
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023


KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT China 2023

使用可插拔和可定制的智能运行时提升工作负载的QoS

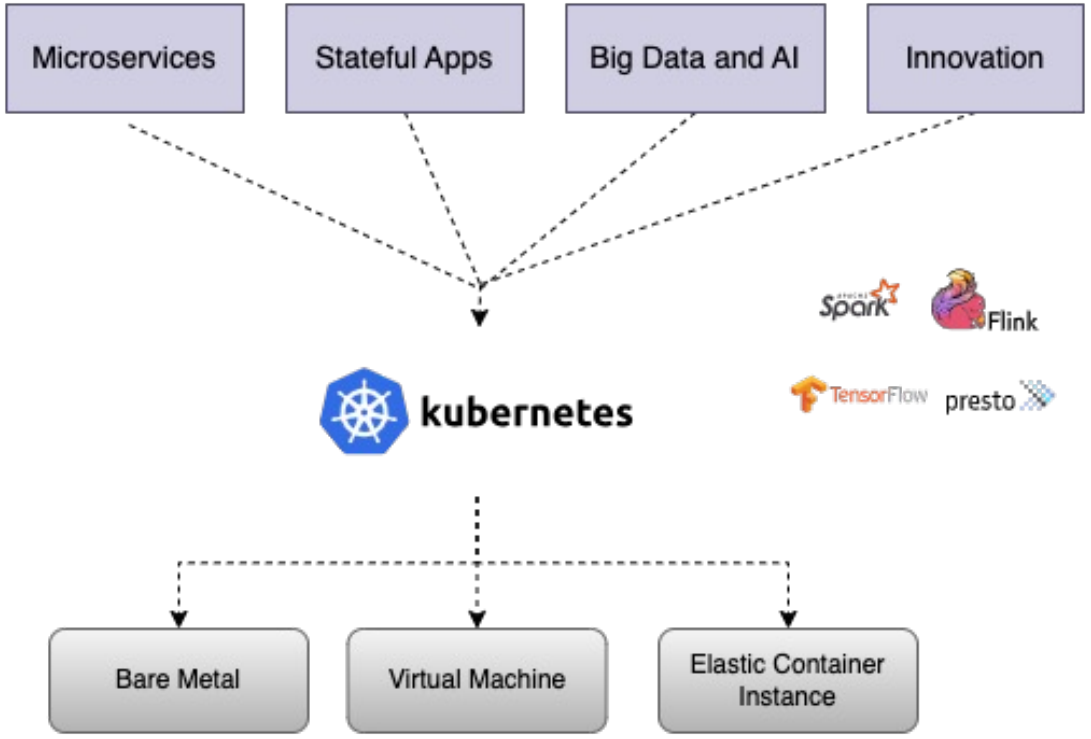
韩柔刚- 阿里云
张康 – 英特尔

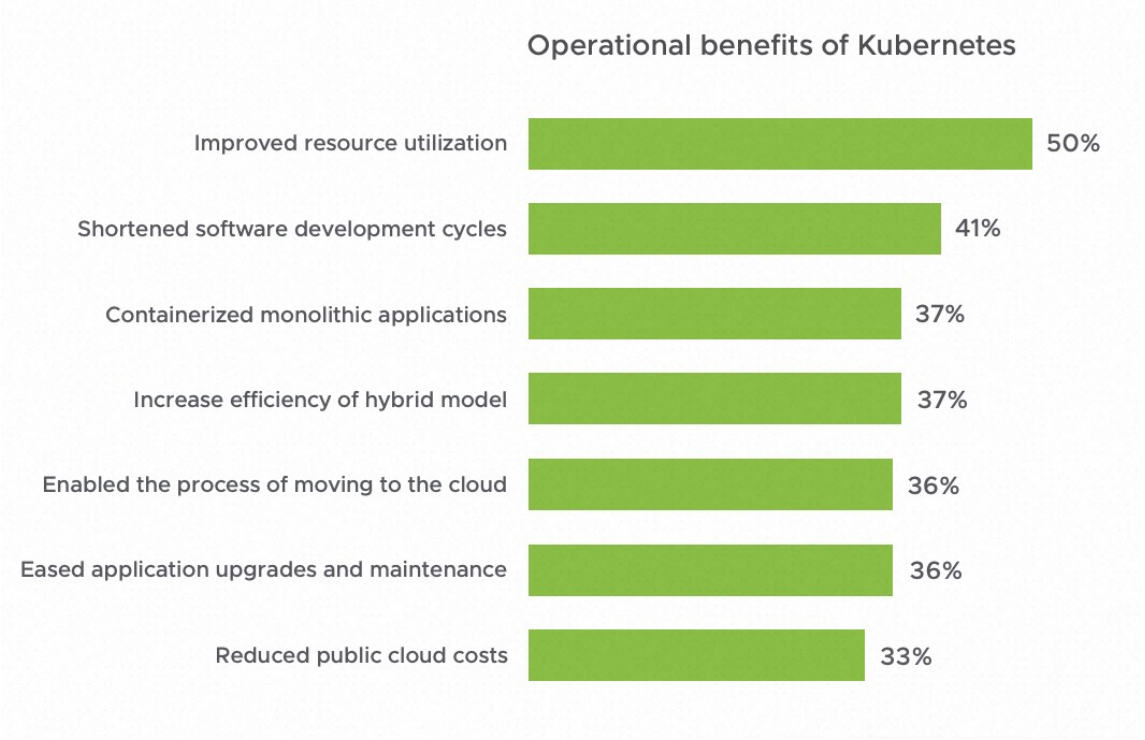
概要

1. 介绍
2. 增强Pod QoS的方法
3. 集成NRI到Koordinator上
4. 示例
5. 将来的工作
6. Q & A

介绍

大规模Kubernetes集群的**工作负载整合**和**高利用率**的现状.

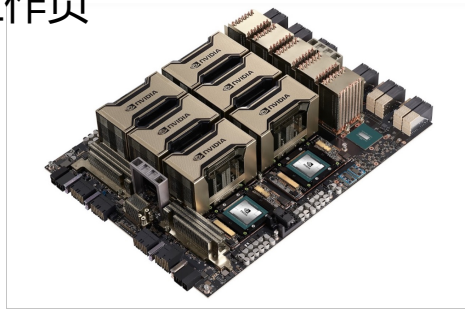




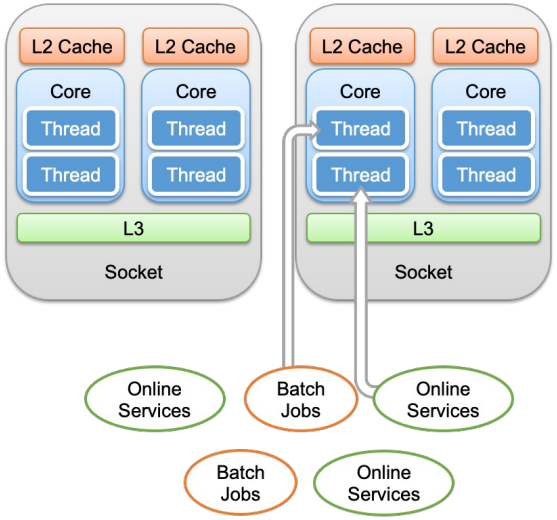
State of Kubernetes 2023, vmware[1]

介绍

- **工作负载整合**带来多元化的资源需求
  - 在线服务工作负载通常对延迟敏感，其资源使用情况会随着时间的推移而波动，而某些批处理作业工作负载是计算密集型的，以实现高吞吐量、资源质量容忍度
  - AI 训练任务根据硬件拓扑的不同分配GPU和RDMA资源等异构设备



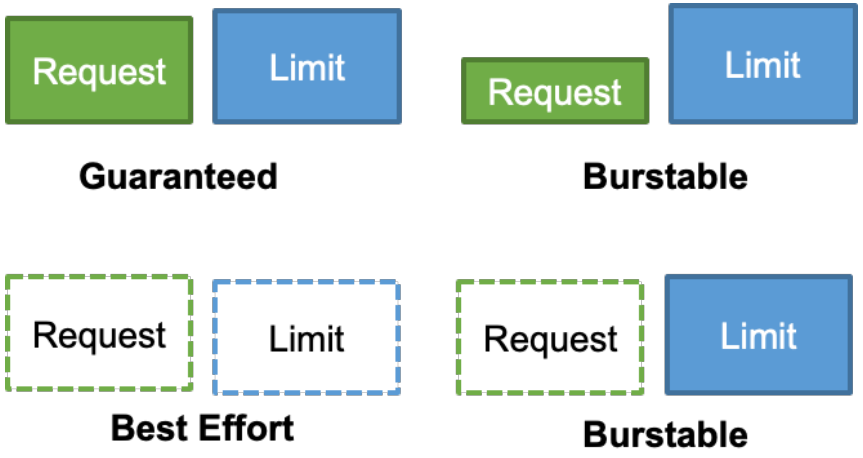
- **高资源利用率**会增加Pod间和Pod内资源争用的风险
  - 工作负载混部：将在线服务和批处理作业部署在同一个集群甚至同一个节点上
    - 通过时分复用提高资源利用率
    - 但也引入了近邻干扰问题
      - 资源超发。例如CPU
      - 资源缺乏隔离。例如末级缓存，内存带宽

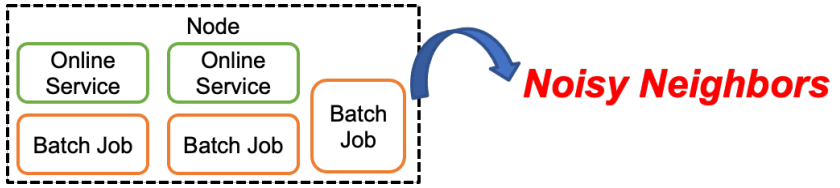


关键挑战: 节点资源管理

介绍

- Kubernetes 1.28提供的资源模型
- 资源请求
  - **Requests**: 保证得到的资源
  - **Limits**: 资源使用的上限
- 服务质量等级
  - **Guaranteed**: requests & limits都设置了而且相等
  - **Burstable**: 非Guaranteed并且至少一个容器有内存或CPU的request或limit
  - **Best Effort**: request和limit都没设置





- 应用混部?
  - **在线服务** -> Guaranteed/Burstable, 为了更高的资源
  - **批处理任务** -> Best Effort, 为了提高集群资源利用率
- 没有应用喜欢混部
  - 在线应用受到干扰
    - SMT, LLC, 内存带宽受到竞争
  - 批处理任务性能变差.
    - 没有可参考的BE容量
    - BE Pods之间没有公平
- 如果作业要求分配GPU等设备资源怎么办?
  - 可选方案
    - Device Plugin + 包装的运行时
    - Dynamic Resource Allocation
  - 关于设备拓扑感知和QoS增强

为了提升Pod服务质量.....

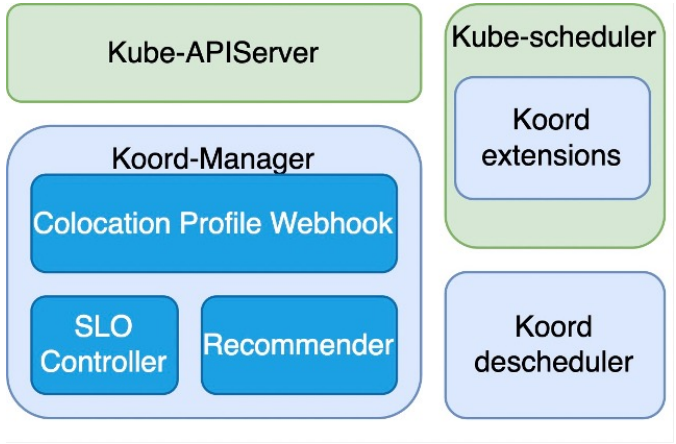
1. **增强资源模型**以实现更好的应用混部
2. 利用**可插拔机制**来管理更多资源类型

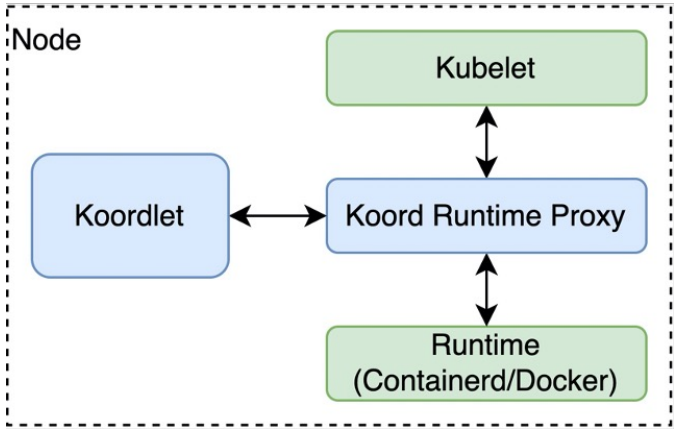
增强Pod QoS的方法

增强 Pod QoS 的方法

Koordinator, 是一个基于QoS的Kubernetes混合工作负载调度系统

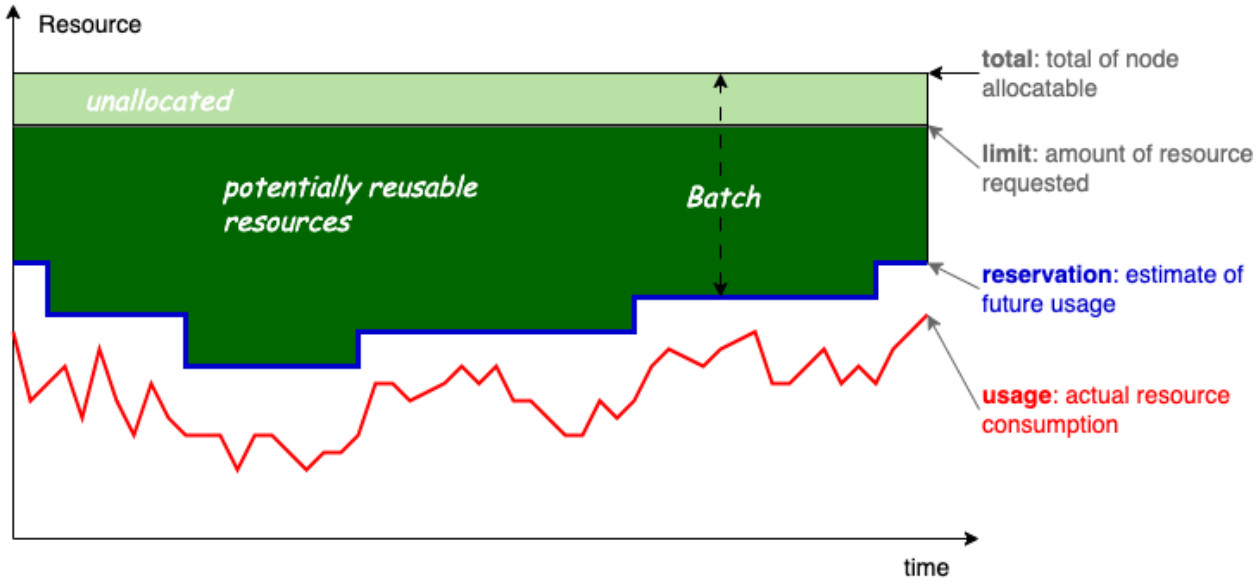
- https://github.com/koordinator-sh/koordinator
- 旨在提高Kubernetes集群的资源利用率和性能，减少容器之间的干扰.
- 组件
  - 控制面
    - Koord-Manager: 管理QoS策略并计算可回收资源
    - Koord-Scheduler extensions: 针对QoS感知、作业调度和异构资源调度的调度插件
    - Koord-descheduler: 重调度插件用来提高应用的服务等级协议和节点负载均衡
  - 节点
    - Koordlet:分析节点和Pod的资源、调整容器资源以增强QoS
    - Koord-runtime-proxy: Kubelet和容器运行时之间的代理，允许Koordlet注入资源参数





增强 Pod QoS 的方法

- 增强的资源模型和工作负载混部策略
  - **BatchResource**: 可量化的BE资源
    - BE 容量: 为Batch Pods分配可回收资源
    - BE limit & requests: 管理Batch资源的cgroups




```
apiVersion: v1
kind: Pod
metadata:
  labels:
    koordinator.sh/qosClass: BE
    spark-role: driver
  ...
spec:
  containers:
  -  args:
      - driver
      - --properties-file
      - /opt/spark/conf/spark.properties
      - --class
      - org.apache.spark.examples.SparkTC
      - local:///opt/spark/examples/jars/spark-examples_2.12-3.2.1-tc1.2.jar
     resources:
       limits:
         kubernetes.io/batch-cpu: "1000" # milli-cores
         kubernetes.io/batch-memory: 3456Mi
       requests:
         kubernetes.io/batch-cpu: "1000" # milli-cores
         kubernetes.io/batch-memory: 3456Mi
  ...
```

增强Pod QoS的方法

- 增强的资源模型和工作负载混部策略
  - **ResctrlQoS**: 用resctrl来限制末级缓存和内存带宽
    - 机制: 基于末级缓存和内存带宽控制的硬件特性 (Intel RDT, AMD QoS, ARM MPAM, ...)
    - 当前策略: 根据 QoS 进行静态划分。 BE Pods会被限制
  - **CPUSetAllocator**: 细粒度的CPU编排
    - NUMA级别的cpu共享池: 绑定BE Pod的cpu亲和到一个或多个 NUMA 节点
  - ...

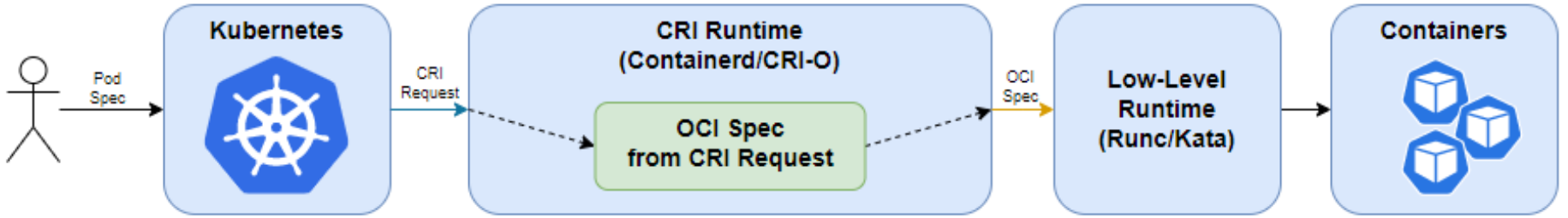
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: slo-controller-config
  namespace: koordinator-system
data:
  resource-qos-config: |
    {
      "clusterStrategy": {
        "lsClass": {
          "resctrlQOS": {
            "enable": true,
            "catRangeEndPercent": 100,
            "mbaPercent": 100
          }
        },
        "beClass": {
          "resctrlQOS": {
            "enable": true,
            "catRangeEndPercent": 30,
            "mbaPercent": 30
          }
        }
      }
    }
```

增强Pod QoS的方法

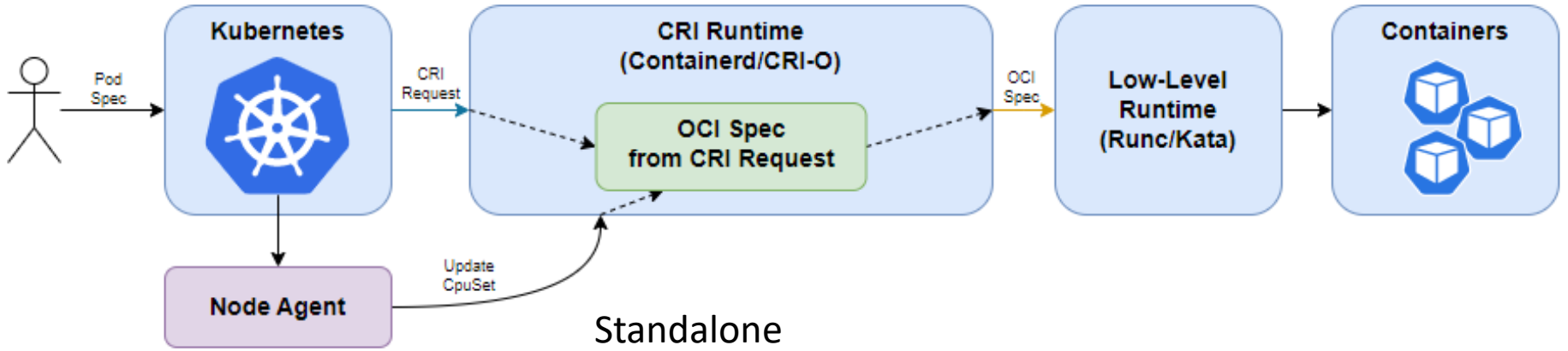
- 利用可插拔机制来管理更多资源类型
  - 要求
    - 管理常见的cgroup资源(例如 cpuset, cfs quota, memory limit)
      - 避免与Kubelet和容器运行时发生冲突 (containerd, cri-o, oci)
      - 对Kubelet和容器运行时没有侵入性
      - 可以同步容器状态
    - 管理out-of-tree资源 (例如龙蜥OS的Group Identity特性)
  - 备选方案
    - Standard CRI: Kubelet和容器运行时的默认容器资源管理
    - Standalone: 基于标准CRI，代理直接读写cgroup资源
    - CRI Proxy: 在Kubelet和CRI 运行时之间运行代理并注入CRI参数
    - NRI: 通过容器运行时的NRI框架注入和调整OCI spec

集成NRI到Koordinator上

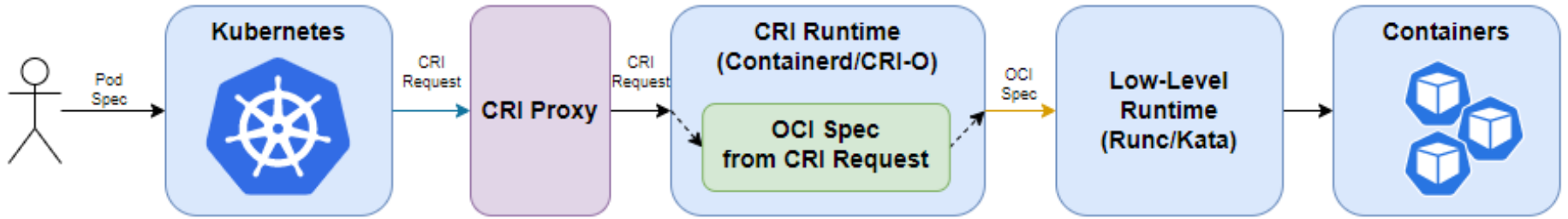
使QoS资源生效的不同方法



Standard CRI



Standalone

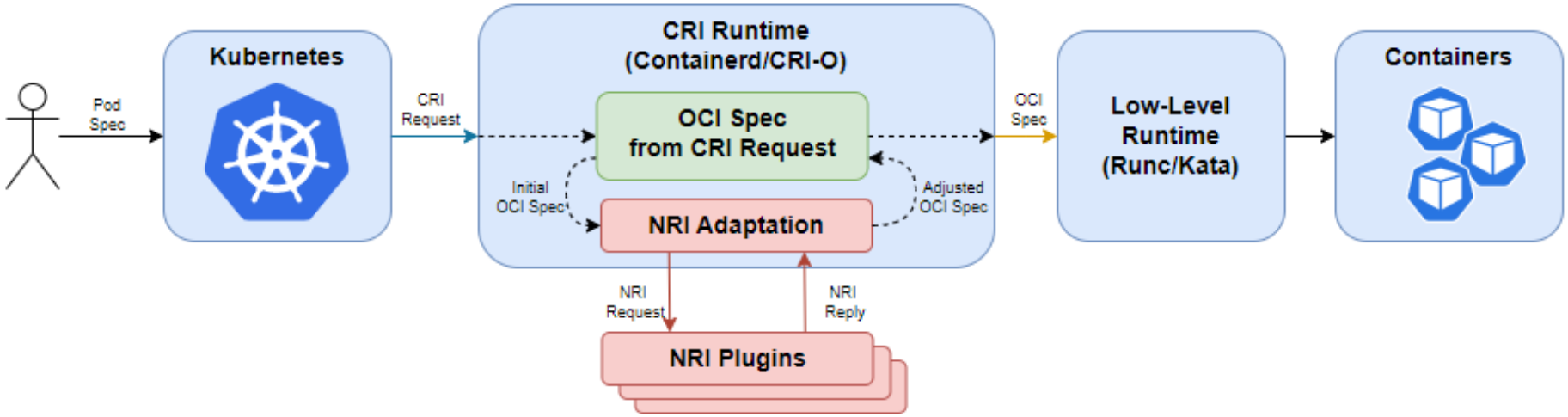


CRI proxy

使QoS资源生效的不同方法

NRI (节点资源管理) 是一个通用的框架，它可以

- 将扩展插件插入OCI兼容的运行时
- 实现自定义容器资源配置



比较使QoS资源生效的不同方法

	Standalone	Proxy	NRI
功能完整性	不支持GPU 环境变量注入	所有	所有
可操作性	对原生组件没有侵入性	对Kubelet有侵入性	对原生组件没有侵入性
实时性	N	Y	Y
前置条件	N	N	容器运行时支持NRI
可适用性	K8s和非K8s环境	K8s 环境	K8s和非K8s环境

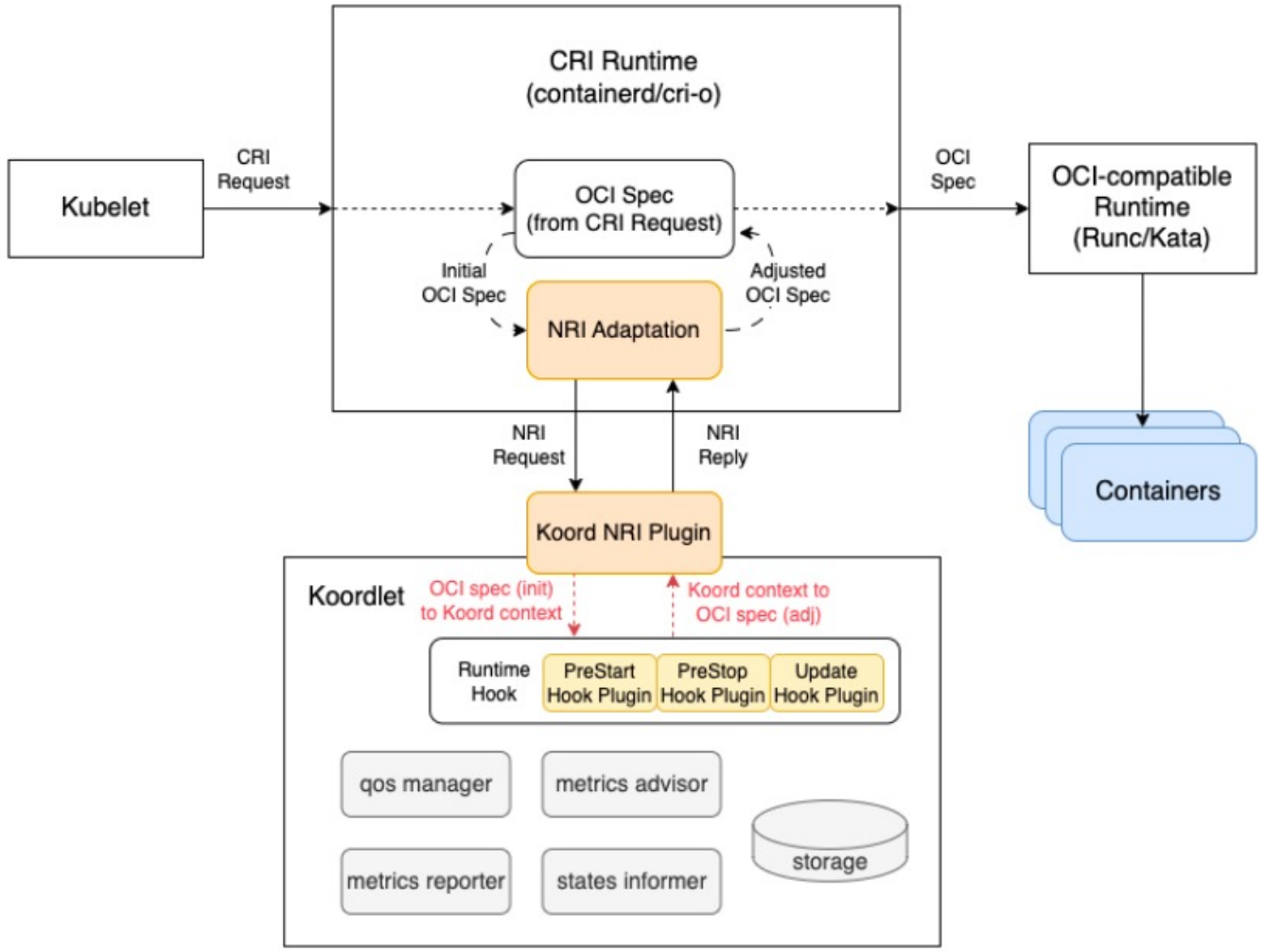
NRI在KubeCon+CloudNativeCon的演进

- KubeCon + CloudNativeCon EU 2021
  - Maximizing Workload'sPerformance With Smarter Runtimes - Krisztian Litkey & Alexander Kanevskiy,Intel
- KubeCon + CloudNativeCon NA 2022
  - NRI: Extending Containerd And CRI-O With Common Plugins - Krisztian Litkey, Intel & Mike Brown, IBM
- KubeCon + CloudNativeCon China 2023
  - 利用集成了NRI的Koordinator来具有洞察力的解决真实世界的问题

Koordinator on NRI的设计

https://koordinator.sh/blog/release-v1.3.0

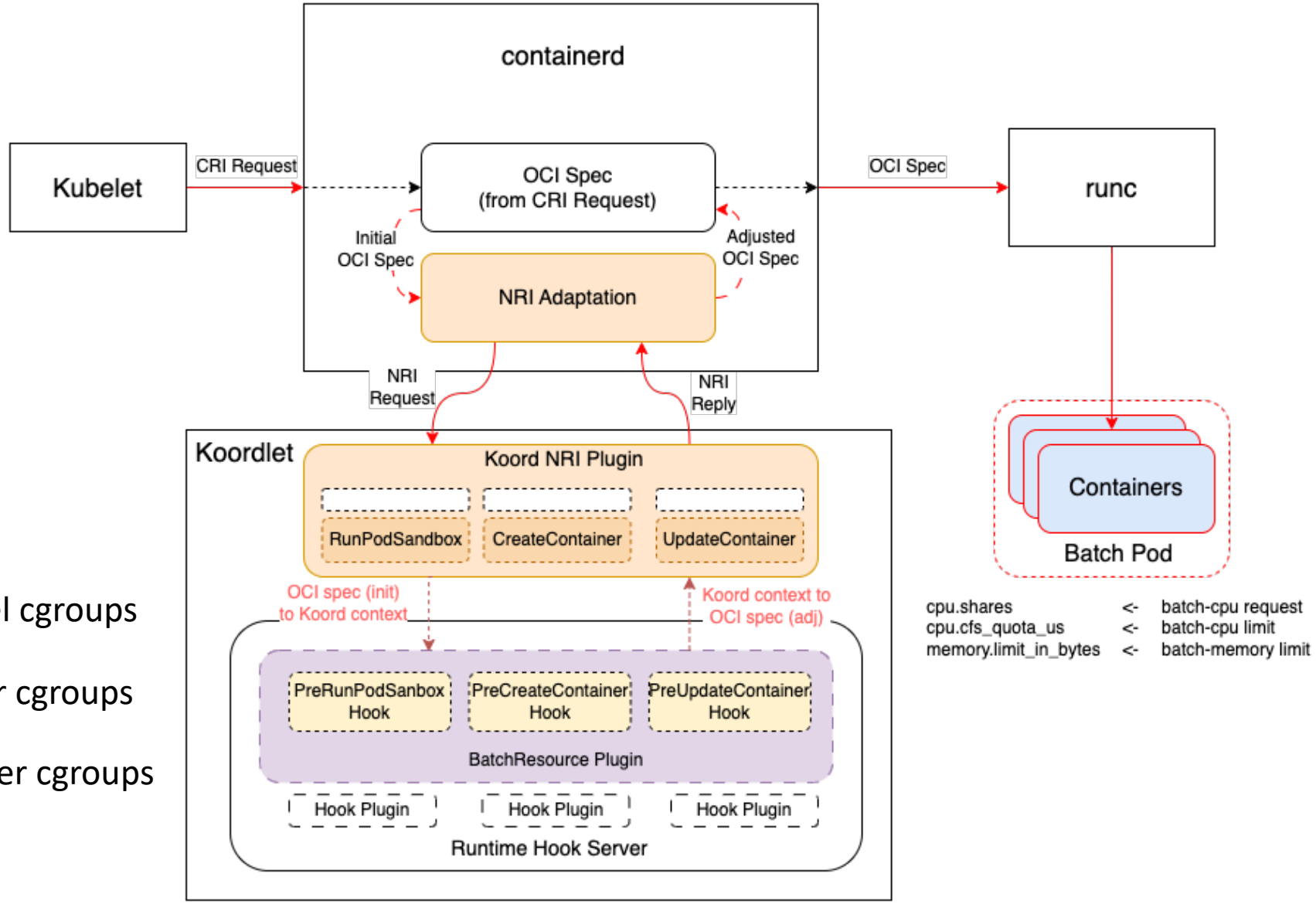
- Koordlet作为一个NRI plugin使用
- NRI Request
  - 把OCI spec转成Koord protocols
- NRI Reply
  - 把Koord protocols转成OCI spec
- 订阅NRI事件
  - RunPodSandbox
  - CreateContainer
  - UpdateContainer



挂载插件BatchResource的工作流程

BatchResource插件

- Cgroups
  - cpu.cfs_quota_us
  - cpu.shares
  - memory.limit_in_bytes
- Hook Events
  - RunPodSandbox: 设置 pod-level cgroups
  - CreateContainer: 注入container cgroups
  - UpdateContainer: 注入 container cgroups



示例

示例

- 使用Koordinator部署在离线混部应用
- Koordinator示例：nginx+ffmpeg混部 - asciinema

将来的工作

将来的工作

- 总结
  - 为云原生系统提供优雅的资源管理
  - 促进混部技术标准化
  - 提高Koordinator部署灵活性和处理的及时性

- Koordinator将来的工作
  - 集成新的NRI特性并为out-of-tree内核特性改进NRI框架
  - 平台感知QoS增强，如RDT的高阶使用
  - 节点资源放大
  - 基于QoS的拓扑感知调度
  - 异构设备管理
  - 等价类调度

Q & A

- 联系我们:
  - Rougang Han, rougang.hrg@alibaba-inc.com
  - Kang Zhang, kang.zhang@intel.com

- Koordinator & NRI相关资料:
  - https://github.com/koordinator-sh/koordinator
  - https://koordinator.sh/blog/release-v1.3.0
  - https://github.com/containerd/nri

- 问题和建议

谢谢!