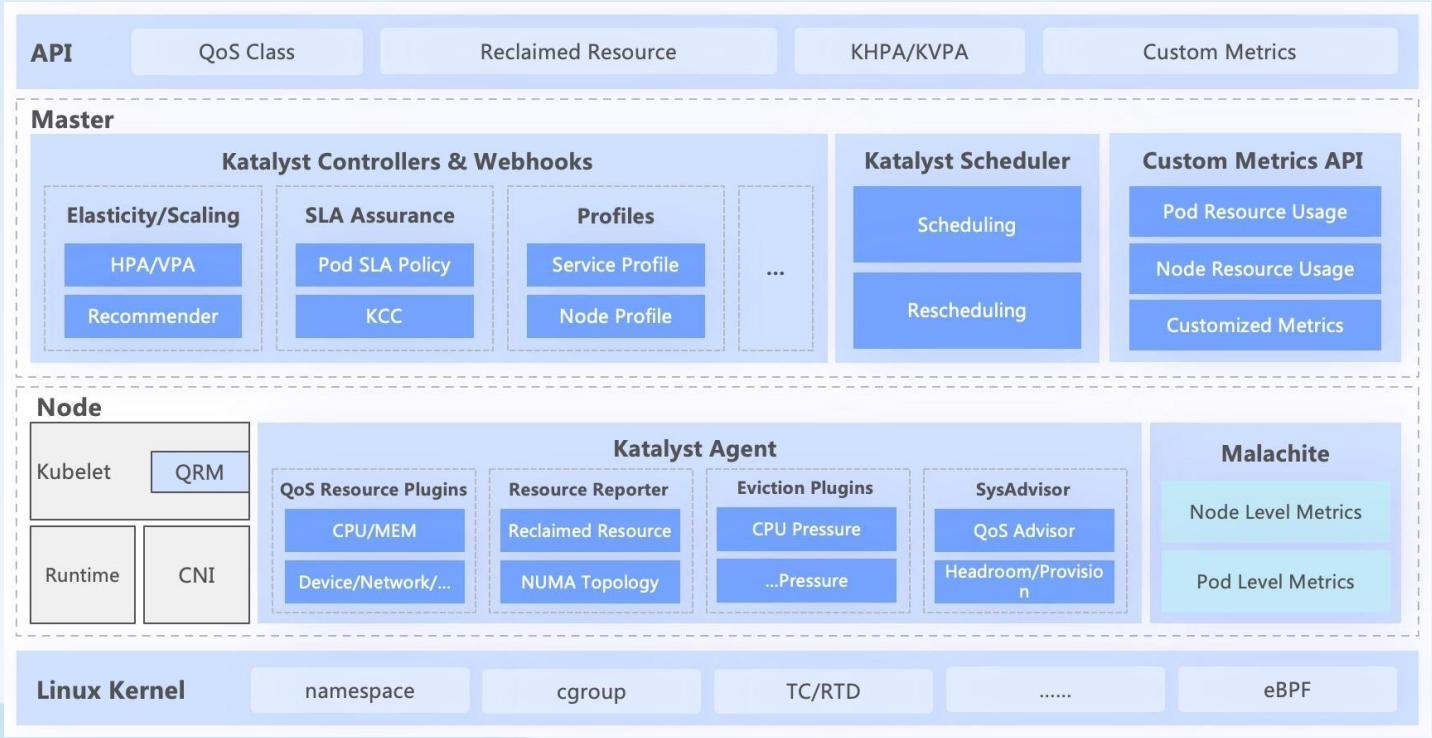


# 资源管理系统 Katalyst

Katalyst

引申自 Catalyst，本意为化学反应中的催化剂

旨在为运行在 Kubernetes 之上的工作负载提供更加强劲的资源管理能力





https://github.com/kubewharf/katalyst-core

# Part 02

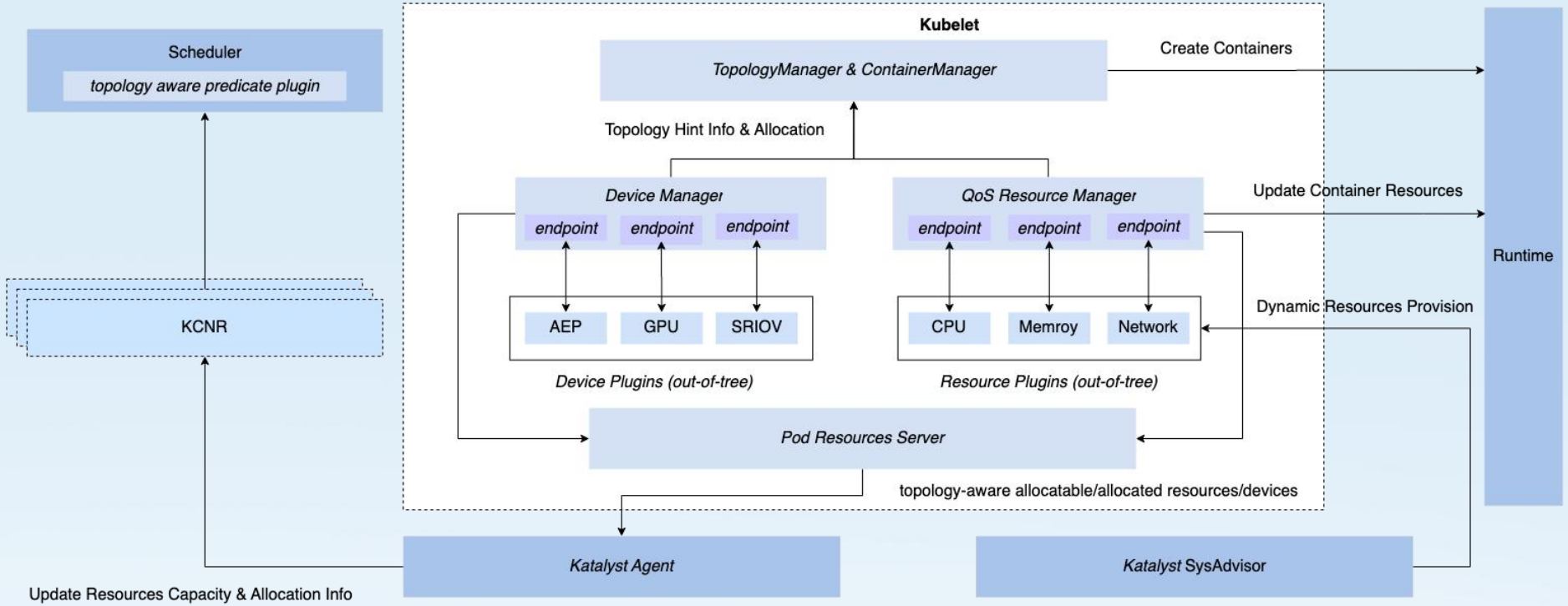
# Katalyst 插件化的资源管理机制

# 精细化的资源管理策略

- **4 种扩展的 QoS 级别**
  - 表达了服务对资源质量的要求
  - 以 CPU 为主导的维度来命名
- **更多 QoS Enhancement**
  - NUMA 绑定
  - NUMA 独占
  - 流量分级
  - ...

| QoS Class | 特性 | 适合的负载类型 | 与 K8s QoS 的映射关系 |
| --- | --- | --- | --- |
| dedicated_cores | • 独占核心，不与其他负载共享<br>• 支持 NUMA 绑定，提供更极致的性能体验 | 对延迟极度敏感的业务，比如搜索、广告、推荐等 | Guaranteed |
| shared_cores | • 共享 CPU 池<br>• 支持根据业务场景进一步切分 CPU 池 | 可以容忍一定的 CPU 限流或者干扰的业务，比如微服务 | Guaranteed/ Burstable |
| reclaimed_cores | • 超售资源<br>• 资源质量相对无保障，甚至可能被驱逐 | 对延迟不敏感、更在乎吞吐的业务，比如离线训练和批处理作业 | BestEffort |
| system_cores | • 预留核心<br>• 保障系统组件的稳定性 | 关键的系统组件 | Burstable |

# QRM 框架：使 kubelet 的资源管理策略可扩展

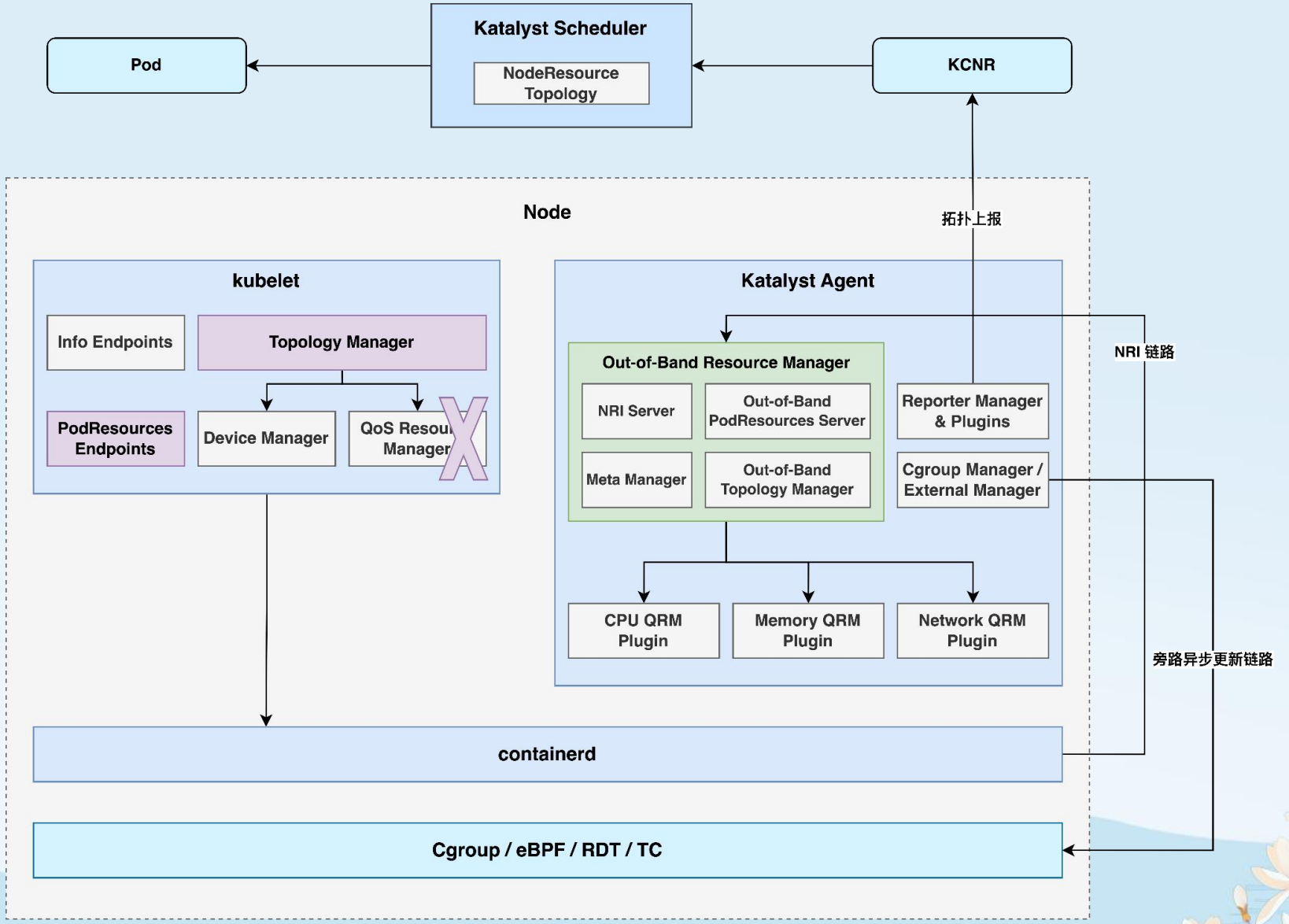



- **QoS Resource Manager**
  - 提供插件化的资源管理能力
  - 与 Topology Manager 集成以实现 NUMA 亲和
  - 在容器运行的过程中，动态调整其资源分配
- **QoS Resource Plugin**
  - 定制对容器的资源分配策略

# ORM 框架：将 QRM 框架从 kubelet 中解耦

## ORM: Out-of-Band Resource Manager

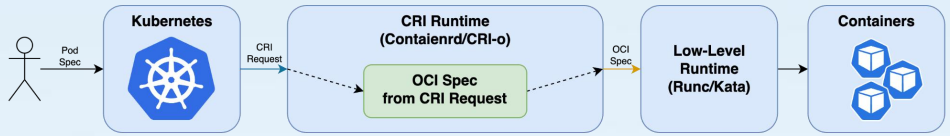
- **两种模式**
  - NRI 模式
  - 旁路异步更新模式 (Bypass 模式)
- **支持 NUMA 亲和**
  - 带外的 Topology Manager
  - 带外的 PodResources Server



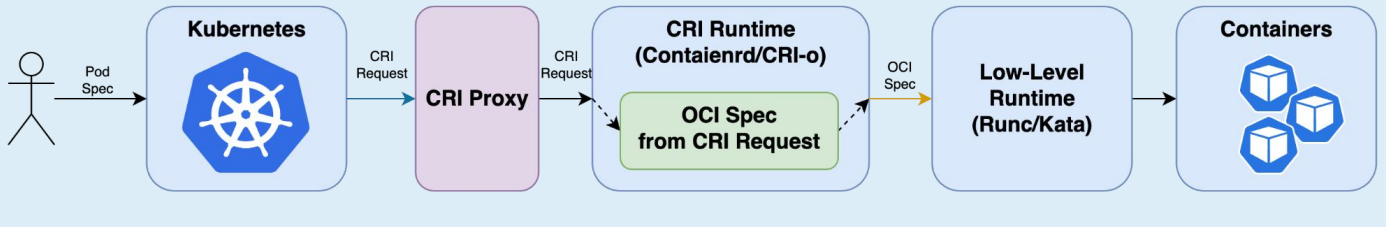
# Part 03

# NRI 机制简介

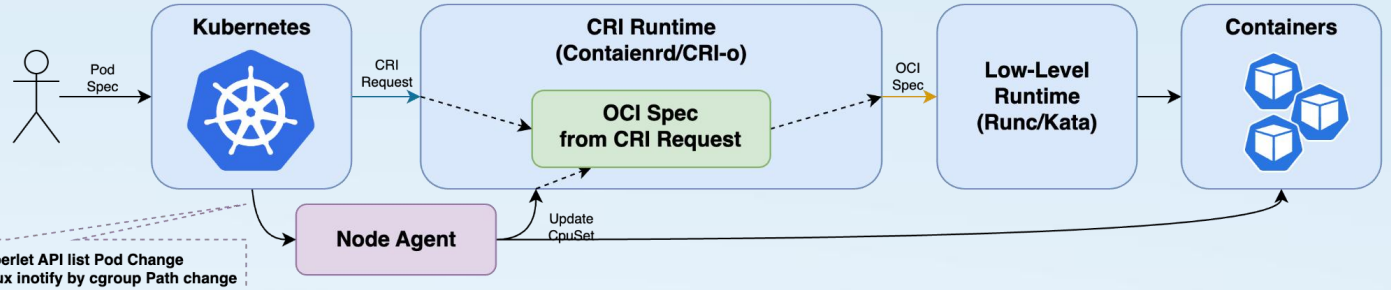
# Kubernetes QoS 资源管理目前的实现方式



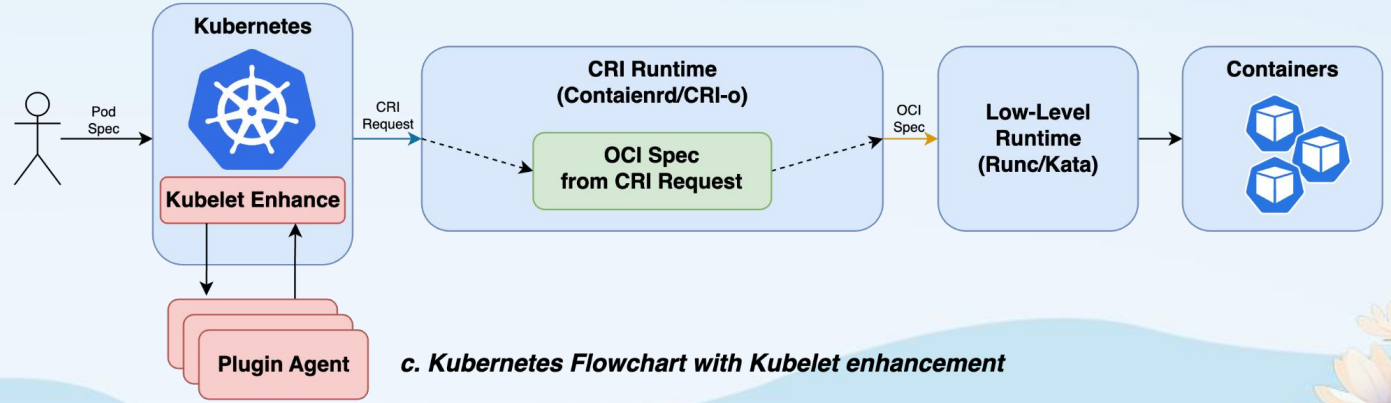

*Kubernetes Default Flowchart*




*a. Kubernetes Flowchart with CRI Proxy*



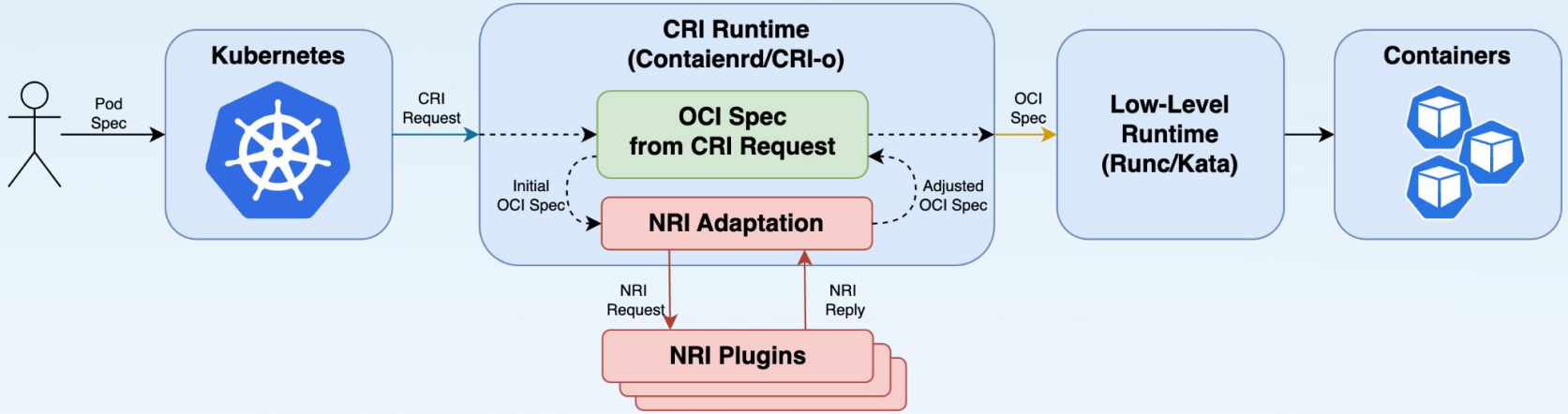

*b. Kubernetes Flowchart with Standalone Agent*




*c. Kubernetes Flowchart with Kubelet enhancement*

# NRI 节点细粒度资源管理利器

- NRI 是一个CRI运行时插件扩展管理的通用框架，可以适用于 Containerd, CRI-O 等容器运行时。
- NRI 为扩展插件提供了跟踪容器状态，并对其配置进行有限修改的基本机制。
- NRI 插件是运行时无关的，可以适用于 Containerd 也可以适用于 CRI-O。



# NRI 插件运行机制

## 运行方式

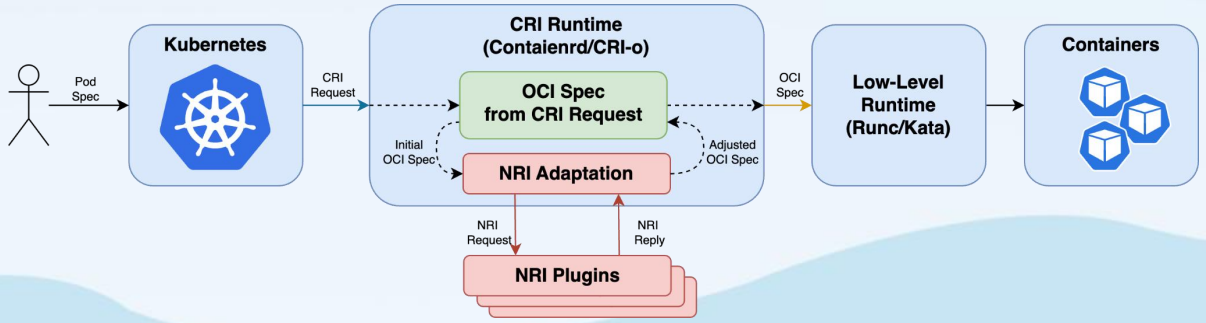
- NRI 插件作为一个独立的进程，通过 NRI Socket 与 Runtime 通信，可以部署为 Daemonset。
- NRI 插件作为预编译的二进制文件放置在指定路径下由 Containerd 发起调用 (类似 CNI)。

## Hook 事件

- Pod event： *RunPodSandbox(),StopPodSandbox(),RemovePodSandbox()*.
- Container events：
  *CreateContainer(),StartContainer(),UpdateContainer(),StopContainer(),RemoveContainer(),*
  *PostCreateContainer(),PostStartContainer(),PostUpdateContainer()*

## 主动发起

- *stub.UpdateContainer()*

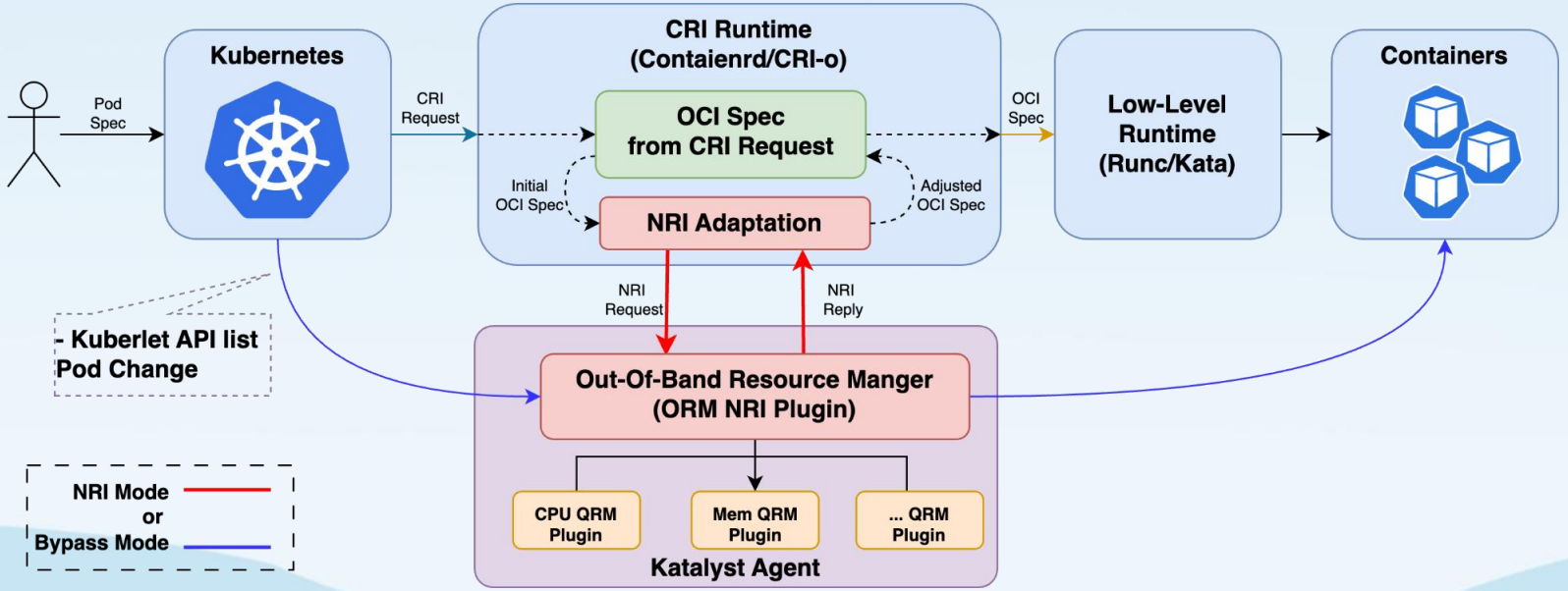


# Part 04

# NRI 在 Katalyst 中的应用

# NRI Enhanced ORM in Katalyst

- Katalyst Agent 作为一个 NRI Plugin 从 Containerd 获取 Pod/Container 事件 *RunPodSandbox(),CreateContainer(),RemovePodSandbox()*
- 旁路模式与 NRI 并存，适配低版本的 Containerd 环境
- 复用 QRM 插件机制，降低迁移成本
- Reconcile 机制通过 NRI UpdateContainer() 更新容器资源



# Demo: NRI Enhanced ORM in Katalyst


Demo 请观看演讲视频：https://b23.tv/2i0M75q

# Part 05

# 社区介绍

# 参与社区

- **GitHub 仓库**
  - Katalyst: https://github.com/kubewharf/katalyst-core
  - NRI: https://github.com/containerd/nri
  - NRI Plugins: https://github.com/containers/nri-plugins
- **Katalyst 社区双周会**
  - 周四 19:30 (北京时间)
  - 会议记录与日程


- **曹贺**
  - Email: caohe.ch@bytedance.com
  - GitHub: @caohe
- **任强**
  - Email: qiang.ren@intel.com
  - GitHub: @Airren


# Thanks.