
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

Enhance Workload QoS With Pluggable and Customizable Smarter Runtimes

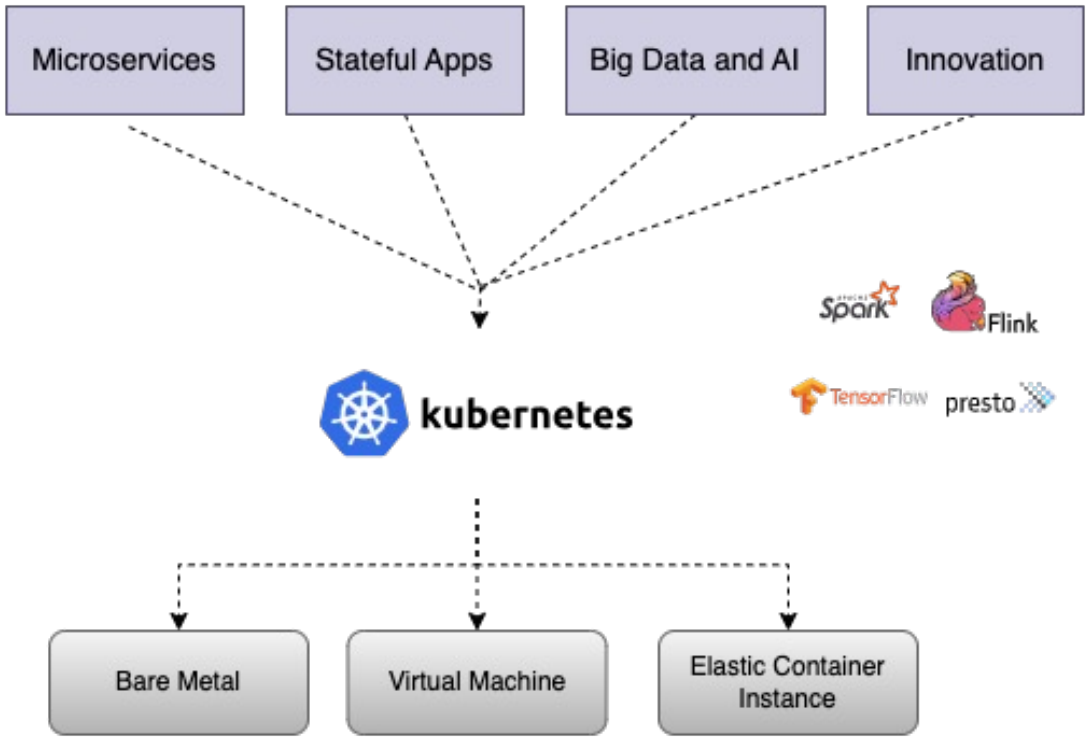
Rougang Han - Alibaba Cloud
Kang Zhang - Intel

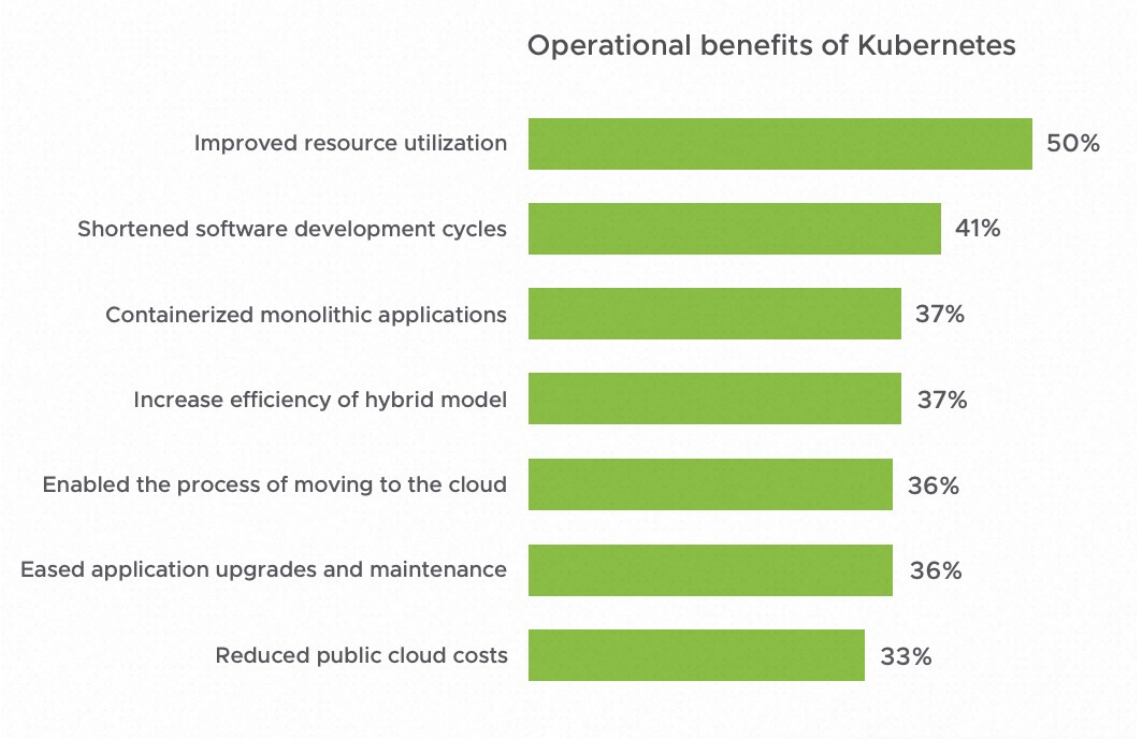
Overview

1. Introduction
2. Approaches to enhance Pod QoS
3. Koordinator on NRI
4. Demo
5. Future work
6. Q & A

Introduction

The state of **workload consolidation** and **high utilization** of large Kubernetes clusters.

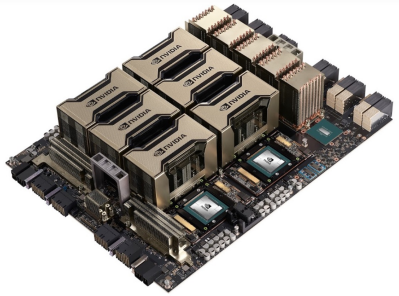




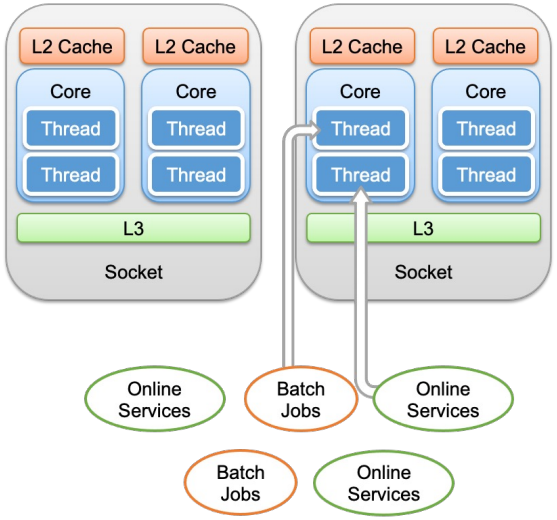
State of Kubernetes 2023, vmware[1]

Introduction

- **Workload consolidation** brings diversified resource requirements.
  - **Online service** workloads are generally **latency-sensitive** and their resource usage fluctuates over time, while some **batch job** workloads are **computation-intensive** for high throughput, resource quality tolerance.
  - AI training tasks allocate **heterogenous devices** like GPU and RDMA resources, **varying** on hardware topologies.



- **High utilization** increases the risks of inter-pods and intra-pods resource contention.
  - Workloads co-location: deploying online service and batch jobs in the same cluster or even on the same node.
    - Improve the resource utilization via Time-Division Multiplexing.
    - But also introduce the Noisy Neighbor problem since
      - Resources are overcommitted.e.g. CPU
      - Resources lake of isolation. e.g. last-level cache, memory bandwidth

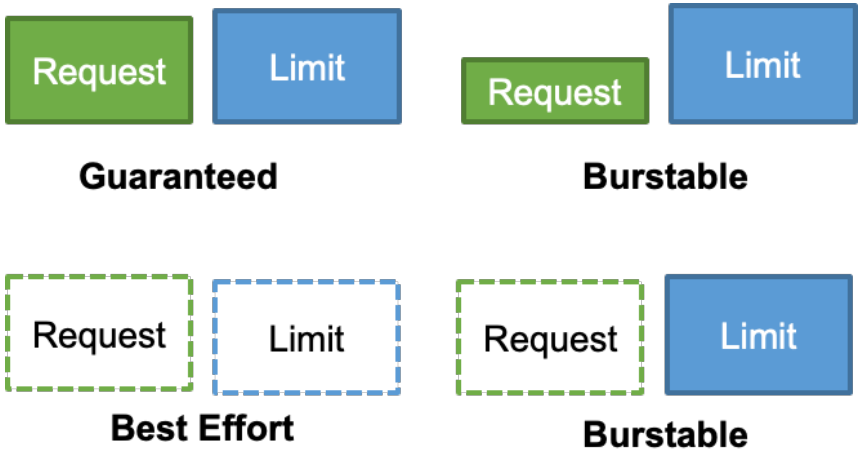


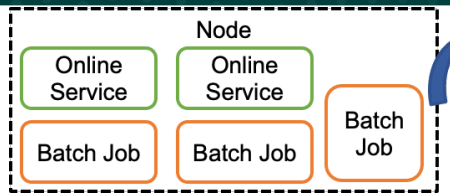
Key Challenge : Node Resource Managment

Introduction

- Resource models offered by Kubernetes 1.28

- Resource requirements
  - **Requests**: resources guaranteed to get.
  - **Limits**: resources usage never goes above.
- QoS (Quality of Service) Class
  - **Guaranteed**: requests & limits are both set and equal.
  - **Burstable**: Not Guaranteed, and at least one container in the pod has a memory or CPU request or limit.
  - **Best Effort**: request & limit are both not set.





Noisy Neighbors

- How about ... for co-location?
  - **Online Services** -> Guaranteed/Burstable, for high-quality resources.
  - **Batch Jobs** -> Best Effort, for cluster resource utilization improvement
- No one is happy for co-location.
  - Online service got interference.
    - SMT, LLC, memory bandwidth resources are contented.
  - Batch job have bad performance.
    - NO reference for Node BE capacity.
    - NO fairness over BE pods.
- What if the job allocates device resources like GPU?
  - Alternatives
    - Device Plugin + wrapped runtime
    - Dynamic Resource Allocation
  - For device topology-awareness and QoS enhancement

To improve the Pod QoS...

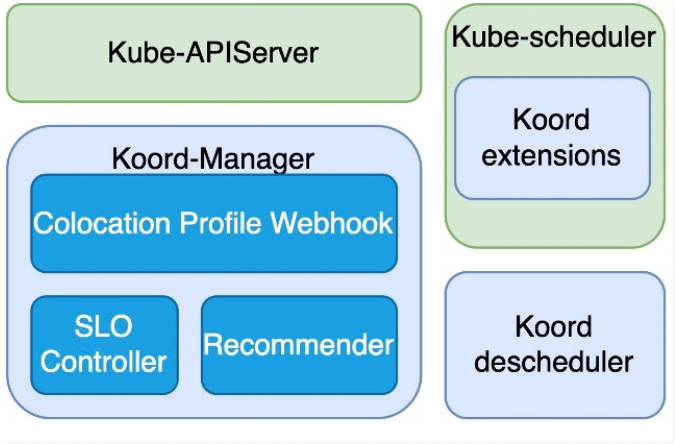
1. **Enhanced resource model** for better colocation
2. **Pluggable mechanism** to manage more resource types

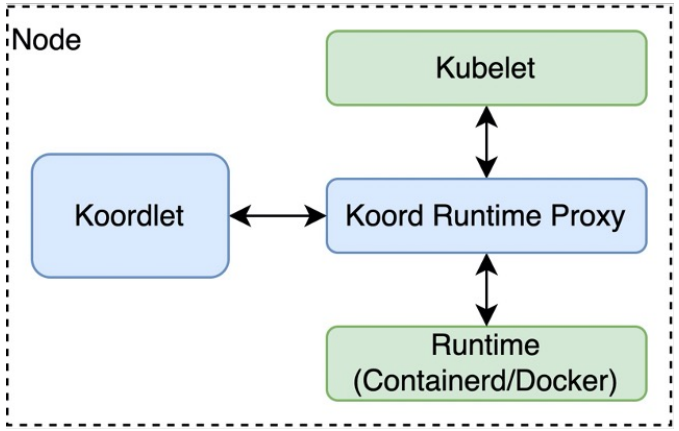
Approaches to enhance Pod QoS

Approaches to enhance Pod QoS

Koordinator, a QoS based scheduling system for hybrid workloads orchestration on Kubernetes.

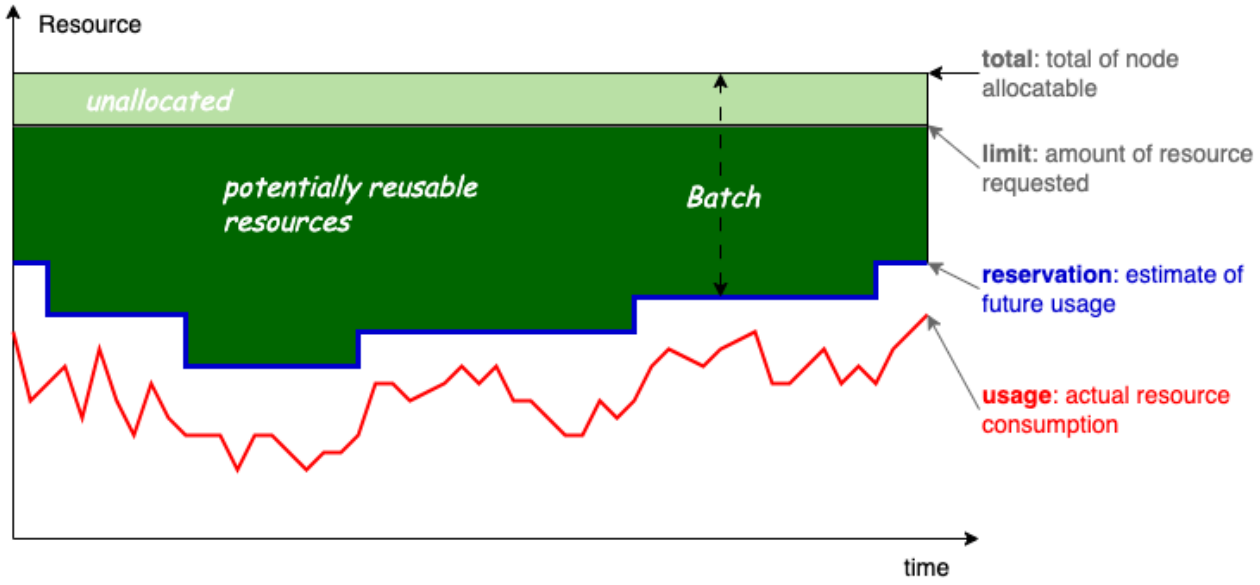
- https://github.com/koordinator-sh/koordinator
- Aim to improve the resource utlization and the performance of Kubernetes clusters, reduce interference between containers.
- Components
  - Control Plane
    - Koord-Manager: Manage QoS strategies and calculate reclaimable resources.
    - Koord-Scheduler extensions: Scheduler plugins for QoS-aware scheduling, job scheduling and heterogeneous resource scheduling.
    - Koord-descheduler: Descheduling plugins to improve apps' SLA and node rebalancing.
  - Node
    - Koordlet: Resource profiling for node and pods, container resource tuning for QoS enhancement.
    - Koord-runtime-proxy: Proxy between Kubelet and container runtime, allow Koordlet to inject resource params.





Approaches to enhance Pod QoS

- Enhanced resource model and strategies for workload colocation
  - **BatchResource**: Quantified BE resources.
    - BE capacity: Schedule reclaimable resources for Batch pods.
    - BE limit & requests: Manage cgroups for Batch resources.




```
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
  ...
```

Approaches to enhance Pod QoS

- Enhanced resource model and strategies for workload colocation
  - **ResctrlQoS:** Limit last-level cache and memory bandwidth with resctrl.
    - Mechanism: Based on the hardware features of LLC and MBW control (Intel RDT, AMD QoS, ARM MPAM, ...)
    - Current strategy: Static partitioning according to QoS. BE pods are limited.
  - **CPUSetAllocator**: Fine-grained CPU orchestration.
    - NUMA-level cpushare pool: Bind BE pod's cpu affinity to one or more NUMA nodes.
  - ...

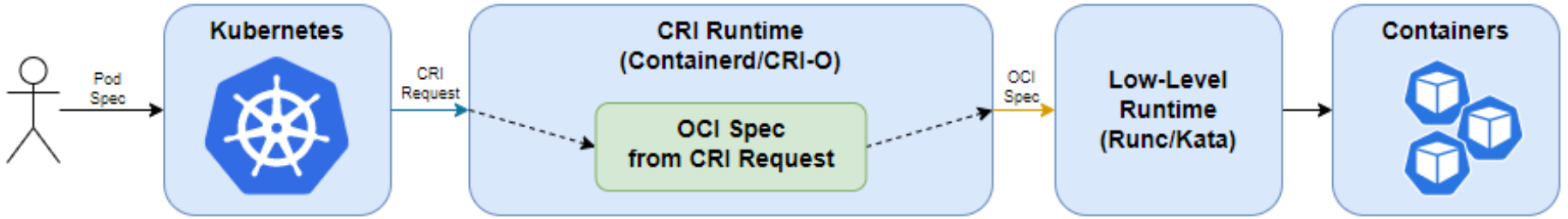
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: slo-controller-config
  namespace: koordinator-system
data:
  resource-qos-config: |
    {
      "clusterStrategy": {
        "lsClass": {
          "resctrlQOS": {
            "enable": true,
            "catRangeEndPercent": 100,
            "mbaPercent":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

Approaches to enhance Pod QoS

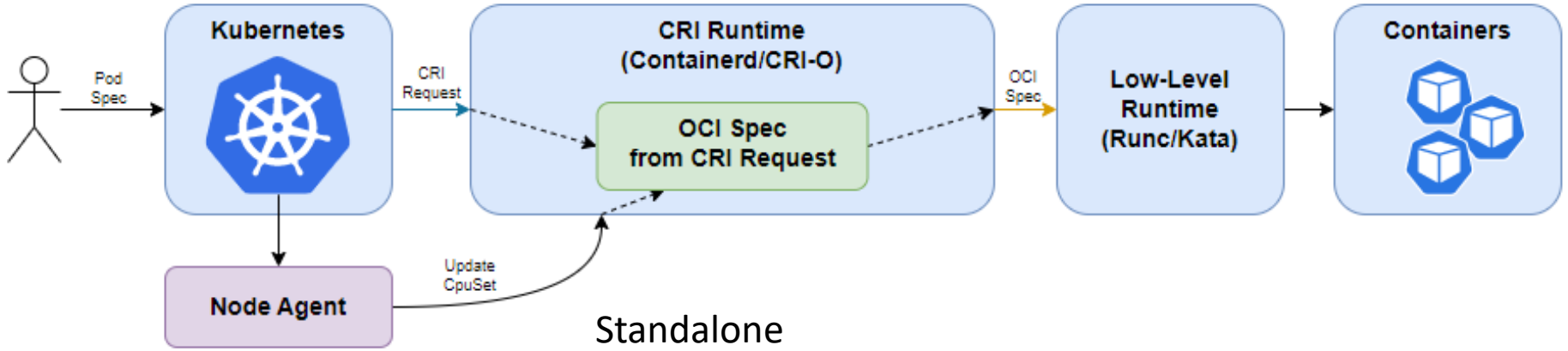
- Pluggable mechanism to manage different types of resources.
  - Requirements
    - Manage common cgroup resources (e.g. cpuset, cfs quota, memory limit)
      - Avoid conflicts with Kubelet and container runtime (containerd, cri-o, oci).
      - Non-invasive to Kubelet and container runtime.
      - Synchronizable to the container states.
    - Manage out-of-tree resources (e.g. Group Identity feature powered by Anolis OS kernel).
  - Alternatives
    - Standard CRI: Default container resource management by Kubelet and container runtime.
    - Standalone: Based on Standard CRI, the agent reads/writes cgroup resources directly.
    - CRI Proxy: Proxy and inject CRI params between Kubelet and CRI runtime.
    - NRI: Inject and adjust OCI spec via container runtime's NRI framework.

Koordinator on NRI

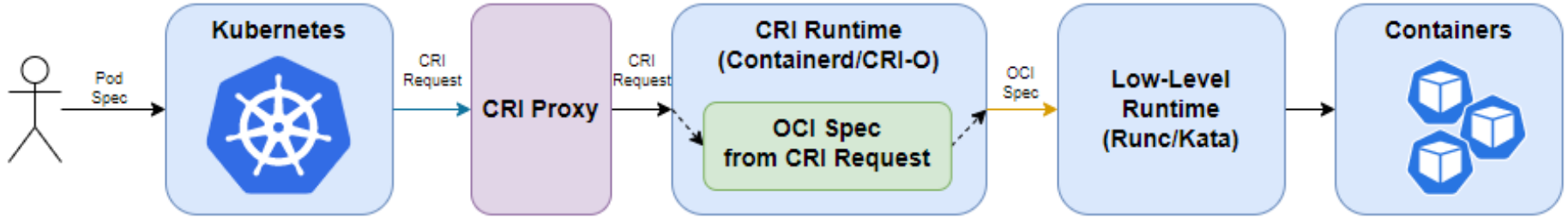
Different approaches to apply QoS resources



Standard CRI



Standalone

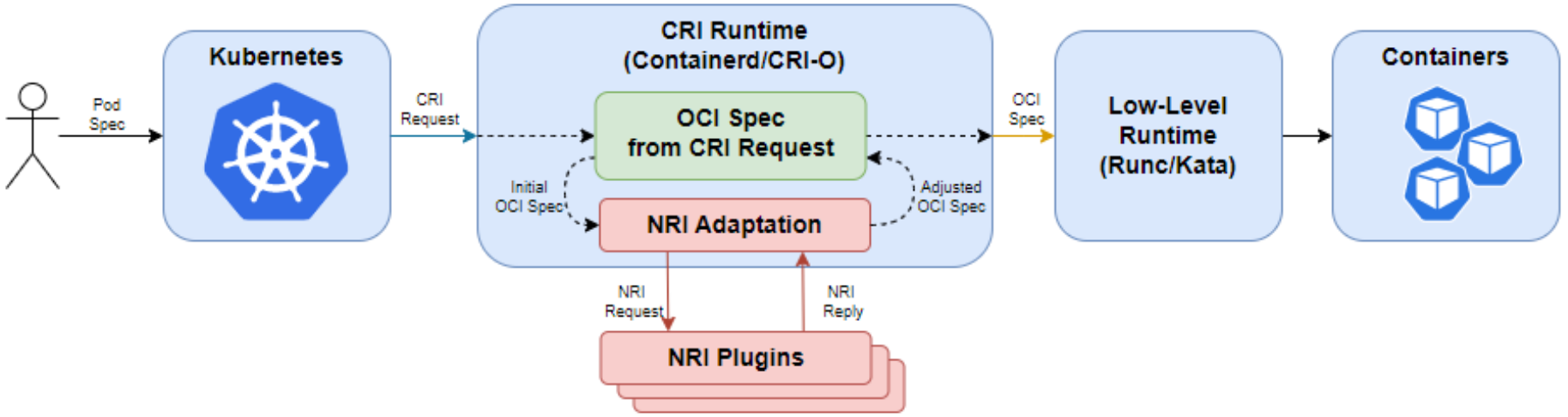


CRI proxy

Different approaches to apply QoS resources

NRI (Node Resource Interface) is a common **framework** for

- **plugging** extensions into OCI-compatible runtimes
- implementing **custom** container configuration logic



Comparing Different Approaches

	Standalone	Proxy	NRI
Functional Completeness	GPU environment inject not support	all	all
Operability	non-intrusive to native components	intrusive to kubelet	non-intrusive to native components
Real-time	N	Y	Y
Prerequisites	N	N	Container Runtime support NRI
Applicability	non-k8s and k8s envrionment	k8s envrionment	non-k8s and k8s envrionment

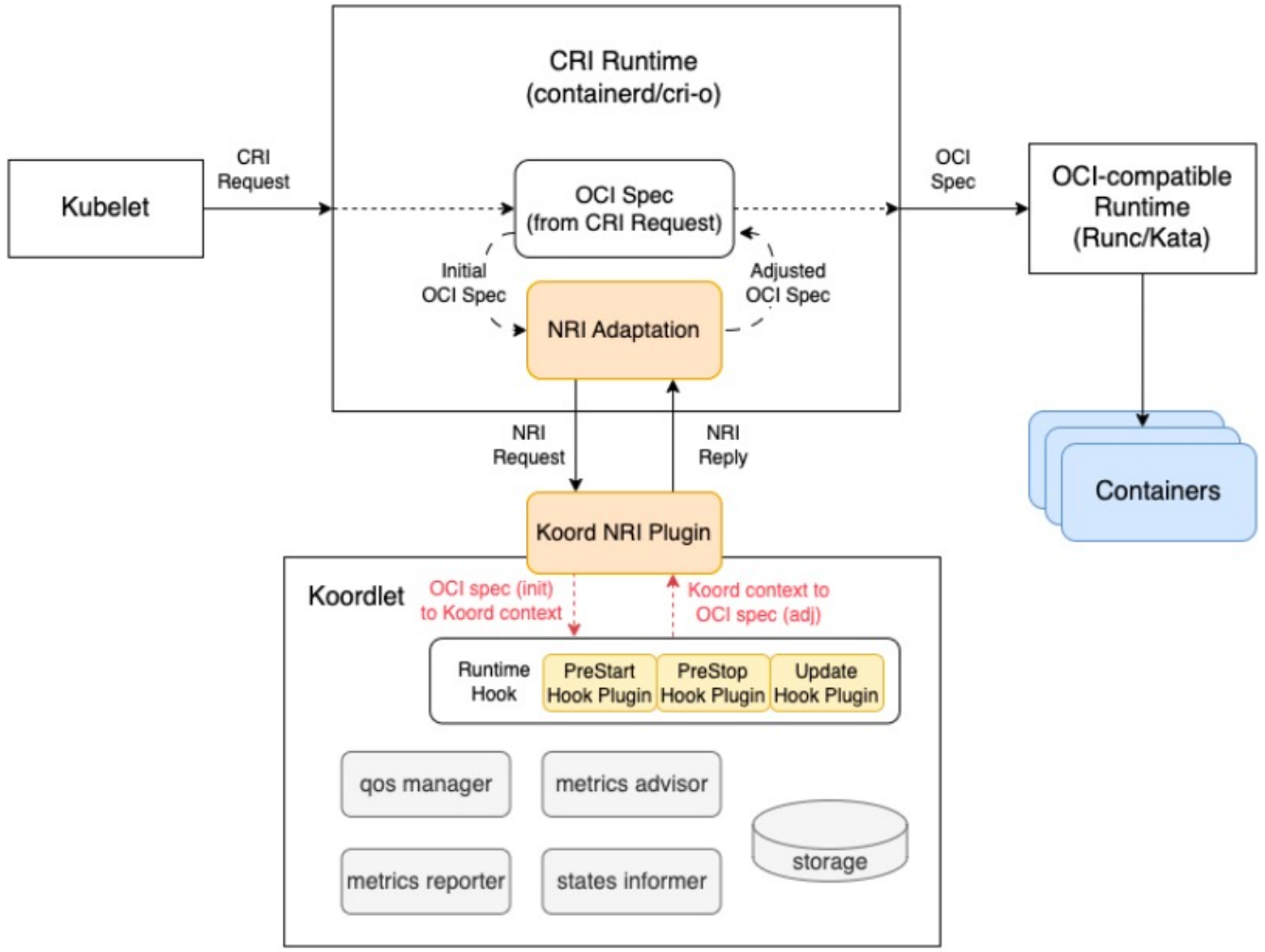
NRI Evolution in KubeCon+CloudNativeCon

- 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
  - What's ***New*** in this topic: An integration of NRI on koordinator to address ***real world*** problem with insights

Design of Koordinator on NRI

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

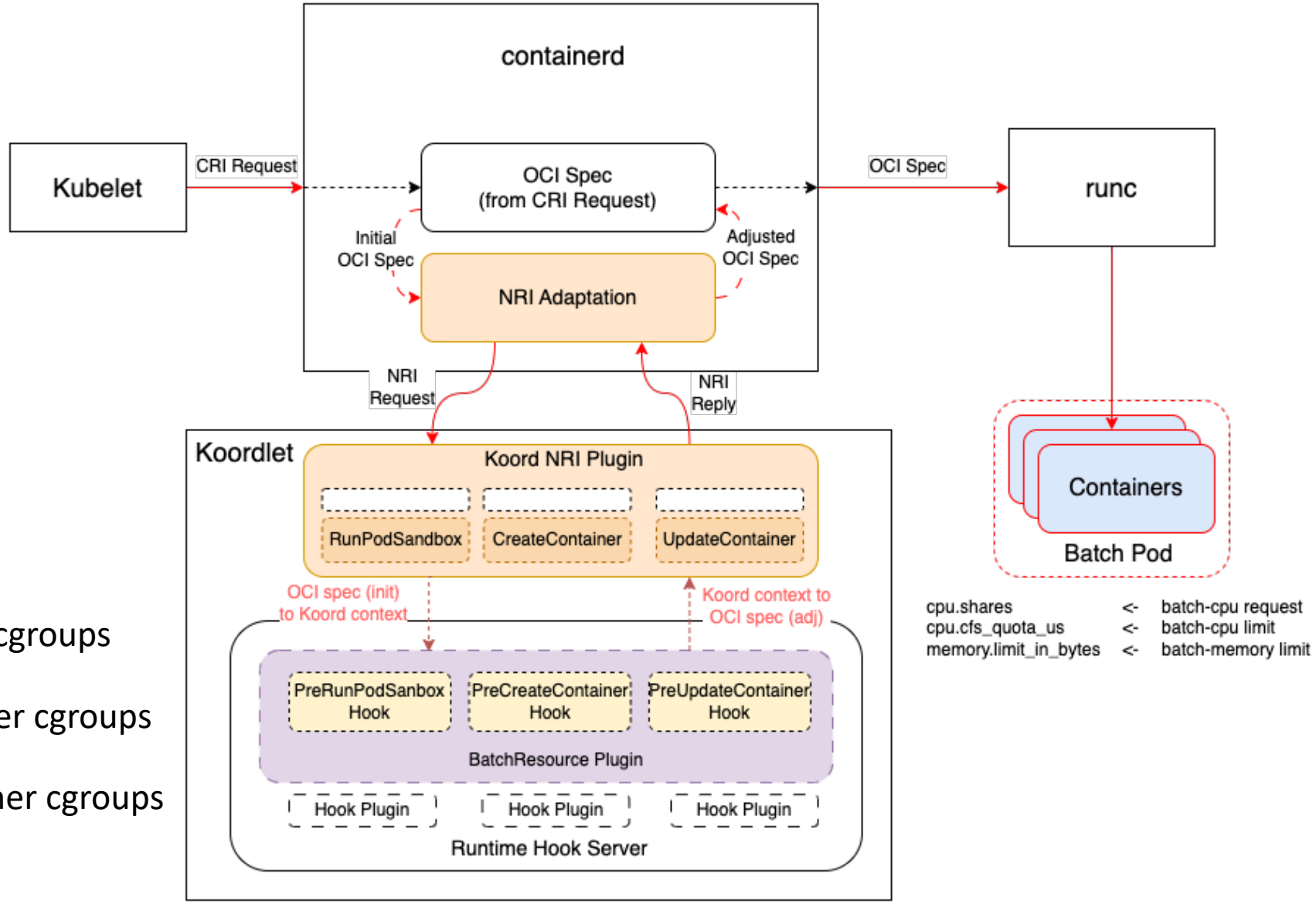
- Koordlet works as NRI plugin
- NRI Request
  - Transform OCI spec into Koord protocols
- NRI Reply
  - Transform Koord protocols into OCI spec
- Subscribe NRI Events
  - RunPodSandbox
  - CreateContainer
  - UpdateContainer



Workflow of hook plugin BatchResource

BatchResource Plugin

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



Demo

Demo

- Lab - Deploy colocated online and offline apps with Koordinator
- Koordinator Demo: nginx+ffmpeg colocation [Short Version] - asciinema

Lab QR code

Future Work

Future work

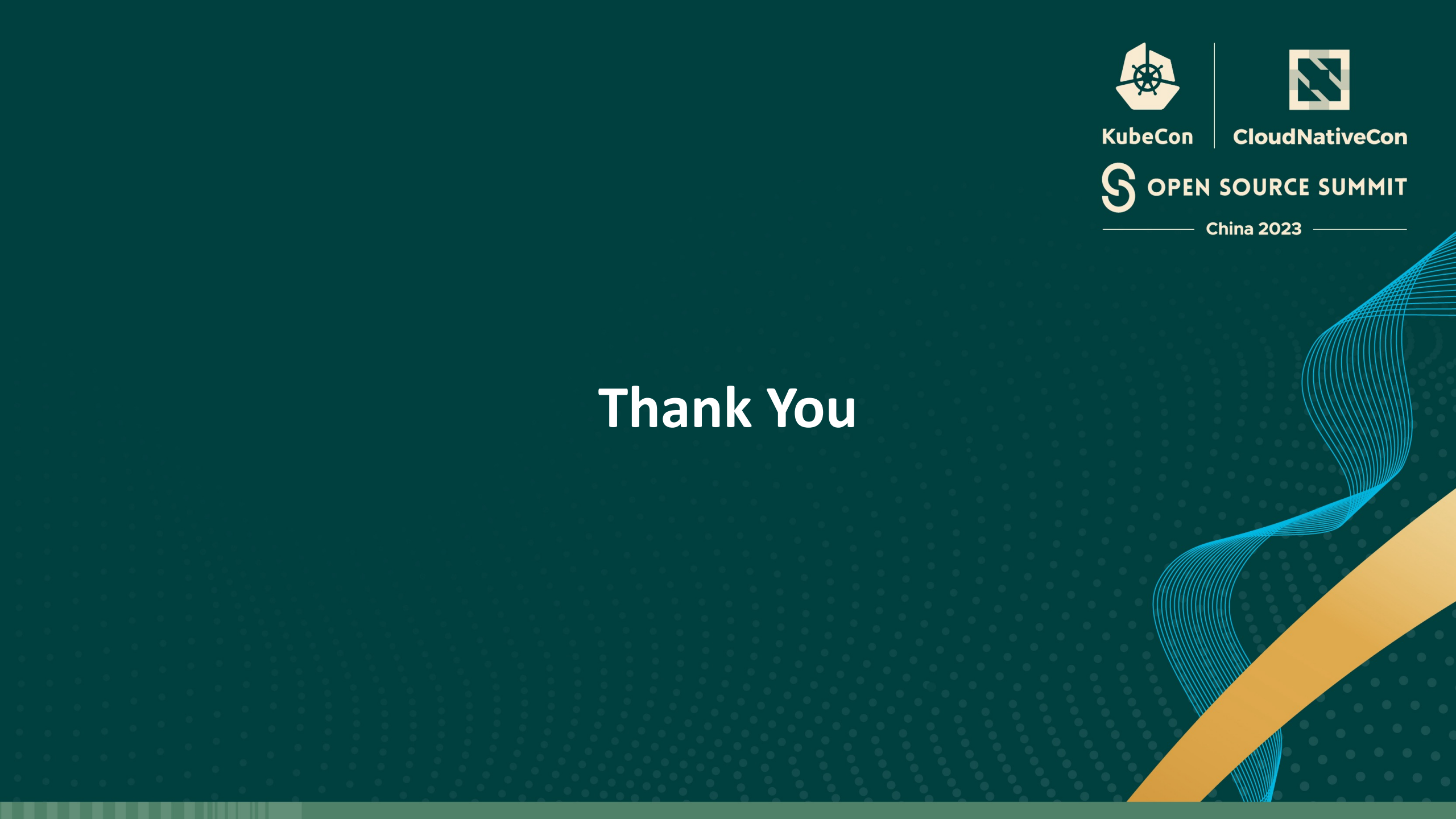
- Summary
  - Propose an elegant resource management for cloud native system
  - Promote the standardization of co-location technology
  - Improve the flexibility and timeliness of Koordinator deployment and processing
- Koordinator future work
  - Integrate new NRI feature and improve NRI framework for out-of-tree kernel features
  - Platform-aware QoS enhancement such as advanced RDT usage
  - Node resource amplification
  - QoS-based topology-aware scheduling
  - Heterogeneous device scheduling and management
  - Equivalence class scheduling
- Call for action

Q & A

- Contact info:
  - Rougang Han, rougang.hrg@alibaba-inc.com
  - Kang Zhang, kang.zhang@intel.com

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

- Questions & Comments


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Thank You