

# CLOUD NATIVE AI +

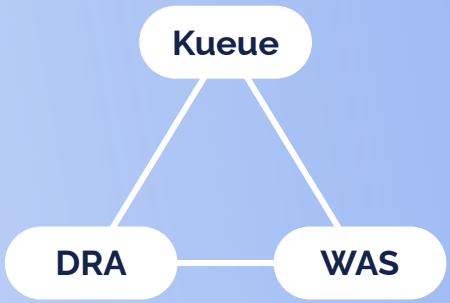
# INFERENCE DAY

## NORTH AMERICA

CLOUD NATIVE AI +
INFERENCE DAY
NORTH AMERICA

# Kueue, DRA, and WAS: The Kubernetes Scheduling Triangle

*Michał Woźniak, Google*
*Sohan Kunkerkar, Red Hat*



# Problem statement



Orchestrate AI training and inference across teams that compete for scarce accelerators, **while enforcing organizational policy.**

## 1 Fair sharing

Share scarce capacity across teams by policy: quota, borrowing, preemption.



## 2 Gang scheduling

Start a distributed job with all of its Pods, or not at all.



## 3 Topology-aware scheduling

Place Pods that talk a lot close together on the accelerator fabric.



# What is Kueue?





## Workload orchestrator

queueing and scheduling

- Decides **when** a workload starts or stops
- Decides **where** it runs: flavor, topology
- Controls **how** it runs: all or nothing



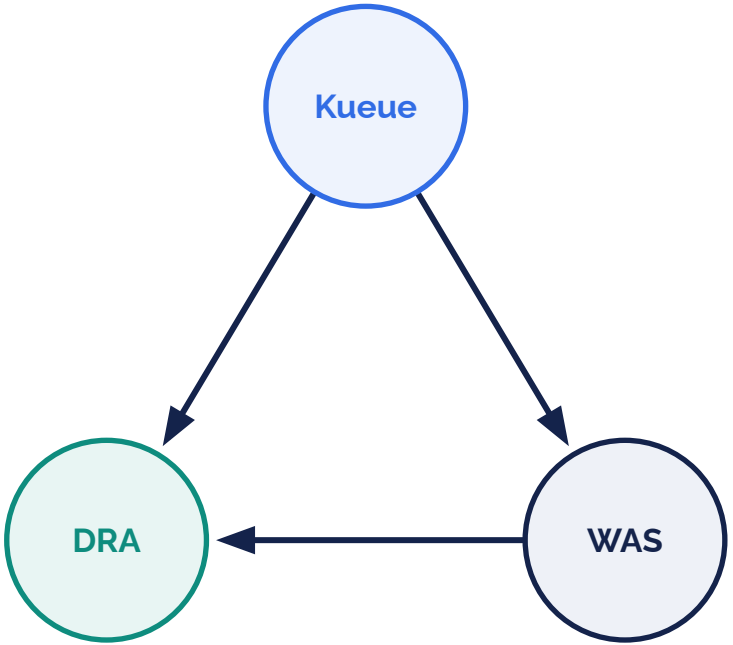
## Quota manager

multi-tenant

- Quota per team: **ClusterQueues, cohorts**
- **Borrowing** and **preemption** between teams
- Quota for **any hardware**, even DRA devices

# The scheduling triangle

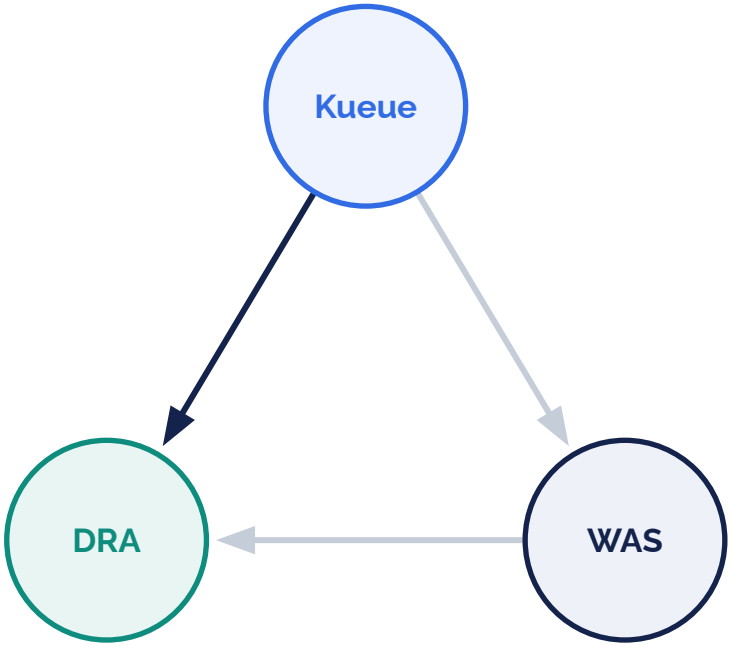





**Kueue**
who runs, and when

**WAS** · Workload-Aware Scheduling
where the group runs

**DRA** · Dynamic Resource Allocation
which devices




1

# DRA integration deep dive

**Kueue → DRA** · Sohan Kunkerkar

# What is DRA?

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

## Device plugins

nvidia.com/gpu: 2



Asks for a count of one named resource

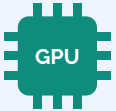
## Dynamic Resource Allocation




ResourceClaim: one GPU with ≥ 32 GiB memory

### DRA brings to core Kubernetes

- ✓ **MIG slices and shared GPUs, per claim**
- ✓ **Pick by attributes, with fallbacks**
- ✓ **GPU and NIC aligned**
- ✓ **One claim shared by a group**




DRA decides **which devices** a Pod gets;
**Kueue** decides **who** gets them, and **when**

# Why integrate Kueue with DRA?

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

**Core quota has no queues**
it counts devices, but can't borrow, share fairly or preempt

→ **Quota for devices, even slices**
charged at admission, by count or by memory

**Admitted, but no node fits**
the Pods stay Pending and hold quota

→ **Device check before admission**
kube-scheduler's DRA allocator, node by node (opt-in)

**Groups need device state**
one Pod at a time can't track or free devices

→ **Track devices across the group**
planned: WAS + kube-scheduler's DRA plugin

# DRA in Kueue: quota

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

## Whole devices

**ResourceClaimTemplate**
2 devices of class gpu.example.com

**deviceClassMappings**
gpu.example.com → quota resource gpus

**ClusterQueue quota · gpus**




2 of 4

## MIG slices

**Four claims · 1g.10gb slices**
each slice is one device, ~9.6 GiB

**By device count · quota: 4 devices**




4 of 4: full

**By memory · quota: 4 × 80 GiB**




12%

**Borrowing, preemption and fair sharing use device quota too**

# Shared GPUs and fallbacks

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

Alpha

## Shared GPU capacity

one 80 GiB GPU, three claims




Kueue charges the capacity each claim requests

charged
**50 GiB**

Alpha

## Prioritized list

**first** fast GPU × 1

**else** GPU × 1

→ charged **1 GPU**

Charged once: every alternative needs the same count and the same quota resource

**Kueue charges what you declare, not what runs**

# Quota fits. Every node fails.

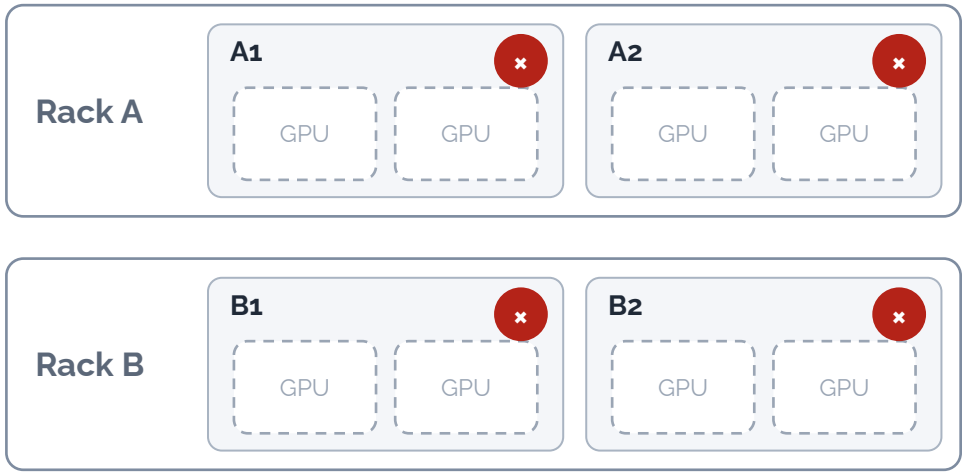
CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

1 Pod GPU GPU GPU

ClusterQueue quota · gpus

GPU GPU GPU GPU 3 of 4

✓ quota admits




Admitted anyway: a Pending Pod holds 3 GPUs of quota

# Node feasibility: two checks per node

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

Kueue-native
**TAS**

Before admission, it picks the rack, then a node for each Pod

**Can this Pod run on this node?**

one Pod, one node

yes / no

Alpha · opt-in

**kube-scheduler filters**

via scheduler-library

Alpha · opt-in

**DRA device check**

kube-scheduler's DRA allocator called directly

**All four nodes say no: it stays queued, and its 3 GPUs of quota stay free**

# Gap 1: one yes can't serve four Pods



**Inference: 4 Pods × (1 GPU + 2 CPU), one rack**

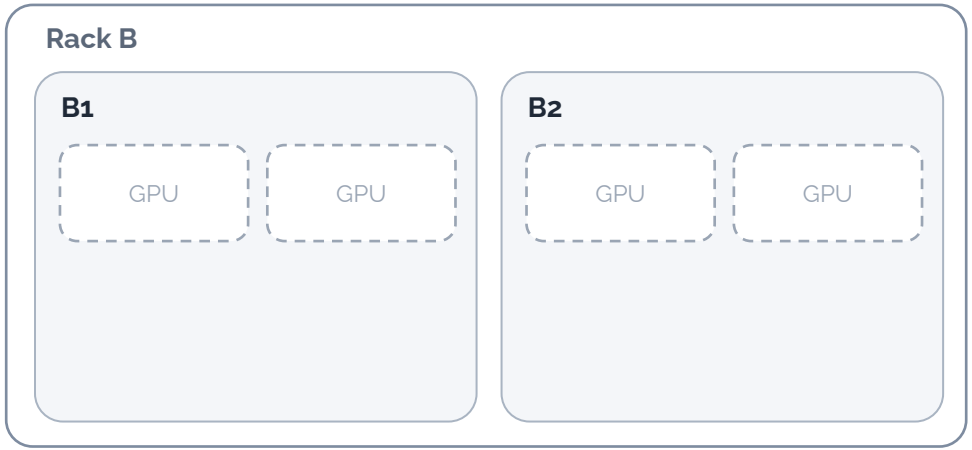
**Rack B: 2 nodes × (2 GPUs + 8 CPU), all free**

## Per-node device check

**B1**
2 free GPUs → **1 Pod?** → ✓ yes

**B2**
2 free GPUs → **1 Pod?** → ✓ yes

**same device state for every Pod**



# Gap 1: one yes can't serve four Pods

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

**Inference: 4 Pods × (1 GPU + 2 CPU), one rack**

**Rack B: 2 nodes × (2 GPUs + 8 CPU), all free**

## Per-node device check

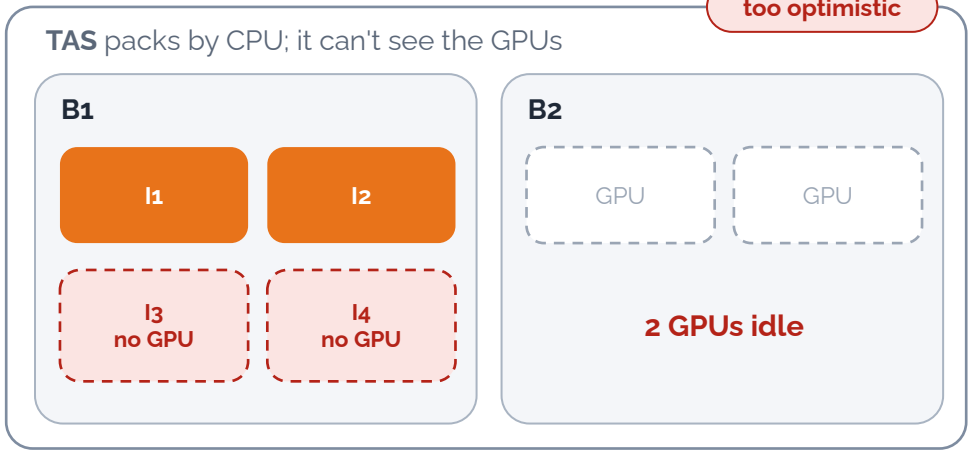
**B1**
2 free GPUs → **1 Pod?** → ✓ yes

**B2**
2 free GPUs → **1 Pod?** → ✓ yes

**same device state for every Pod**

too optimistic

**TAS** packs by CPU; it can't see the GPUs




**Missing: device state that changes as each Pod is placed**

# Gap 2: the simulation frees CPUs, not GPUs



B1: training job, 2 GPUs + 8 CPU, low priority

Incoming: 4 Pods × (1 GPU + 4 CPU) · max 2 per node

1 CPU simulation

B1

training job: removed

2 Pods

B2

2 Pods

CPU view: all 4 Pods fit ✓

# Gap 2: the simulation frees CPUs, not GPUs

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

B1: training job, 2 GPUs + 8 CPU, low priority

Incoming: 4 Pods × (1 GPU + 4 CPU) · max 2 per node

## 1 CPU simulation

**B1**

training job: removed

2 Pods

**B2**

2 Pods

CPU view: all 4 Pods fit ✓

too restrictive

## 2 DRA device check

**B1**

claims still allocated

0 GPUs free

**B2**

2 GPUs free

Only 2 Pods fit: the workload waits

Both gaps need device state that changes during the simulation

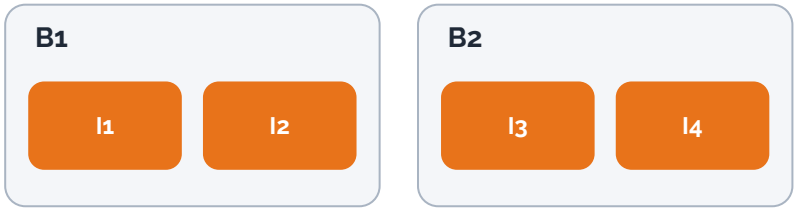
# Simulated device state closes both gaps

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

planned

### Gap 1: reserve as each Pod is placed

B1 is full after I1 and I2, so I3 and I4 go to B2

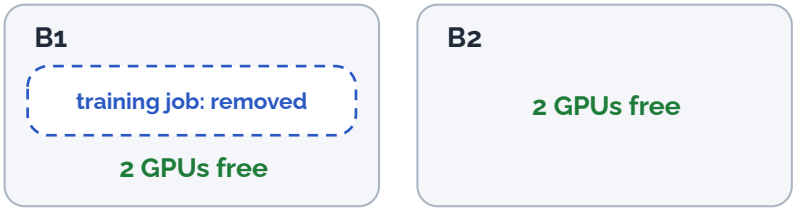



Every Pod has a GPU: 2 on B1 + 2 on B2 ✓

planned

### Gap 2: release the victim's devices

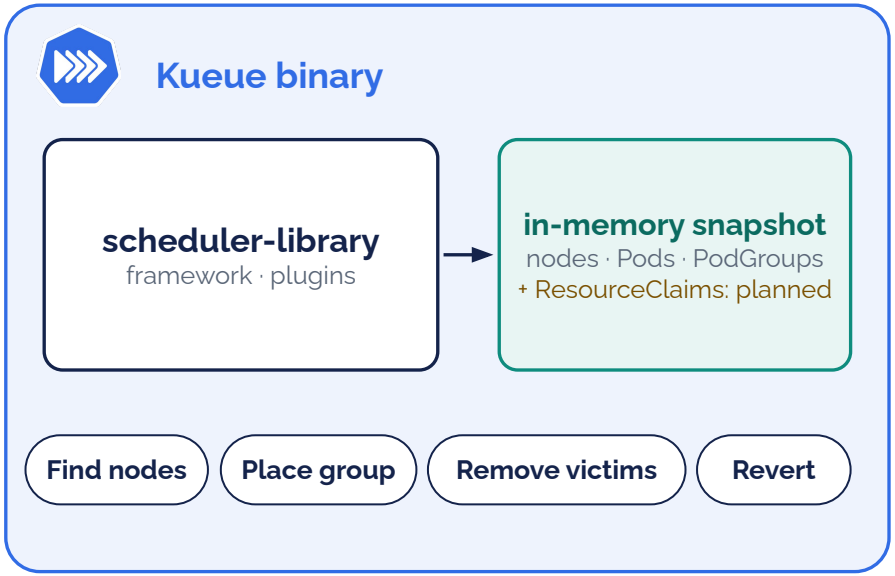
The removed training job's GPUs count as free




All 4 Pods fit: admitted ✓

Planned: Kueue reserves and releases devices inside the simulation

# kube-scheduler as a Go library




## Why kube-scheduler's own DRA plugin?

**Same DRA code as kube-scheduler**
Kueue's own check gives a different answer on 6 DRA features (KEP-2941)

**Example: a claim shared by a group**
its allocation pins the group to one node, but Kueue's check can spread the Pods across nodes

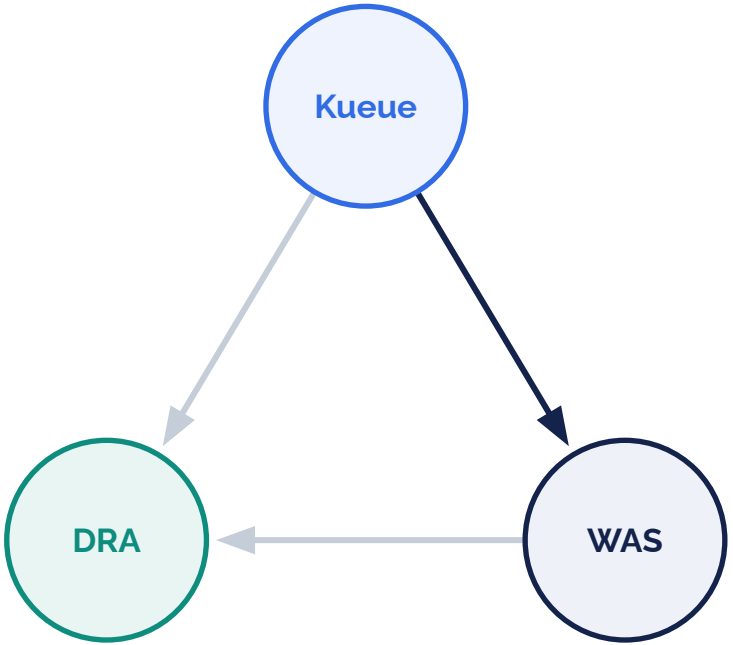
**Long term: kube-scheduler's DRA plugin, run in Kueue, replaces Kueue's check**

# From node checks to group placement




| Question | Status |
| --- | --- |
| **How much to charge?** | ✓ whole GPUs and slices (Beta) · shared GPUs (Alpha) |
| **Does one Pod fit?** | ✓ device check, node by node, before admission (Alpha, opt-in) |
| **Does the group fit?** | ✓ placement by CPU and topology<br>planned: count and release devices across the group |

**Next: workload-aware placement with DRA**




2

# WAS integration deep dive

Kueue → WAS · Michał Woźniak

# What is WAS?

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

## Pod-by-Pod scheduling



Partially scheduled: the job can't progress

## Workload-aware scheduling



Fully scheduled, or not at all

### WAS brings to core Kubernetes

- ✓ **Gang scheduling**
- ✓ **Topology awareness**
- ✓ **Workload-aware preemption**
- … **DRA integration** · in progress

Largely inspired by **Kueue** and other ecosystem projects, which had these concepts long before core Kubernetes

# Why integrate Kueue with WAS?

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

| | |
|---|---|
| **Custom TAS APIs and semantics**<br>not unified across the ecosystem | → **Unified APIs and semantics**<br>defined in core Kubernetes |
| **No custom scheduler plugins**<br>ignored when Kueue places workloads | → **Custom plugins supported**<br>built into Kueue, with Deep integration |
| **Complex code in Kueue**<br>to support Pod directives and DRA devices | → **Reuse kube-scheduler code**<br>Pod directives and DRA, maintained upstream |

# WAS integration principles





## Kueue keeps the policy

Queues, quota, fair sharing, and who may be preempted




## Scheduler code places

Same plugins at admission and at binding: no split-brain

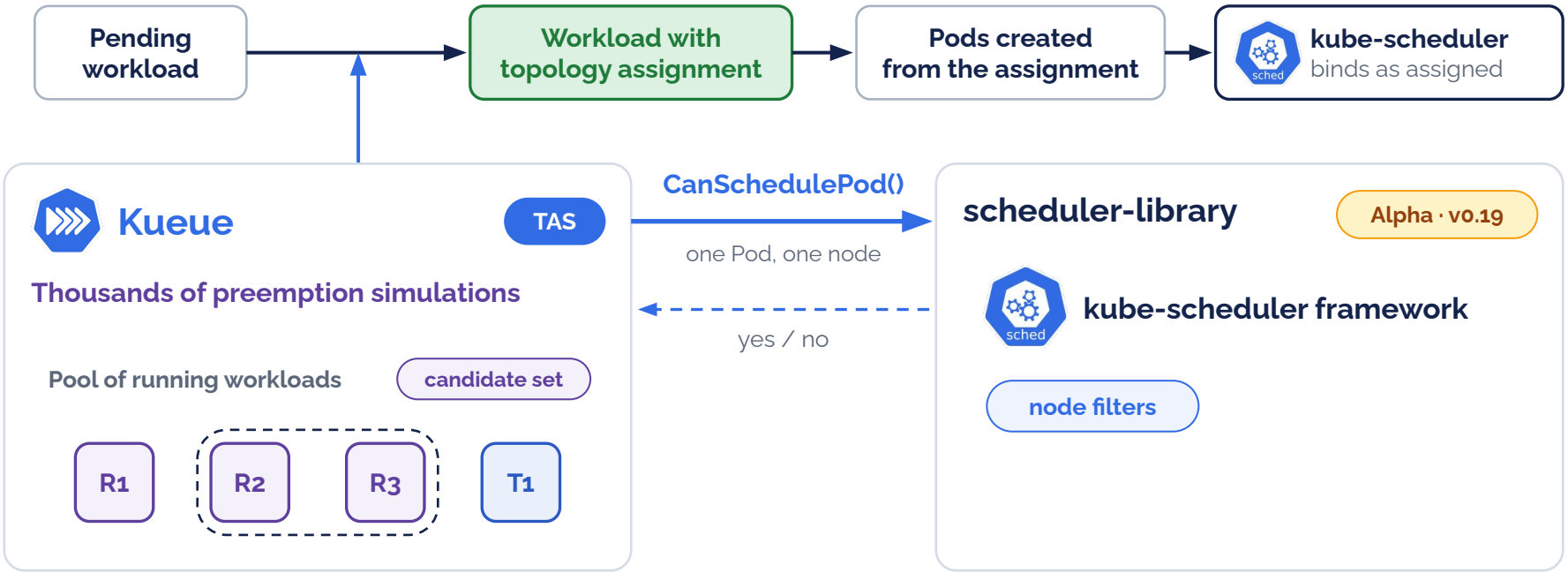



## Simulate in-process

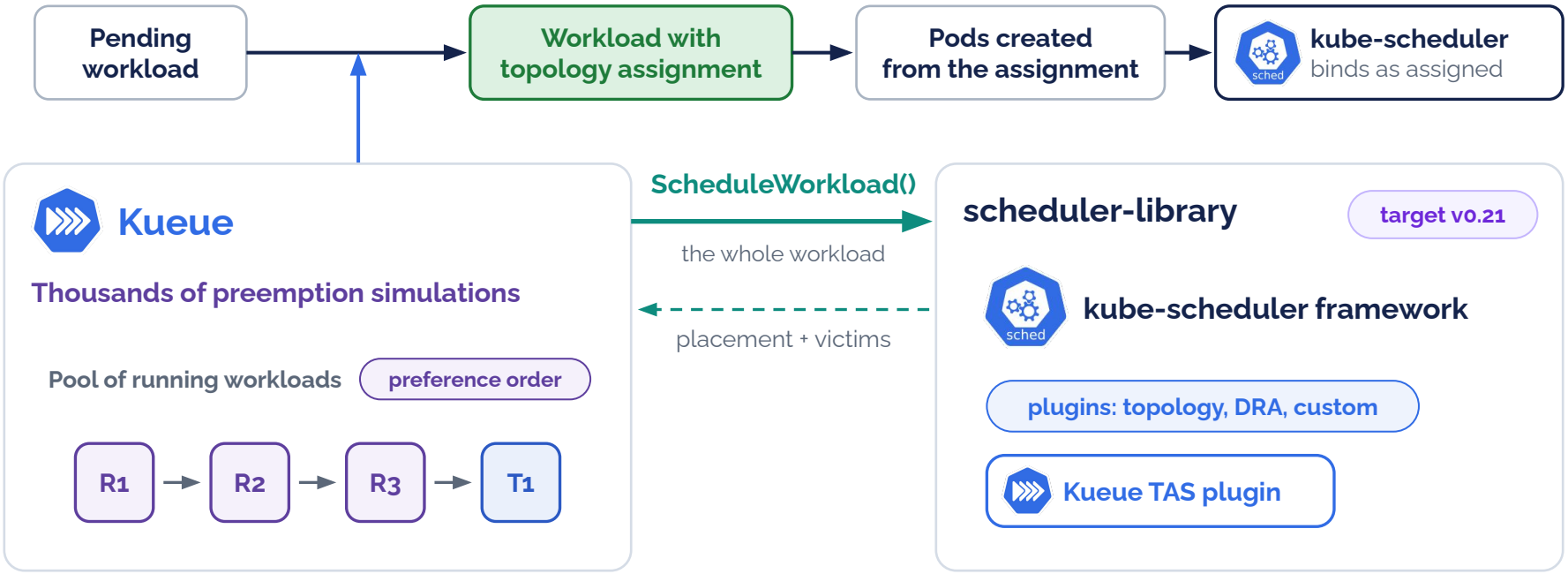
Thousands of simulations per cycle, so the scheduler-library is compiled into Kueue

# Basic integration: feasibility checks

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA



# Deep integration: workload scheduling



# Deep integration: two-phase preemption

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

**From Kueue** R1 R2 R3 T1 ... preemption candidates, ordered by multi-tenant fair sharing

## 1 Find the shortest prefix

Try 1, 2, 4, 8 … until it fits, then narrow down

100 candidates: ~15 tries instead of 100

each try: quota check, then placement in memory

## 2 Place, then backfill

Place the workload once, without that prefix

Restore what isn't needed, in reverse order

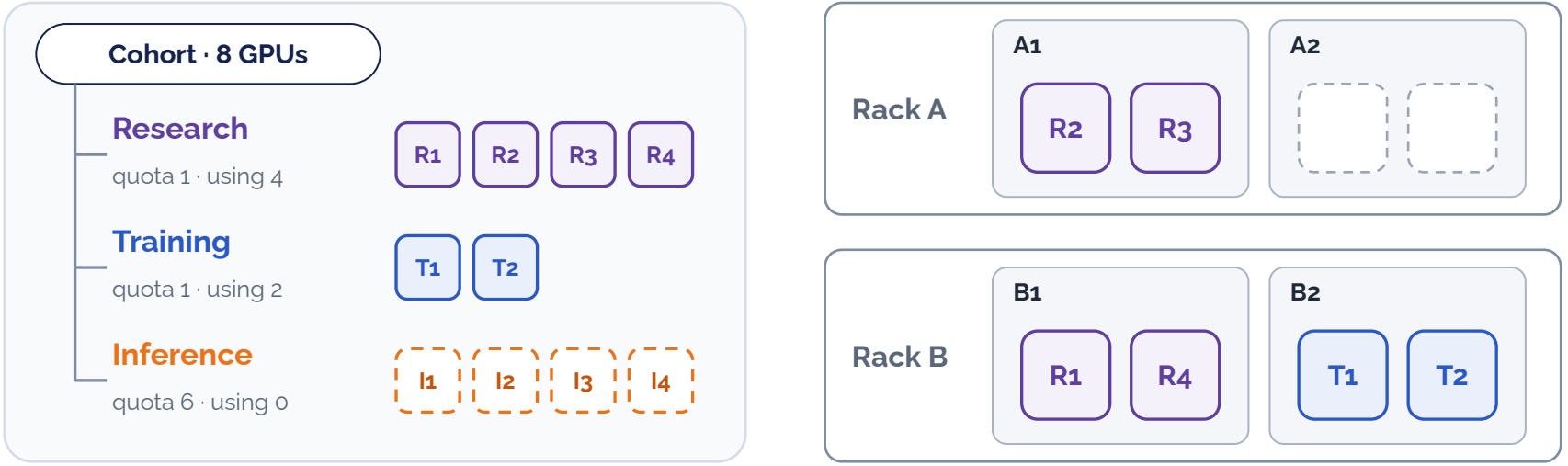
each restore: only if quota and topology still fit

**Back to Kueue** **the placement and the victims** → Kueue evicts, then admits

Inspired by **Kueue**'s preemption, optimized: binary search in phase 1, fixed placement in phase 2

# Example: who gets preempted?

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

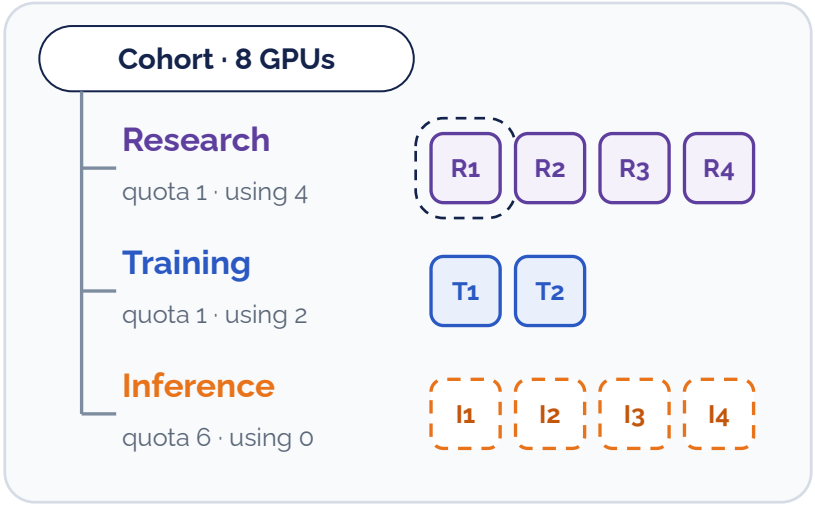



**Preference order:** R1 → R2 → R3 → T1 · R4, T2 never offered: each team keeps its quota

**Inference wants 4 GPUs back**
I1–I4: 4 Pods × 1 GPU, all in one rack

**Fair sharing: biggest borrower first**
Research borrows 3, Training 1 · never below quota

# Phase 1: try 1 candidate







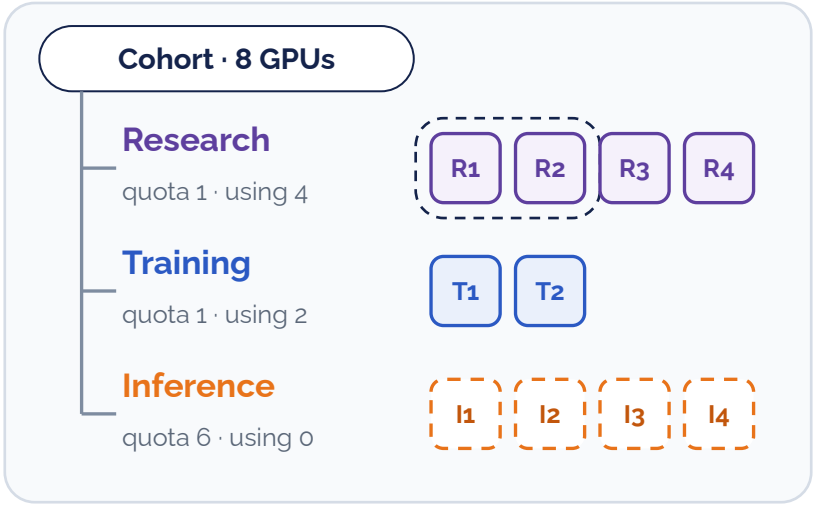
Preference order: R1 → R2 → R3 → T1

**Quota ✗**
3 GPUs free, 4 needed

→

**Topology**
skipped: quota failed first

# Phase 1: try 2 candidates







Preference order: R1 → R2 → R3 → T1

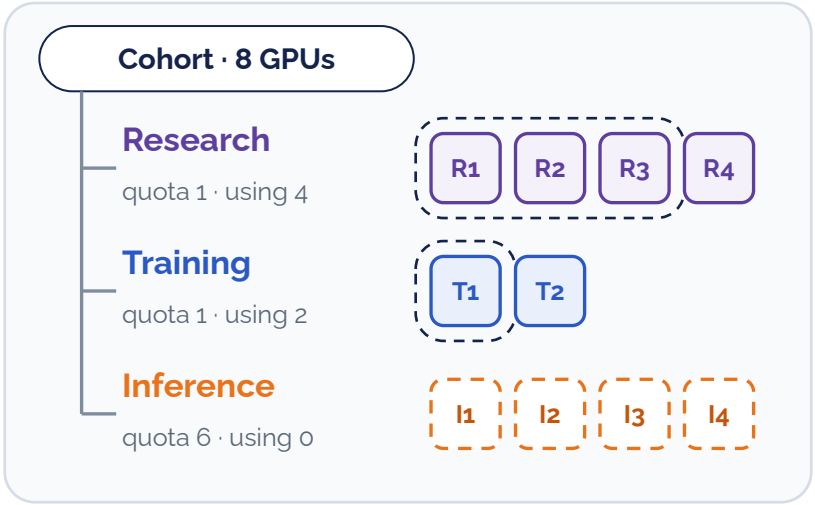
**Quota ✓**
4 GPUs free: enough

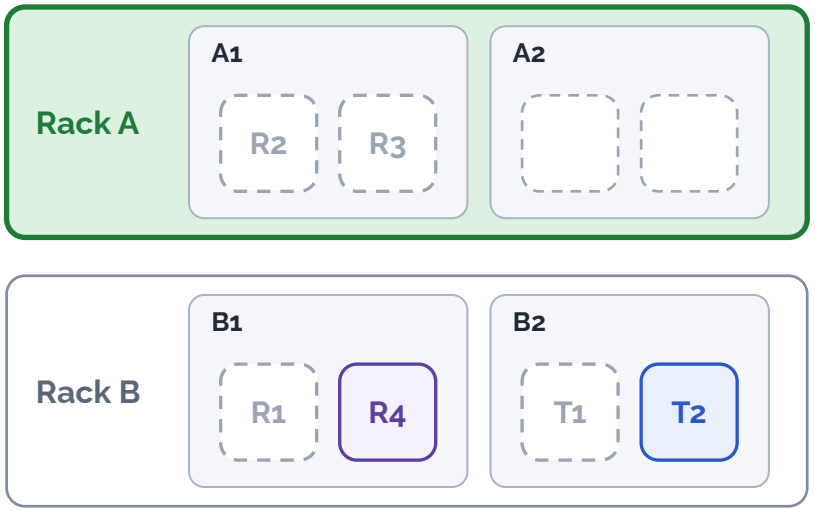
→

**Topology ✗**
rack A 3 free, rack B 1

# Phase 1: try 4 candidates





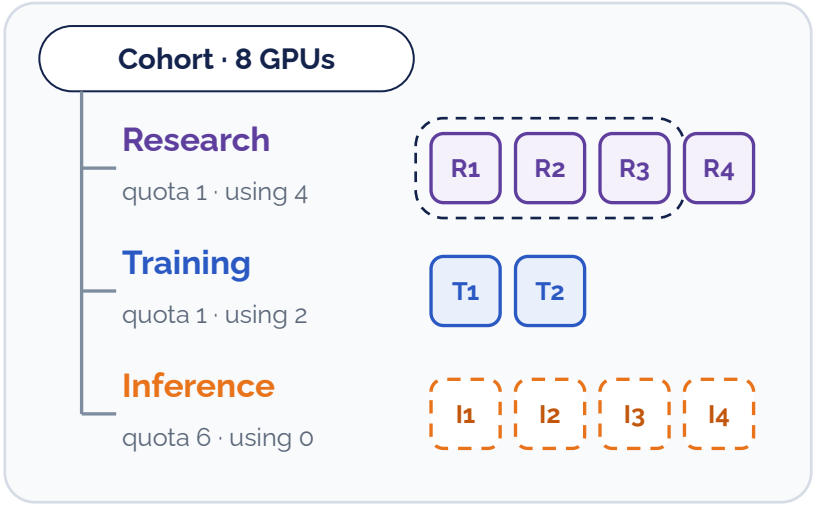

Preference order: R1 → R2 → R3 → T1

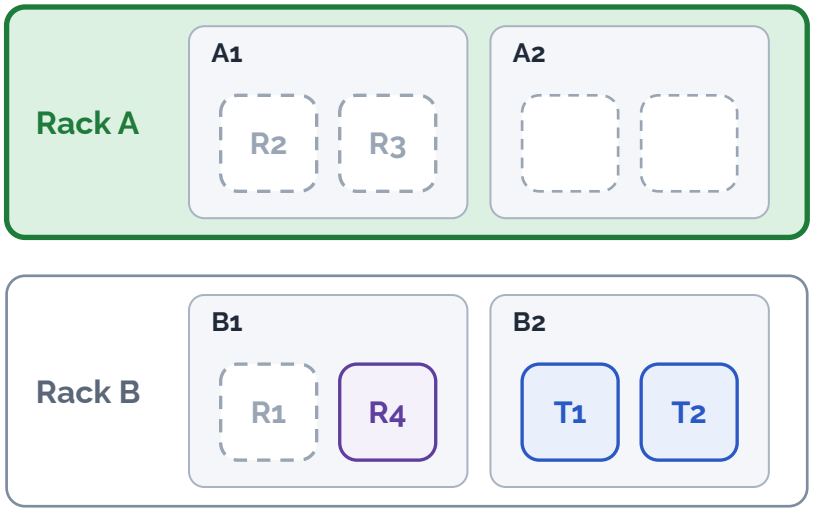
**Quota ✓**
6 GPUs free: enough

→

**Topology ✓**
rack A has 4 free GPUs

# Phase 1: narrow to 3 candidates




Preference order: R1 → R2 → R3 → T1

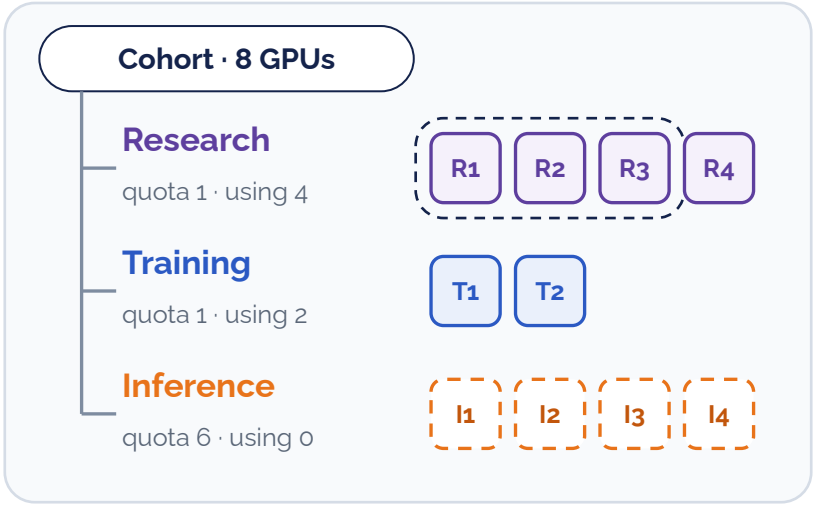
**Quota ✓**
5 GPUs free: enough

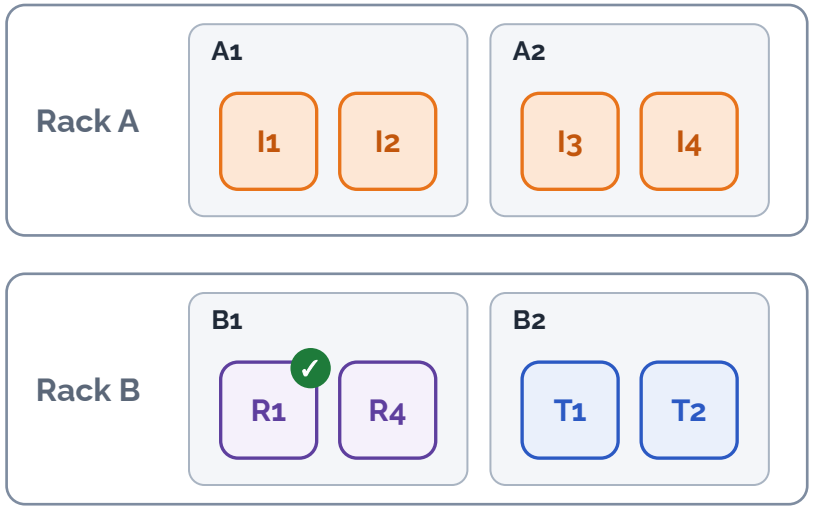
→

**Topology ✓**
rack A has 4 free GPUs

# Phase 2: place, then backfill



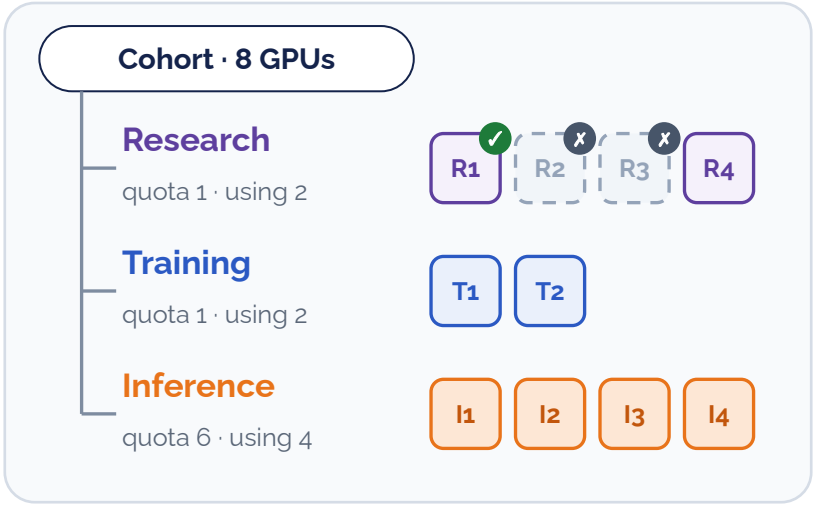


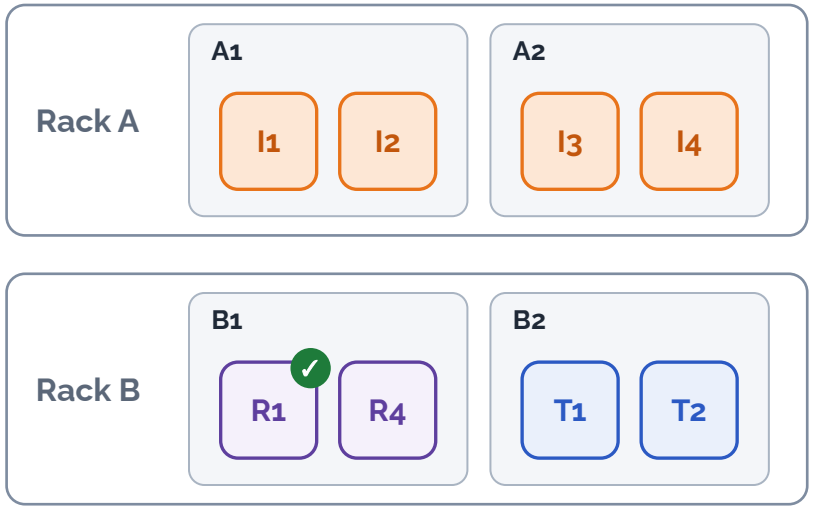




**Backfill in reverse order:** R3 → R2 → R1 · each one comes back only if quota and topology still fit

| | | |
|---|---|---|
| R3 | ✗ not restored<br>GPU taken by I2 | |
| | R2 | ✗ not restored<br>GPU taken by I1 |
| | | R1 ✓ restored<br>B1 is untouched |

# Result: preempted and placed




**✗ Preempted: R2, R3**
Kueue evicts them

**✓ Saved: R1**
Kueue's 1st pick, restored

**Placed: I1–I4 on rack A**
all 8 GPUs in use

# Summary




| | **Native TAS**<br>Beta · on by default | **Basic integration**<br>Alpha · v0.19 | **Deep integration**<br>target v0.21 |
|---|---|---|---|
| **Placement** | **Kueue** | **Kueue** | **scheduler-library** |
| **Node checks** | **Kueue's own subset** | ✓ **kube-scheduler filters** | ✓ **kube-scheduler filters** |
| **Kueue TAS features** | ✓ **all** | ✓ **all** | **"required" only for now**<br>no hot swap |
| **Pod affinity & spread** | ✗ not a goal | ✗ not a goal | ✓ **all Pod directives** |
| **Performance** | **highly optimized** | **similar to native** | **key challenge** |
| **DRA** | **device check per node**<br>Alpha, opt-in | **device check per node**<br>Alpha, opt-in | **DRA plugin, whole gang**<br>in progress |
| **Custom plugins** | ✗ | ✗ | ✓ **built into Kueue** |

# More on DRA and WAS this week

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA

**Mon 3:20pm** · Room 251 D-F
**DRA + Kueue + HAMi + vLLM: A Production Architecture for Scheduler-Agnostic GPU Sharing**
Mengxuan Li, Yuchen Fama

**Tue 12:20pm** · Room 250 D-F
**The Latest in GPU, TPU, NIC and Other Device Support – WG Device Management**
John Belamaric, Morten Jæger Torkildsen

**Tue 4:40pm** · Room 255 BC
**Why Are You Still Wasting Whole GPUs? Hardware-Level Sharing, Scheduled with DRA**
Engin Diri

**Wed 5:00pm** · Grand Ballroom ACE
**One Workload to Rule Them All: Workload Aware Scheduling for Distributed AI Training on Kubernetes**
Andrey Velichkevich

**Thu 11:00am** · Room 250 A-C
**WG-Workload Aware Scheduling: An Introduction**
Kevin Hannon, Andrey Velichkevich, Heba Elayoty, Matt Matejczyk

**Thu 11:50am** · Room 250 A-C
**SIG Scheduling Update: Multi-Level and Topology-Aware Workload Scheduling**
Heba Elayoty, Dominik Marcinski

**Project Pavilion, Hall 5** · **P-5A** Kueue / WAS, Tue–Thu · **P-6A** DRA, Tue 10:30am–2:30pm

# Questions?

CLOUD NATIVE AI + INFERENCE DAY NORTH AMERICA




**Michał Woźniak**

Google

GitHub, Slack: mimowo





**Sohan Kunkerkar**

Red Hat

GitHub: sohankunkerkar


**Rate this session**

KUEUE BOOTH · TUE–THU

P-5A, Project Pavilion, Hall 5

SLACK CHANNEL

slack.k8s.io #wg-batch

KUEUE ON GITHUB

github.com/kubernetes-sigs/kueue