CLOUDERA

SCHEDULING FRAMEWORK: BATCH EXTENSIONS WITH APACHE YUNIKORN

Wilfred Spiegelenburg
Peter Bacsko

KubeCon | CloudNativeCon

Europe 2023

AGENDA

KubeCon CloudNativeCon Europe 2023

YuniKorn

Introduction YuniKorn

Plugin Architecture

Quotas: tracking and enforcement

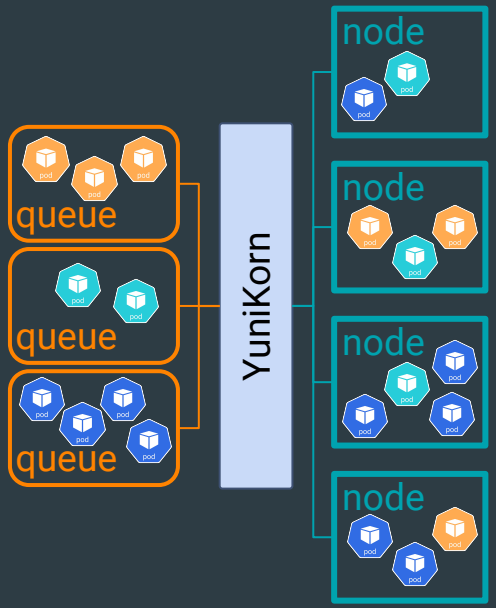
Preemption: queues and jobs

Demo

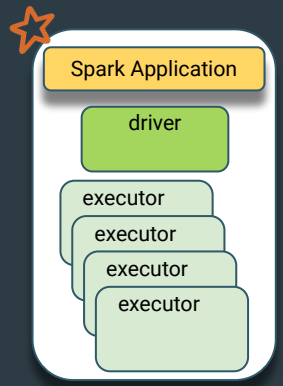
Q & A

WHY APACHE YUNIKORN

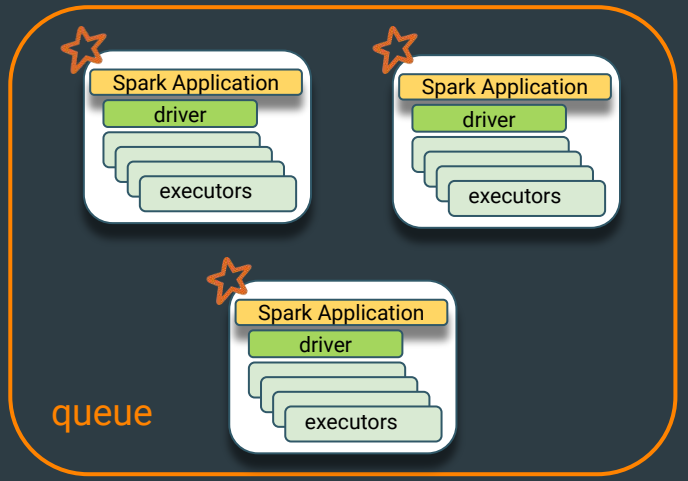
Advanced scheduling requirements



Workload Queueing



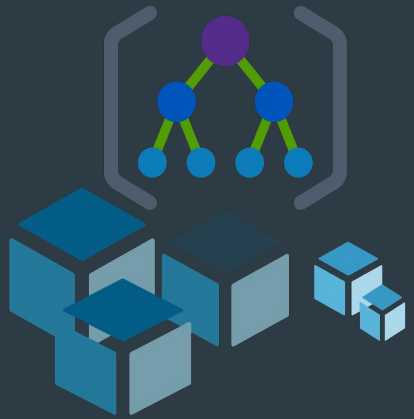
Gang Scheduling



Application Sorting

APACHE YUNIKORN

Advanced scheduling requirements



Schedules any Batch or Service workload
- K8s (job, daemon set, deployment etc)
- Spark, Tensor Flow, MPI, Ray etc

Simple integration:
- No code changes
- Annotations and labels only

Hierarchical queues
- Guaranteed resources
- Quotas
- Scheduling policies

AGENDA

KubeCon CloudNativeCon Europe 2023

YuniKorn

Introduction YuniKorn

Plugin Architecture

Quotas: tracking and enforcement

Preemption: queues and jobs

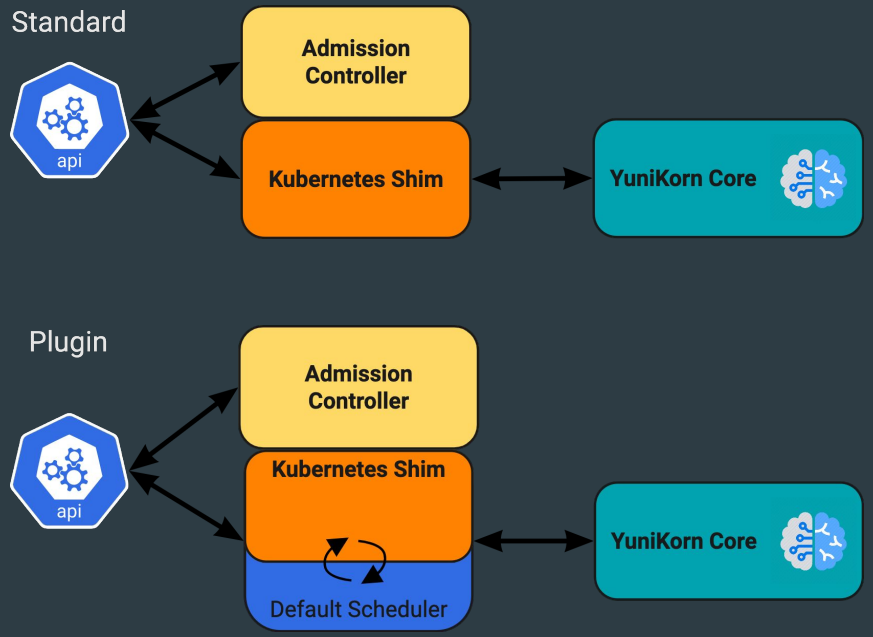
Demo

Q & A

ARCHITECTURE

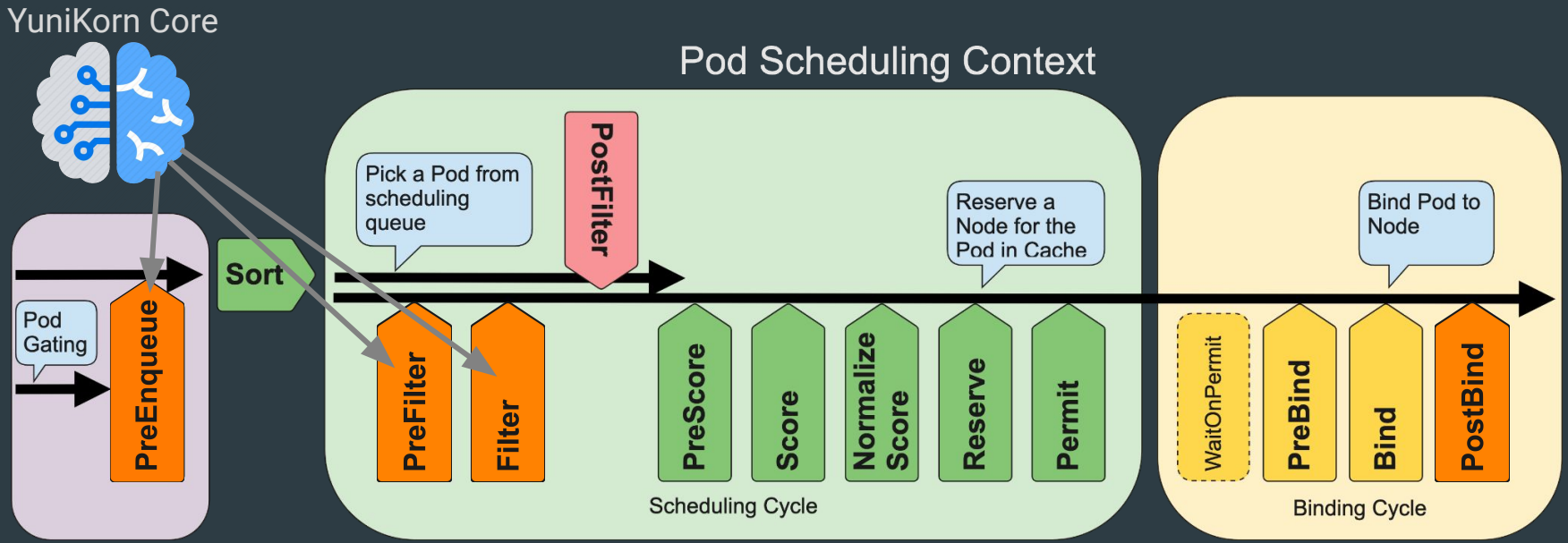
Deployment models

- STANDARD:
  - Custom scheduler
  - Replaces *default* scheduler
- PLUGIN:
  - Kubernetes Scheduling Framework (API)
  - Replace or augment limited functionality



PLUGIN ARCHITECTURE

Extension points and integration



AGENDA

KubeCon CloudNativeCon Europe 2023

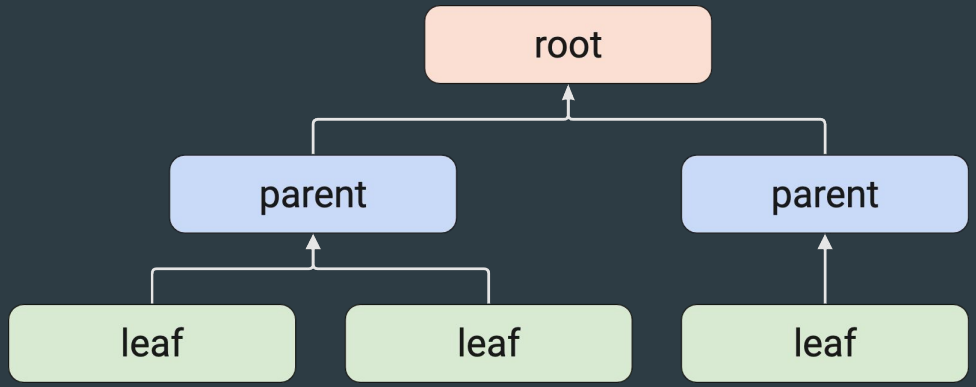
YuniKorn

- Introduction YuniKorn
- Plugin Architecture
- **Quotas: tracking and enforcement**
- Preemption: queues and jobs
- Demo
- Q & A

QUEUE HIERARCHY & QUOTAS

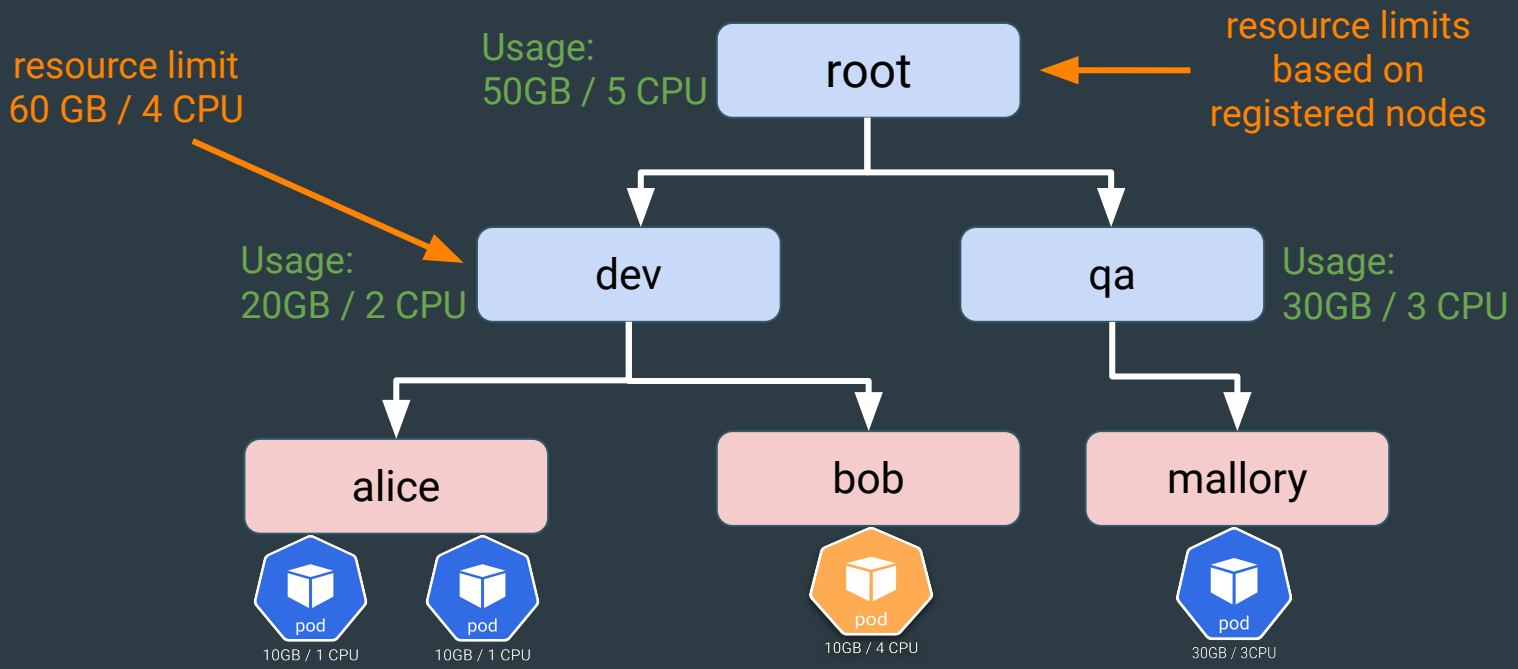
Hierarchical model

- Resources are managed via queues
  - Auto created queues
  - Static queues
- Resource located in leafs
  - propagate up to the root
  - Requests (pending)
  - Allocation (assigned to node)
- Quota
  - set per queue (also for users)
  - enforced at each level in the hierarchy



QUEUE HIERARCHY & QUOTAS

Example



NAMESPACE vs YUNIKORN QUEUE

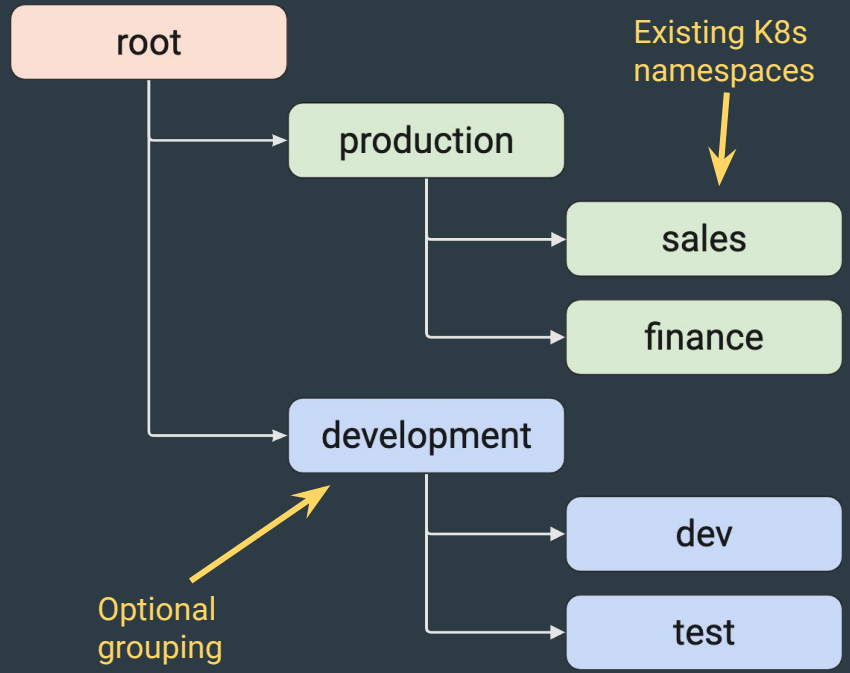
Comparison

K8s Namespace quotas:

- Pods are rejected if quota is exceeded (K8s feature)
- Requires an external retry logic

YuniKorn Queue quotas:

- Pods are queued not rejected
- Placement Rules for auto queue creation
- Namespace annotations
  - Queue quota
  - Parent queue



NAMESPACE BASED QUOTA CONFIGURATION

Scheduler config & annotation on the namespace

```
partitions:
  - name: default
    placementrules:
      - name: tag
        value: namespace
        create: true
        parent:
          name: tag
          value: namespace.parentqueue
    queues:
      - name: root
        queues:
          - name: production
            parent: true
          - name: development
            parent: true
```

Tells Yunikorn how-to auto create queues

Static queues

Annotations on the namespace object

“sales” namespace:

```
yunikorn.apache.org/namespace.quota:
  "{\"cpu\": \"64\", \"memory\": \"100G\"}"
yunikorn.apache.org/parentqueue:
  root.production
```

USER AND GROUP QUOTAS

Overview

- User and Group info lookup (YUNIKORN-1306)
  - Authenticated User info is only available inside the Admission Controller
  - Annotating the spec with the user info (Pod, Job, Deployment…)

- User and Group resource usage tracking (YUNIKORN-984)
  - Tracking per user & group at any point in the queue hierarchy
  - Exposed only via REST API

- User and Group Quota enforcement (YUNIKORN-1573)
  - Development in progress
  - Targeted for 1.3.0

AGENDA

KubeCon CloudNativeCon Europe 2023

YuniKorn

Introduction YuniKorn

Plugin Architecture

Quotas: tracking and enforcement

Preemption: queues and jobs

Demo

Q & A

Preemption

Configuration

- Priority class
  - YuniKorn annotation
- Queue configuration
  - Max resource (quota)
  - Guaranteed resource (preemption base)
- Multi tenancy
  - Preemption Fence
  - Priority Fence
  - Priority leveling (offset)

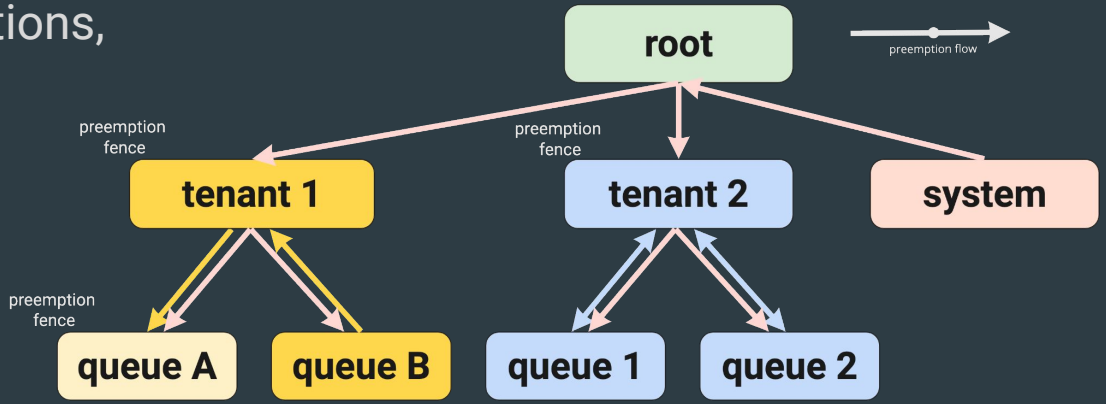
```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
  annotations:
    yunikorn.apache.org/allow-preemption: "true"
value: 1000
globalDefault: false
```

```
queues:
  - name: high
    parent: false
    properties:
      preemption.policy: fence
      preemption.delay: 30s
      priority.policy: "fence"
      priority.offset: "1000"
    resources:
      guaranteed:
        {memory: 24Gi, vcore: 6}
      max:
        {memory: 32Gi, vcore: 8}
```

Preemption

How does it work?

- 7 Laws of preemption
- Guaranteed resources are the base for queues
- Policies are strong suggestions,
  NOT guarantees
- Multi tenancy
  - Fencing
  - Priority offset



[SLA Aware Batch Scheduling in Apache YuniKorn with Multi tenant preemption]

AGENDA

KubeCon CloudNativeCon Europe 2023

YuniKorn

Introduction YuniKorn

Plugin Architecture

Quotas: tracking and enforcement

Preemption: queues and jobs

Demo

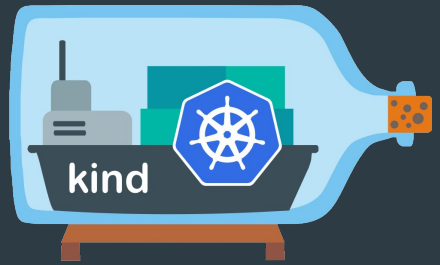
Q & A

DEMO

Prepared cluster

- Showing two pieces of functionality
  - Quotas
  - Preemption

- Kind cluster (1.26.3)
  - Plugin version deployed
  - 3 nodes (control-plane + 2 workers)
  - Priority classes with YuniKorn annotation
  - Hierarchical queues, leafs defined:
    - Same quota (maximum usage)
    - Different guaranteed resources



DEMO

Workload description & behaviour

Two applications, one for each leaf queue

- Application 1:
  - user: peter, group: developers
  - queue: root.low
  - 10 pods, low priority
- Application 2:
  - user: wilfred, group: developers
  - queue: root.high
  - 8 pods, high priority

- Submit Application 1:
  - Queue and user usage
  - Only 8 pods start
  - 2 pods remain pending
- Submit Application 2:
  - Rebalancing queues
  - Only Guaranteed usage
  - Group usage: both applications

AGENDA

KubeCon CloudNativeCon Europe 2023

YuniKorn

Introduction YuniKorn

Plugin Architecture

Quotas: tracking and enforcement

Preemption: queues and jobs

Demo

Q & A


KubeCon | CloudNativeCon

Europe 2023

Website: https://yunikorn.apache.org

Email: dev@yunikorn.apache.org

Slack: YuniKorn Slack

THANK YOU

CLOUDERA