
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023


KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023

Building a Fine-Grained and Intelligent Resource Management System on Kubernetes

He Cao & Wei Shao

ByteDance

Agenda

- Katalyst Overview
- Application Scenarios
  - Colocation
  - GPU-sharing Scheduling
  - Topology-aware Scheduling
  - Resource Efficiency Suite
- Results
- Community

1 Katalyst Overview

ByteDance's Workloads

Microservices

- Implementing app business logic
- Golang
- Complex service invocation chains
- RPC latency is the primary performance metric
- CPU as dominant resource

Search, Ads, Recommendation

- Backend services that provide content lists for the feed and search
- Real-time online inference
- C++
- Extremely high performance requirements
- High resource consumption

AI and Big Data

- Offline training jobs that support search, ads, and recommendation
- Data processing jobs used to provide data reports
- Video transcoding tasks
- Throughput is the primary performance metric
- High memory consumption

Storage

- Traditional storage, databases, NoSQL, etc
- Stateful
- Large blast radius for failures
- High demand for resource stability

ByteDance ❤️ Kubernetes

900,000+
nodes

110M+
pods

Supporting
hundreds of
millions of users

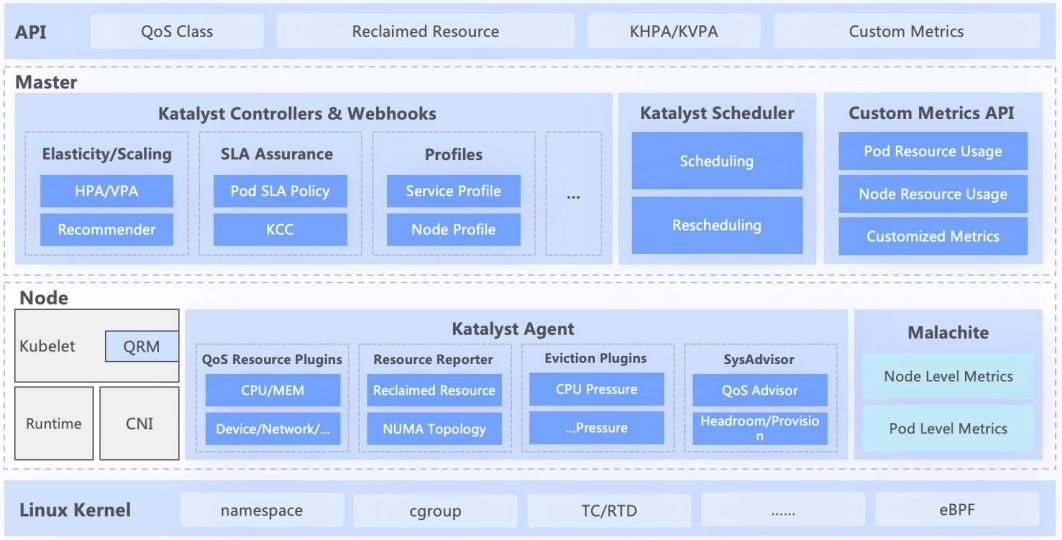
6M+
deployments

300,000+
jobs

Katalyst Overview

Katalyst

Katalyst, derived from the “catalyst” in chemical reactions, provides enhanced resource management capabilities for workloads running on Kubernetes.



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

Master

- Katalyst Controllers & Webhooks
- Katalyst Scheduler
- Katalyst Custom Metric

Node

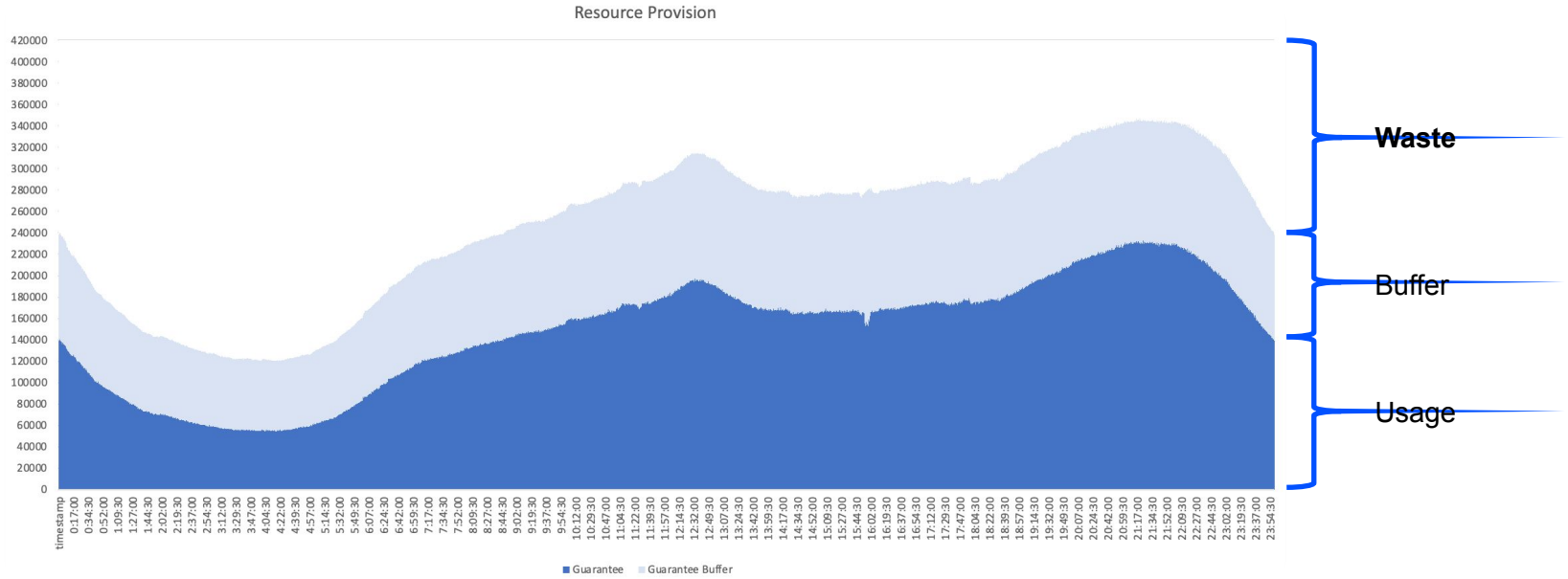
- QoS Resource Manager (QRM)
- Katalyst Agent
  - QRM Plugins
  - SysAdvisor
  - Resource Reporter
  - Eviction Manager
- Malachite

2 Application Scenarios

2.1 Colocation

Capacity Planning Challenges

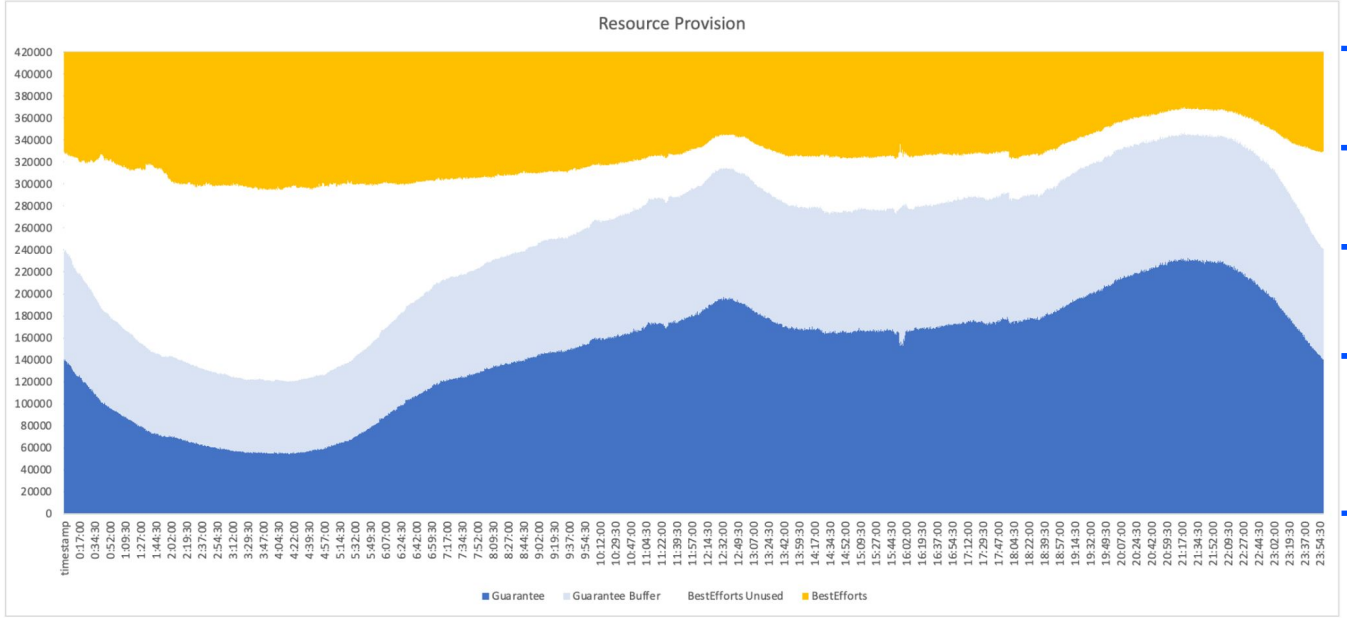
- The resource utilization of online services exhibits a tidal pattern, with very low utilization during the night
- Users tend to over-request resources to ensure service stability, leading to resource wastage



Colocation

The resource utilization patterns of online services and batch jobs are inherently complementary:

- Online services prioritize CPU and RPC latency
- Batch jobs prioritize memory and throughput

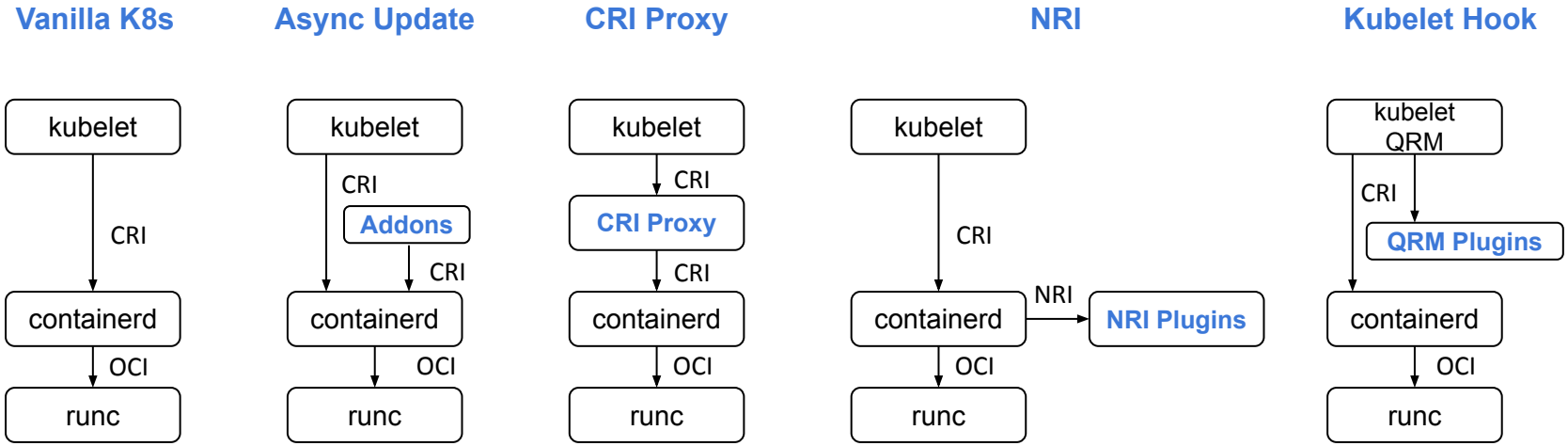


Extended QoS Classes

- **4 Extended QoS Classes**
  - Express services' requirements for resource quality
  - Naming based on CPU as the primary resource dimension
- **More QoS Enhancements**
  - NUMA binding
  - NUMA exclusive
  - Network class
  - …

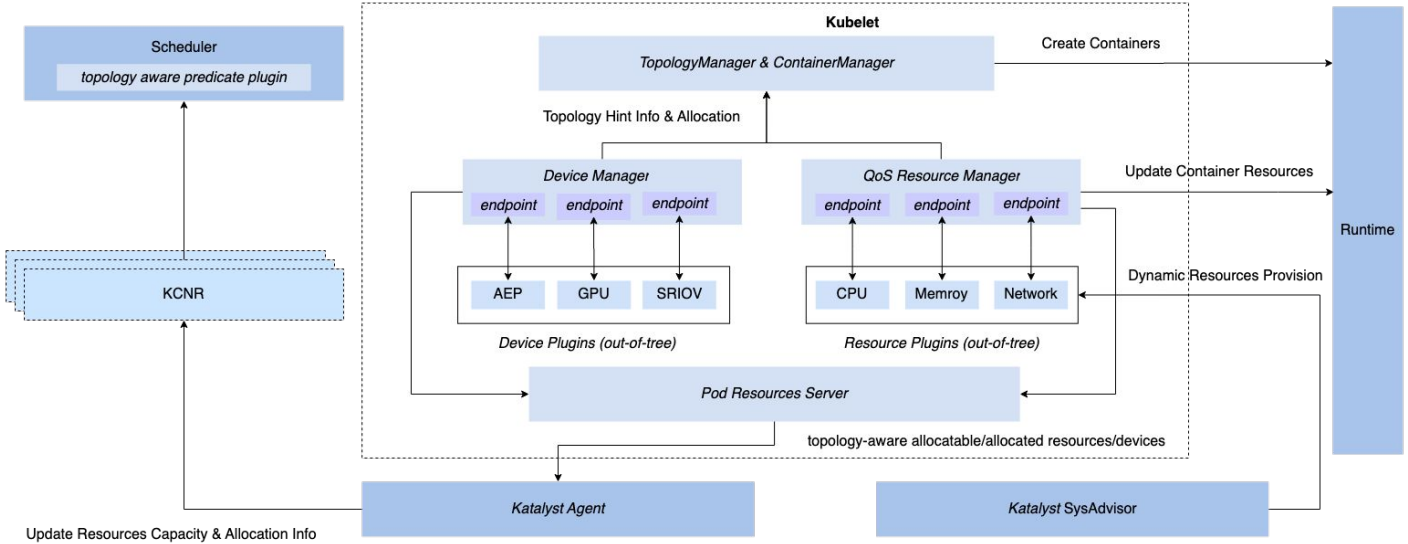
QoS Class	Attributes	Suitable for workload types	Relationship with K8s QoS
dedicated_cores	• Dedicated CPU cores, not shared with other workloads • Supports binding to NUMA nodes for improved performance	Extremely latency-sensitive workloads, such as ads, search, and recommendation	Guaranteed
shared_cores	• Shared CPU pool • Supports further dividing CPU pools based on business types	Workloads that can tolerate a certain degree of CPU throttling or interference, such as microservices	Guaranteed/ Burstable
reclaimed_cores	• Over-committed resources • Resource quality is relatively unguaranteed • May be evicted	Workloads that are not sensitive to latency and prioritize throughput, such as model training and batch jobs	BestEffort
system_cores	• Reserved CPU cores • Ensure the stability of system components	Critical system agents	Burstable

Plugin-Based Resource Management



Find the optimal hook point for extending resource management policies

QoS Resource Manager

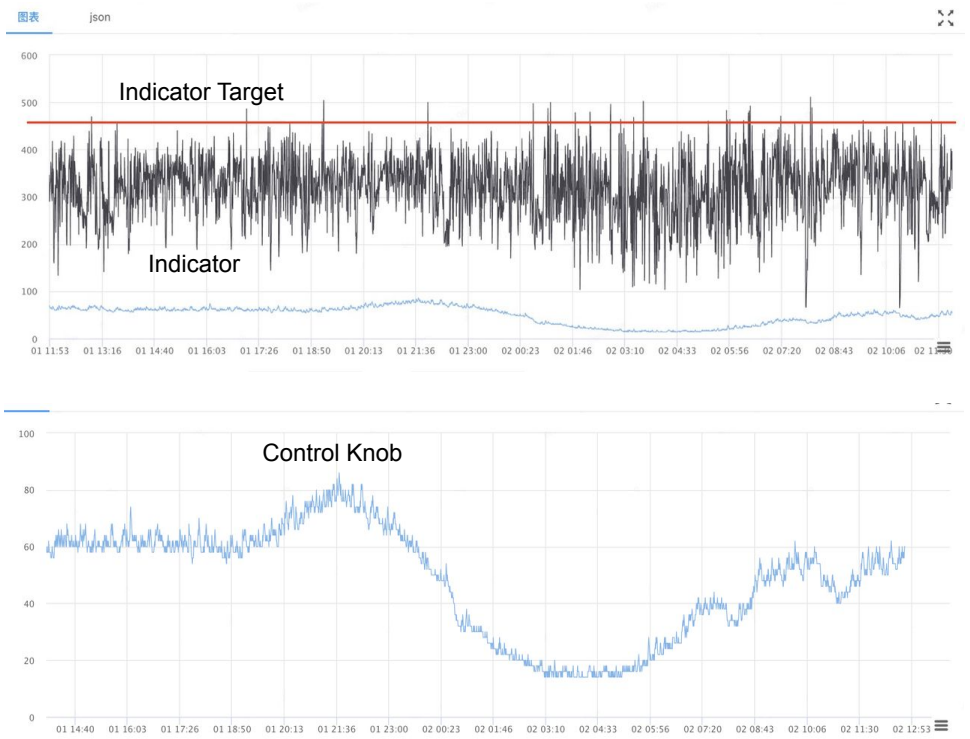


- **QoS Resource Manager**
  - Providing a registration mechanism for resource plugins
  - Registering as a hint provider for the topology manager
  - Periodically invoking the runtime to update container resource allocation
- **QoS Resource Plugin**
  - Customizing resource allocation policies for containers

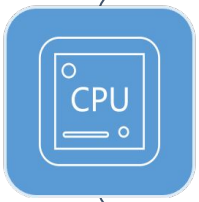
Service Profiling & Resource Prediction

PID Control Algorithm Based on Negative Feedback

- Analyzing the relationship between service business metrics and system metrics
- Continuously adjusting control knobs to make the current indicator value close to the sweet point



Multi-Dimensional Resource Isolation



CPUSet
Intel RDT
SCHED_IDLE

Per-memcg async reclaim
NUMA binding
Userspace advisor

Async IO for logging
Using different disks
IOCost

Using different NICs
net_cls
eBPF + EDT

Find the most suitable approach based on real business scenarios

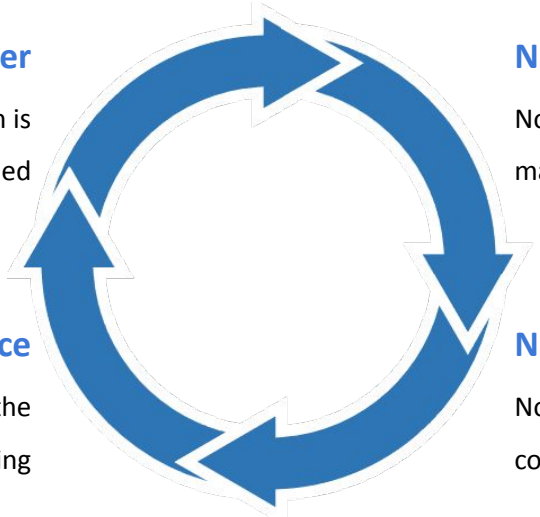
Multi-Tier Dynamic Configuration

Cluster

Default configurations, such as whether colocation is enabled

Node Pool

Node pool-level configurations, such as different resource management policies for CPU nodes and GPU nodes



Service

Service-level differentiated SLO configurations, such as the target indicator calculated from service profiling

Node

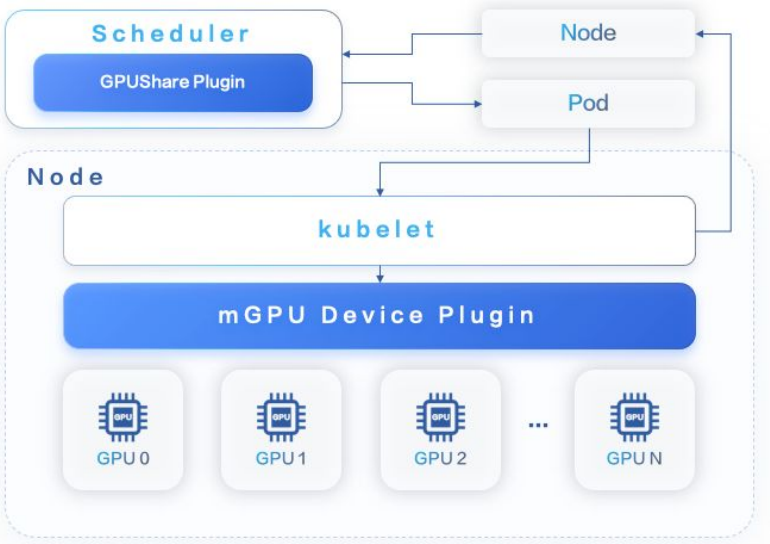
Node-level configurations, such as temporary configurations for debugging

2.2 GPU-Sharing Scheduling

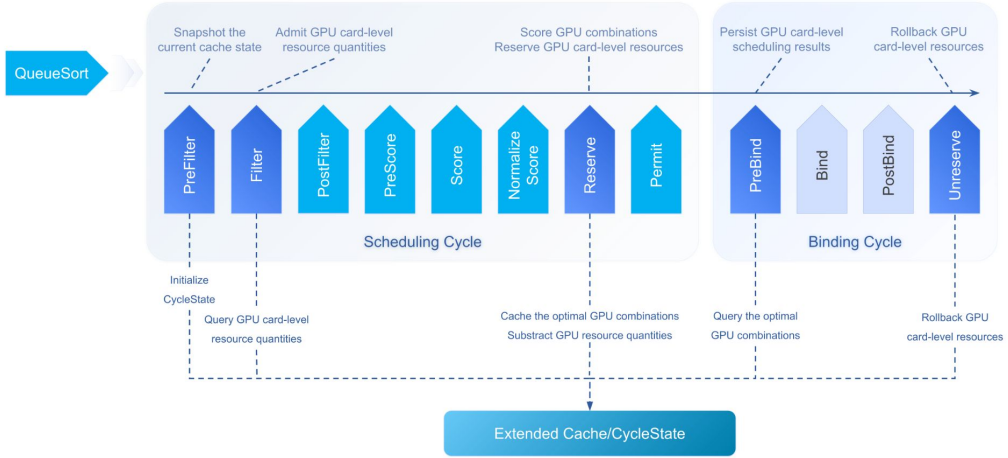
GPU-Sharing Scheduling

- Vanilla K8s only supports full GPU requests, which causes huge GPU waste in AI inference scenarios
- mGPU enables scheduling at a granularity of 1% for computing power and 1 MiB for GPU memory

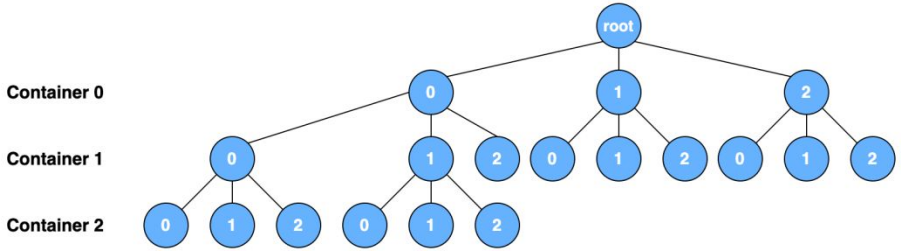
Overall Architecture



Scheduler Architecture



Scheduling Algorithm



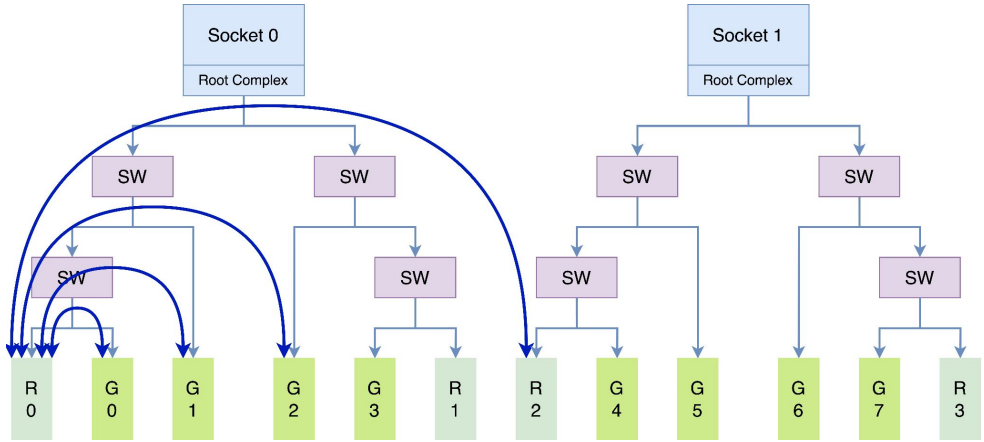
Upcoming session: https://sched.co/1Rj4O

2.3 Topology-Aware Scheduling

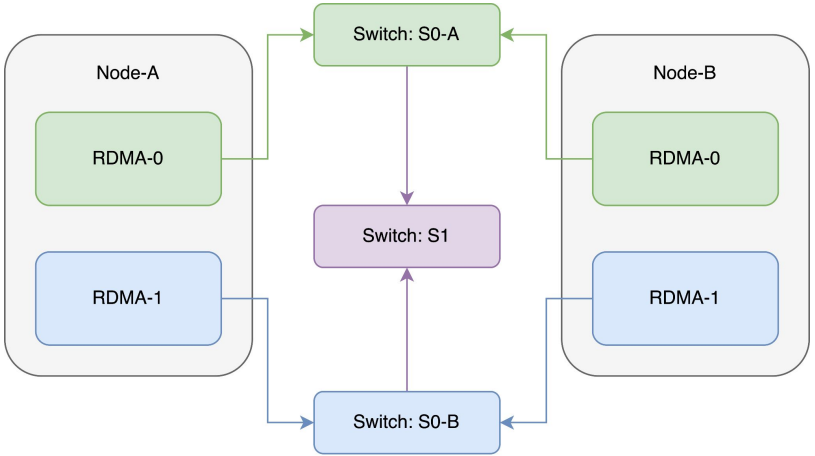
Topology-Aware Scheduling

- The K8s scheduler is not aware of the micro-topology on nodes, which can lead to a high number of admission failures
- The topology affinity strategy only takes into account NUMA topology

GPU-RDMA Affinity at the PCIe Switch Level



Inter-RDMA Affinity at the Switch Level



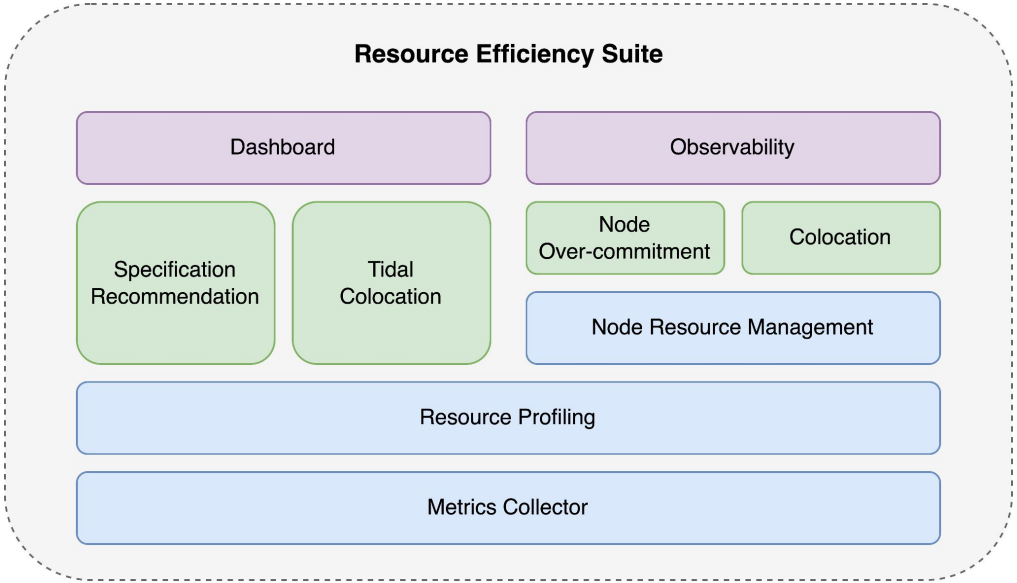
Upcoming session: https://sched.co/1Rj4O

2.4 Resource Efficiency Suite

Resource Efficiency Suite

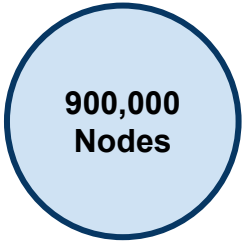
For cloud users, the threshold for using the colocation feature is relatively high.

- **Specification Recommendation**
  - Recreate
  - In-place update
- **Tidal Colocation**
  - HPA/CronHPA/Intelligent HPA
  - Node pool management
- **Node Over-commitment**
  - Allowing the scheduler to schedule more pods to a node without users' awareness
  - Interference detection and mitigation
  - Long short-term node resource prediction algorithm

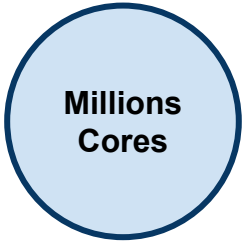


3 Results

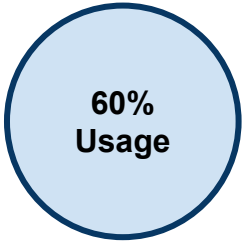
Results



More than 900,000 deployed nodes



Tens of millions of cores under management



Improved daily resource utilization from 23% to 60%

4 Community

Milestone

Version	Status	Date	Key Features
0.1	Released	Feb 27, 2023	• Colocation (MVP version)
0.2	Released	Jun 13, 2023	• Dedicated_cores with numa_binding (node-side) • NIC-NUMA affinity • Packet tagging • Eviction based on RSS overuse
0.3	Released	Aug 8, 2023	• Dynamic configuration • Service profiling based on PID control algorithm • Userspace memory management (drop cache, memory migration, memory limit, etc.)
0.4	Ongoing	End of Sept, 2023 (expected)	• Topology-aware scheduling • Specification recommendation • Tidal colocation • Node resource over-commitment • IOCost
…			• Dedicated_cores with numa_binding (scheduler-side) • OOM priority • …

Contact

- **Bi-weekly Community Meeting**
  - Thursday 19:30 GMT+8 (Asia/Shanghai)
  - Meeting notes and Agenda
- **Slack**
  - kubewharf.slack.com
  - Channel: katalyst
- **Community Lark Group**

- **He Cao**
  - Email: caohe.ch@bytedance.com
  - GitHub: @caohe
- **Wei Shao**
  - Email: shaowei.wayne@bytedance.com
  - GitHub: @waynepeking348
- **Upcoming Sessions**
  - https://sched.co/1Rj4O
  - https://sched.co/1Rj3f

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


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Thank you!