
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

KubeCon CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

Class Resources: Kubernetes' Fastest Way Of Shushing Noisy Neighbors

Markus Lehtonen
Peter Hunt

Not all workloads should be equal...

QoS: Quality of service in Kubernetes

QoS classes specify CPU and Memory limits and requests

CPU management policies further customize behavior

Many more resources on the node to guarantee...

Mission

Improve the Quality-of-Service of applications

Enable controls that don’t fit into current K8s resource model

Cache
Memory bandwidth
Disk I/O

Add a new fundamental resource type to K8s

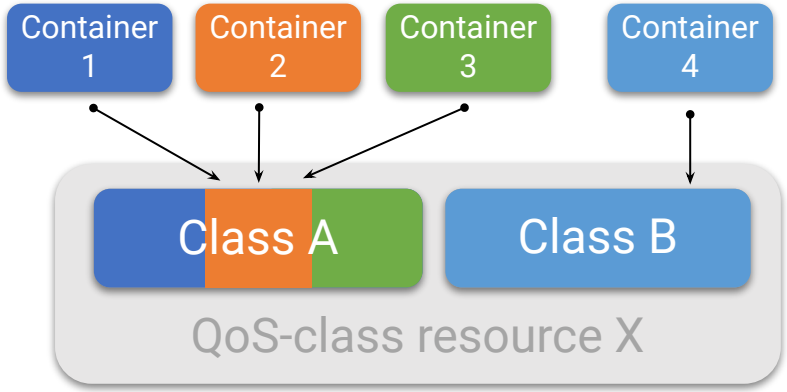
Properties of QoS-class resources

Request class identifier instead of amount of capacity

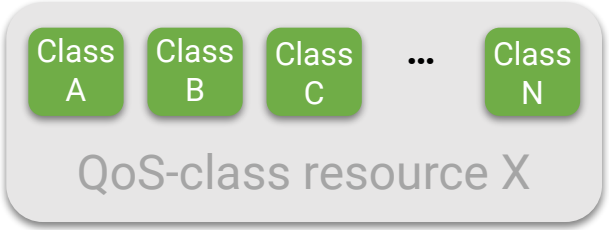
Container 1

```
resources:
  classes:
    qos-resource-x: “class-a”
  requests:
    cpu: 500m
    vendor/xpu: 1
  limits:
    cpu: 1000m
    vendor/xpu: 1
```

Multiple containers can be assigned to the same class



Enumerable set of classes

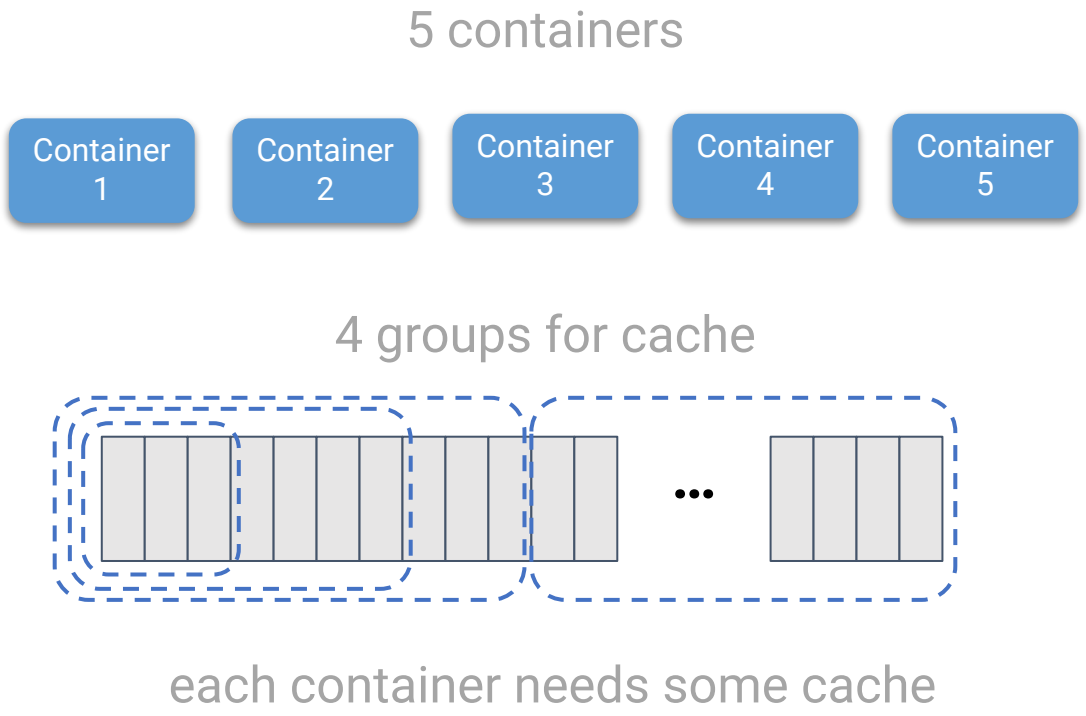


Example: cache allocation

Linux resctrlfs interface is inherently class-based

Hide HW details from user

Cache can only be split into M groups but we have N applications. M<N



Example: blkio – better UX

Not feasible to specify detailed HW-dependent parameters on Pod level

Different nodes might have totally different HW

Throttling parameters (blkio) are per-device

sysfs:

```
$ cat blkio.throttle.write_bps_device
8:0 10000000
8:16 200000000
8:32 10000000
8:48 300000000
```

OCI runtime-spec:

```
"blockIO": {
    "throttleWriteIOPSDevice": [
        {
            "major": 8,
            "minor": 0,
            "rate": 10000000
        },
        {
            "major": 8,
            "minor": 16,
            "rate": 200000000
        },
        ...
```

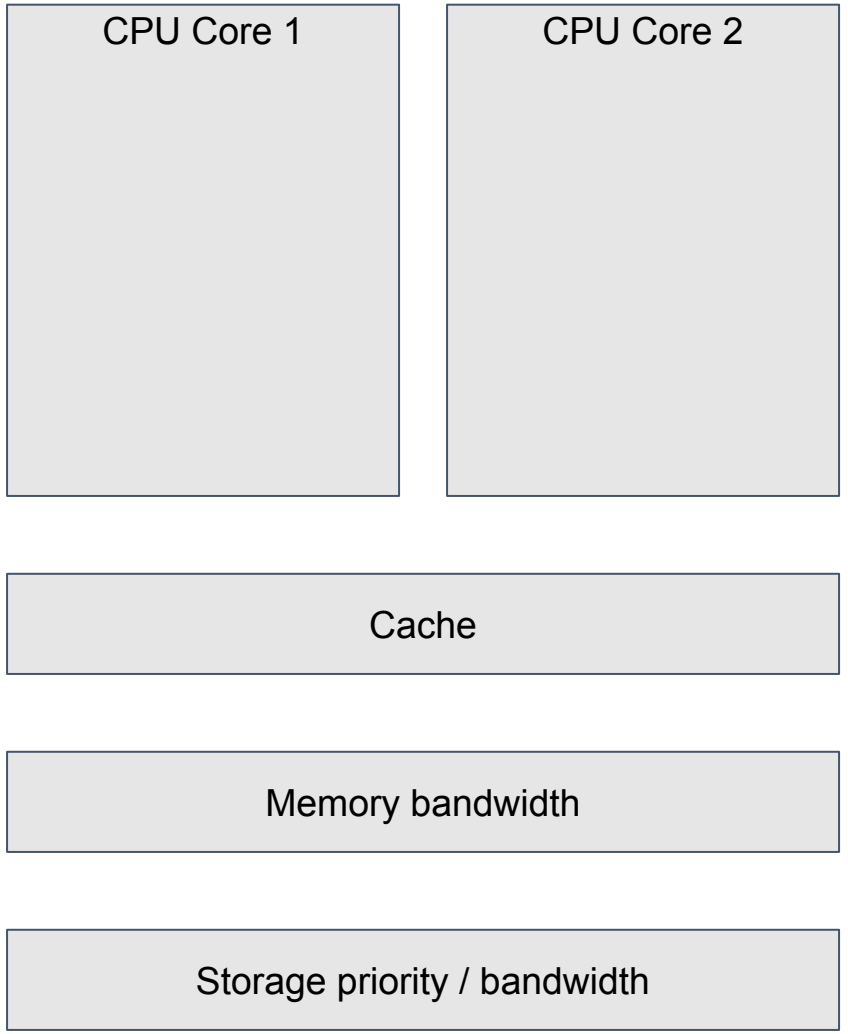
Tuning workloads now

Workloads to be scheduled

Emergency Alarm

Rock Band Site

Kubernetes cluster node



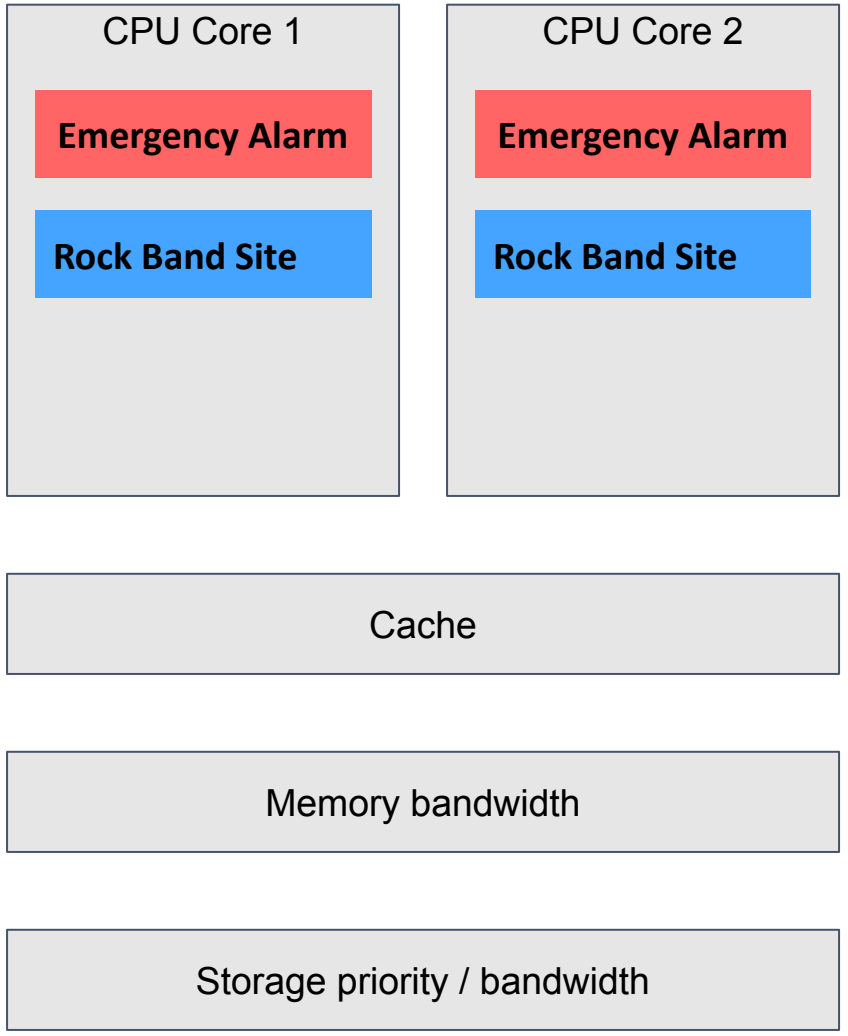
Configuration

```
Defaults
```

Tuning workloads now

Workloads to be scheduled

Kubernetes cluster node



Configuration

`Defaults`

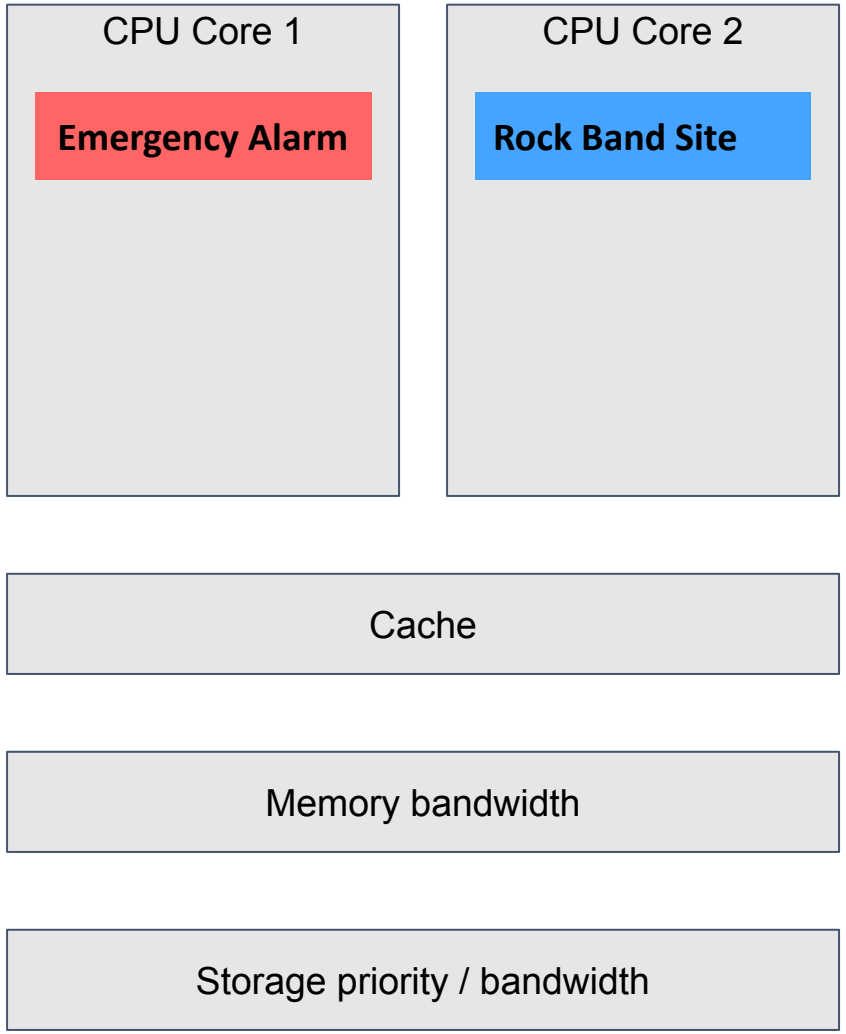
Tuning workloads now

KubeCon CloudNativeCon North America 2022

Tuning performance

- Exclusive CPUs
  kubelet CPU manager

Kubernetes cluster node



Configuration

```
kubelet:
  --cpu-manager-policy=static

emergency.yaml:
  resources:
    limits:
      cpu: 1000m
      memory: 1G
    requests:
      cpu: 1000m
      memory: 1G
```

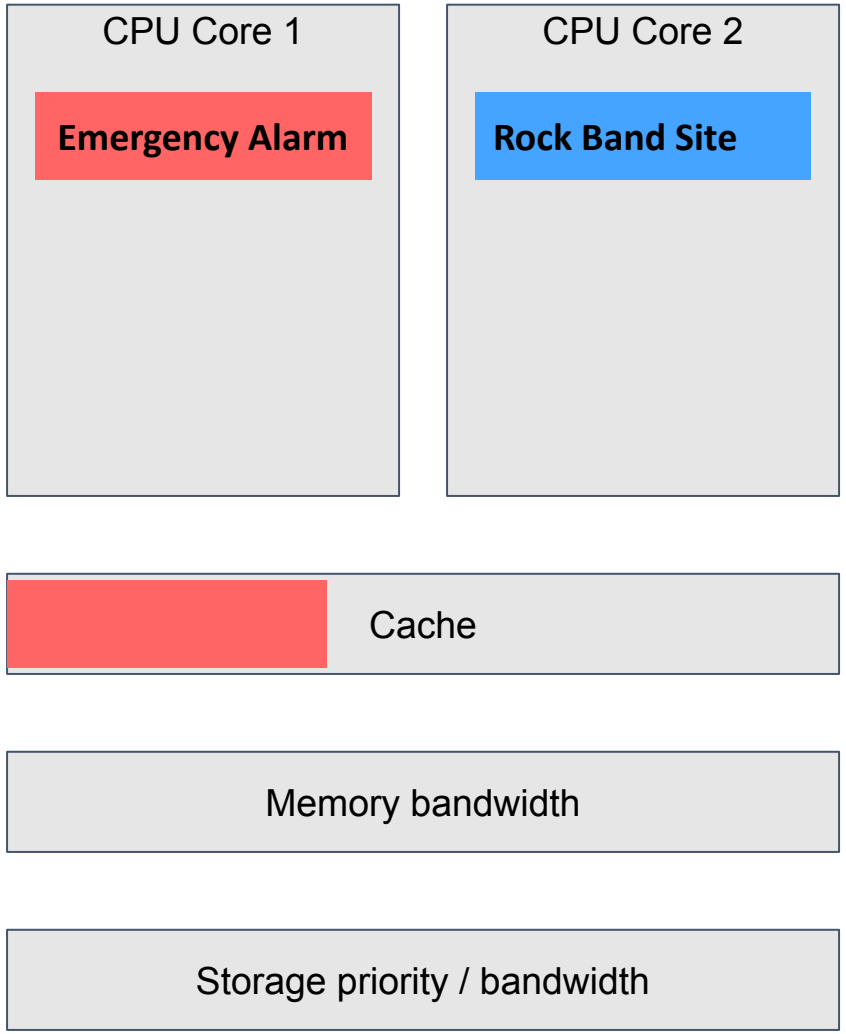
Tuning workloads in the future

KubeCon CloudNativeCon North America 2022

Tuning performance

- Exclusive CPUs
  kubelet CPU manager
- Exclusive cache
  container runtime + RDT

Kubernetes cluster node



Configuration

```
crio:
  --rdt-config-file rdt.cfg

rdt.cfg:
  classes:
    red:
      l2Allocation: ...
      l3Allocation: ...

emergency.yaml:
  classes:
    rdt: red
```

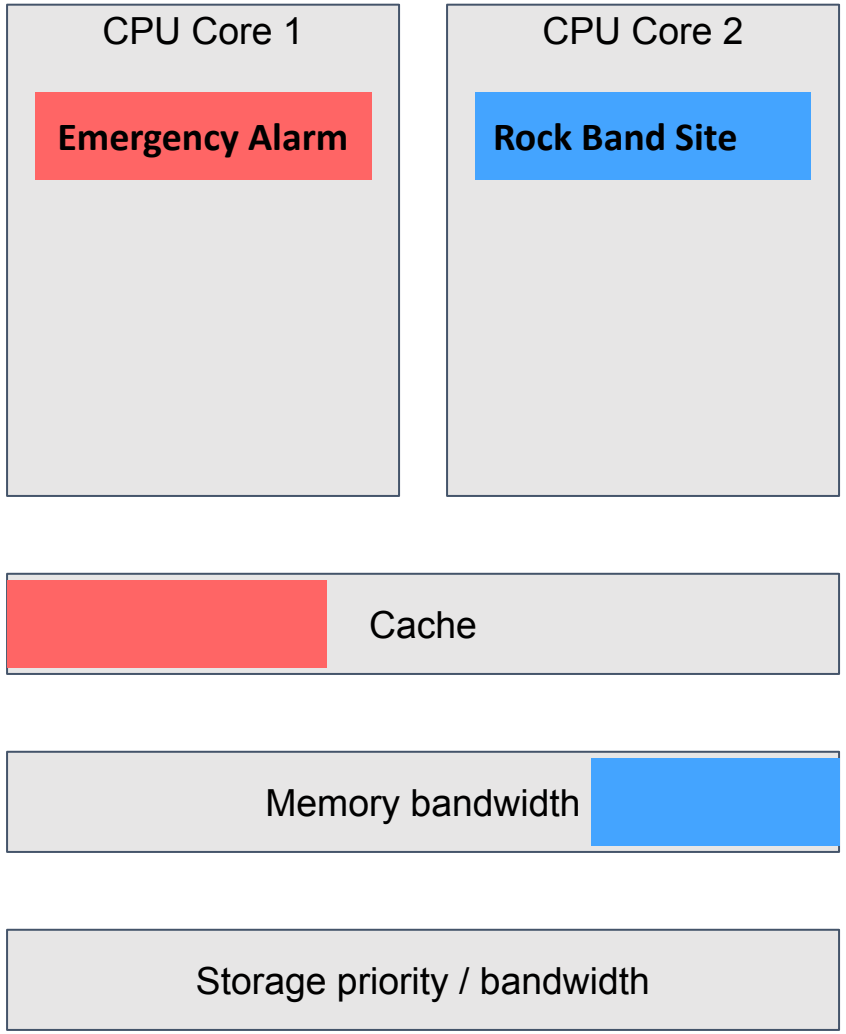
Tuning workloads in the future

KubeCon CloudNativeCon North America 2022

Tuning performance

- Exclusive CPUs
  kubelet CPU manager
- Exclusive cache
  container runtime + RDT
- Mem bandwidth throttling
  container runtime + RDT

Kubernetes cluster node



Configuration

```
crio:
  --rdt-config-file rdt.cfg

rdt.cfg:
  classes:
    blue:
      mbAllocation: ...

rock-site.yaml:
  classes:
    rdt: blue
```

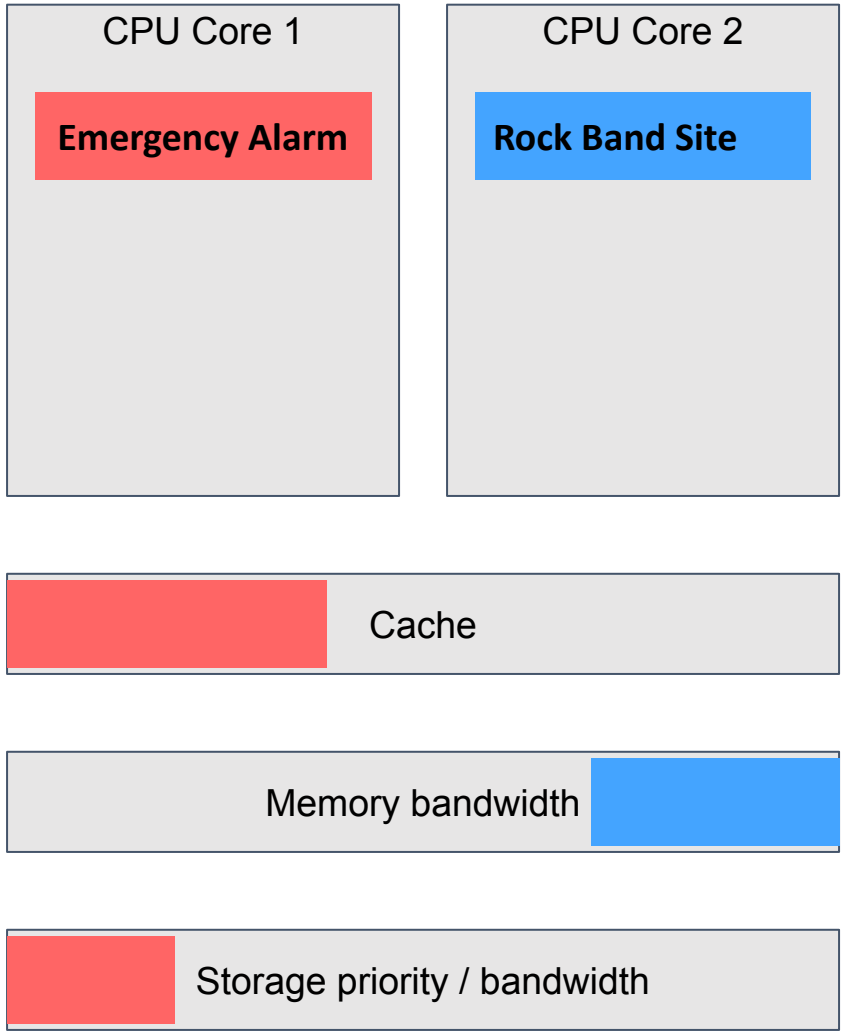
Tuning workloads in the future

KubeCon CloudNativeCon North America 2022

Tuning performance

- Exclusive CPUs
  kubelet CPU manager
- Exclusive cache
  container runtime + RDT
- Mem bandwidth throttling
  container runtime + RDT
- Block I/O priority
  container runtime + blockio

Kubernetes cluster node



Configuration

```
crio:
  --blockio-config-file blockio.cfg

blockio.cfg:
  classes:
    red:
      weight: 400

emergency.yaml:
  classes:
    blockio: red
```

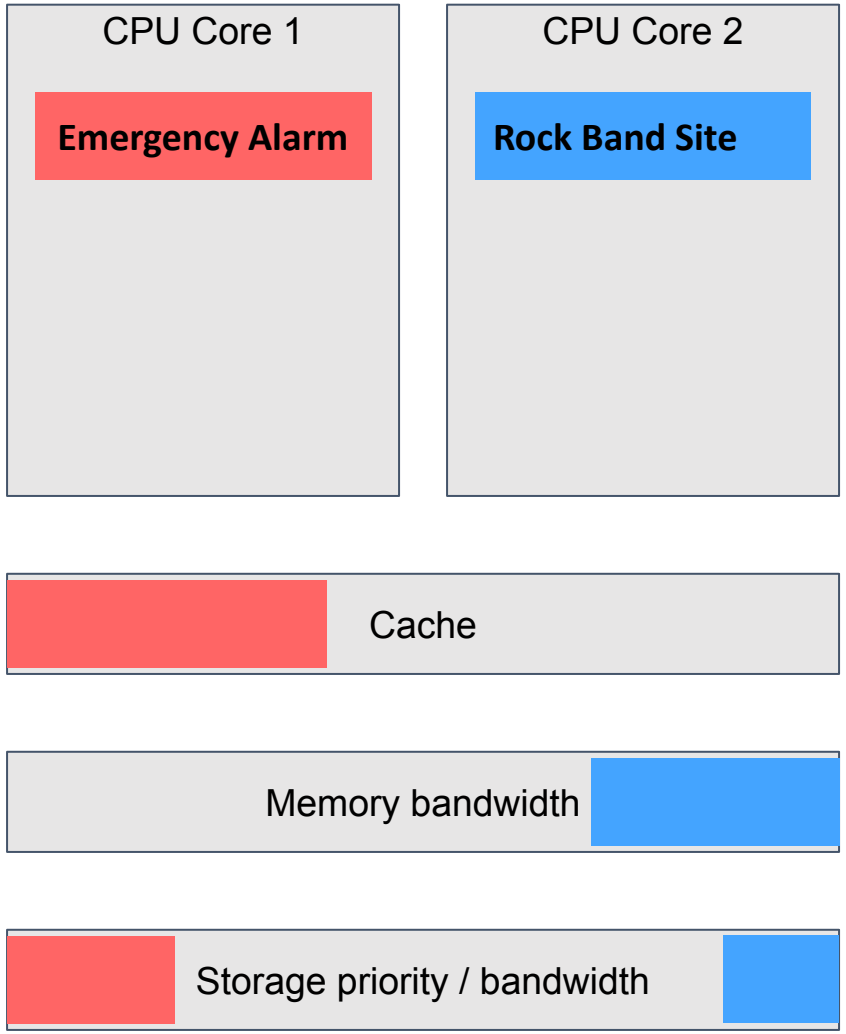
Tuning workloads in the future

KubeCon CloudNativeCon North America 2022

Tuning performance

- Exclusive CPUs
  kubelet CPU manager
- Exclusive cache
  container runtime + RDT
- Mem bandwidth throttling
  container runtime + RDT
- Block I/O priority
  container runtime + blockio
- Block I/O throttling
  container runtime + blockio

Kubernetes cluster node



Configuration

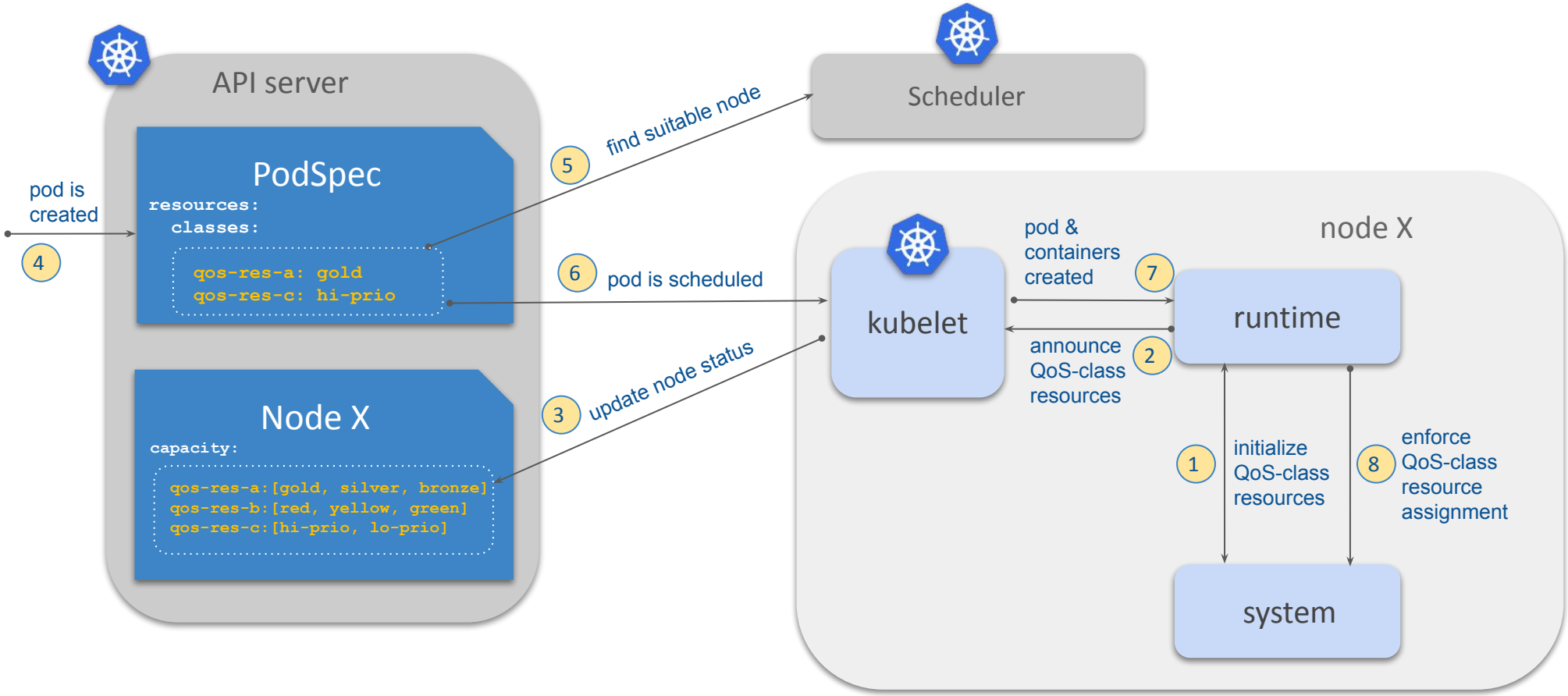
```
crio:
  --blockio-config-file blockio.cfg

blockio.cfg:
  classes:
    blue:
      throttleReadBps: 60M
      throttleWriteBps: 40M

rock-site.yaml:
  classes:
    blockio: blue
```

Finally, some peace

KEP – Proposed design



KEP – Scope

- CRI API
  - QoS-class resource assignment and discovery
  - in-place updates of running containers
- Kubernetes API
  - PodSpec (QoS-class resource requests)
  - NodeStatus (QoS-class resource availability)
    - visibility for users
    - kube-scheduler
  - access control (ResourceQuota)

KEP – Opaque to Kubernetes

Configuration and management of QoS-class resources handled by container runtime

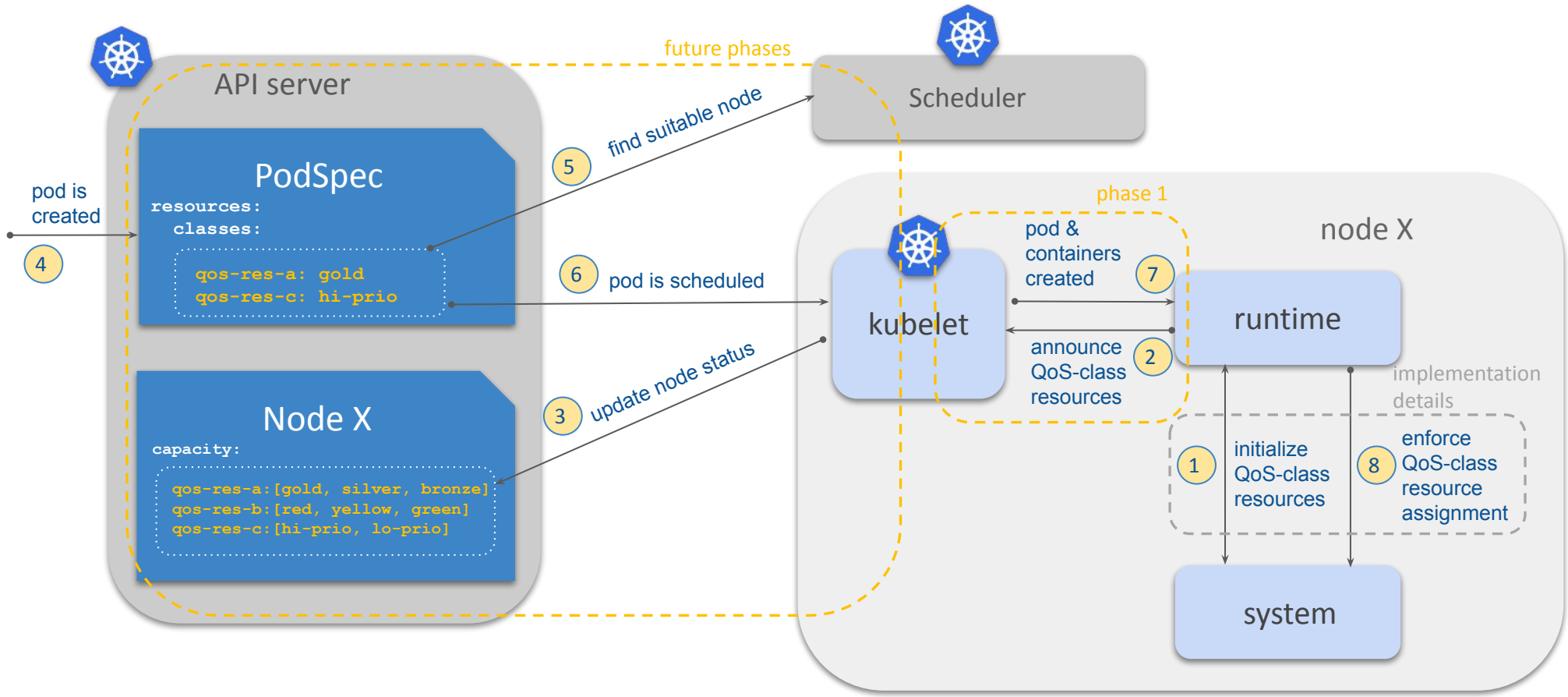
K8s knows what QoS-class resources are available and where

K8s does not (need to) know about implementation details of QoS-class resources

Vendor-specific QoS-class resources

Generalized mechanism – enables simple implementation of new types of QoS-class resources

KEP – Implementation phases



What is currently available

- runtimes (cri-o, containerd) have rudimentary support (Linux)
  - resctrlfs (cache and mem bw)
  - blockio (blkio cgroup controller)
- runtimes manage everything
- pod annotations for UI

crio.conf:

```
# Configuration for resctrl pseudo-fs
rdt_config_file = "rdt-conf.yaml"

# Configuration for the blkio controller
blockio_config_file = "blockio-conf.yaml"
```

rdt-conf.yaml:

```
partitions:
  default:
    classes:
      gold:
        l3Allocation: 100%
      silver:
        l3Allocation: 66%
      bronze:
        l3Allocation: 33%
```

example-pod.yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: example
  annotations:
    rdt.resources.beta.kubernetes.io/container.cnt: bronze
spec:
  containers:
  - name: cnt
...
```

Drawbacks of runtime-only approach

- not Kubernetes
  - documentation missing
  - only for early adopters
- bad UX
  - no visibility what is available
  - no scheduler support
  - “you just need to know”

With QoS-class resources implemented

crio.conf:

```
# Configuration for resctrl pseudo-fs
rdt_config_file = "/etc/rdt-conf.yaml"

# Configuration for the blkio controller
blockio_config_file = "/etc/blkio-conf.yaml"
```

/etc/rdt-conf.yaml:

```
partitions:
  default:
    classes:
      gold:
        l3Allocation: 100%
      silver:
        l3Allocation: 66%
      bronze:
        l3Allocation: 33%
```

example-pod.yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - name: data-pump
    image: my-dp-image:latest
    resources:
      classes:
        rdt: gold
  - name: log-handler
    image: my-lh-image:latest
    resources:
      classes:
        rdt: bronze
```

node.status:

```
$ kubectl describe no
Name:               node-x
...
Class Resources (Container):
  Name     Classes
  ----     -------
  blockio  high-prio, low-prio, normal
  rdt      bronze, silver, gold
...
```

Demo


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: qos-class-resource-demo
spec:
  containers:
  - name: simple
    image: k8s.gcr.io/pause
    resources:
      classes:
        rdt: bronze
        blockio: low-prio
```

KEP – Status

- KEP under review
- Implementation phase 1 (alpha) targeting v1.27
- Future implementation phases v1.28+

- Open concerns:
  - usage of annotations in phase 1 (vs. K8s API)
  - some API details

Future work

Specify Pod Qos explicitly in PodSpec

```
apiVersion: v1
kind: Pod
metadata:
  name: burstable
spec:
  resources:
    classes:
      qos: burstable
  containers:
  - name: cnt-1
    image: k8s.gcr.io/pause
    resources:
      requests:
        cpu: 1
      limits:
        cpu: 1
```

Implement new types of QoS-class resources in runtimes

```
apiVersion: v1
kind: Pod
metadata:
  name: qos-ng
spec:
  containers:
  - name: hbm
    image: k8s.gcr.io/pause
    resources:
      classes:
        memory: high-bw
        swap: no-swap
```

Get involved

KEP #3008

https://github.com/kubernetes/enhancements/issues/3008


KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
Thank you!
TiKV
TiKV