CLOUD NATIVE
AI DAY
EUROPE

Scale your Batch / AI Workloads beyond the Kubernetes Scheduler

Antonin Stefanutti
Anish Asthana

Scale your Batch / AI Workloads beyond the Kubernetes Scheduler

CLOUD NATIVE AI DAY EUROPE

19 March 2024 | Paris, France

Antonin Stefanutti
Software Engineer
Red Hat

Anish Asthana
Engineering Manager
Red Hat

Agenda

CLOUD NATIVE
AI DAY
EUROPE

- Online Services / Offline Jobs
- The Kubernetes Scheduler
- Batch Schedulers
- Projects Review

Online Services / Offline Jobs

CLOUD NATIVE AI DAY EUROPE

	Online Services	Offline Jobs
Examples	Web applications Microservices / Serverless	AI/ML distributed model training
State	Stateless (mostly)	Stateful
Replicas	Homogenous	Heterogenous
Processing Unit	Request / Thread	Data Partition / Worker
SLI	Latency Availability	Throughput Efficiency
Compute Sizing	Fixed / Limited variability	High variability
Auto-scaling Unit	Pod	Node

The Kubernetes Scheduler

A simple scalability test:

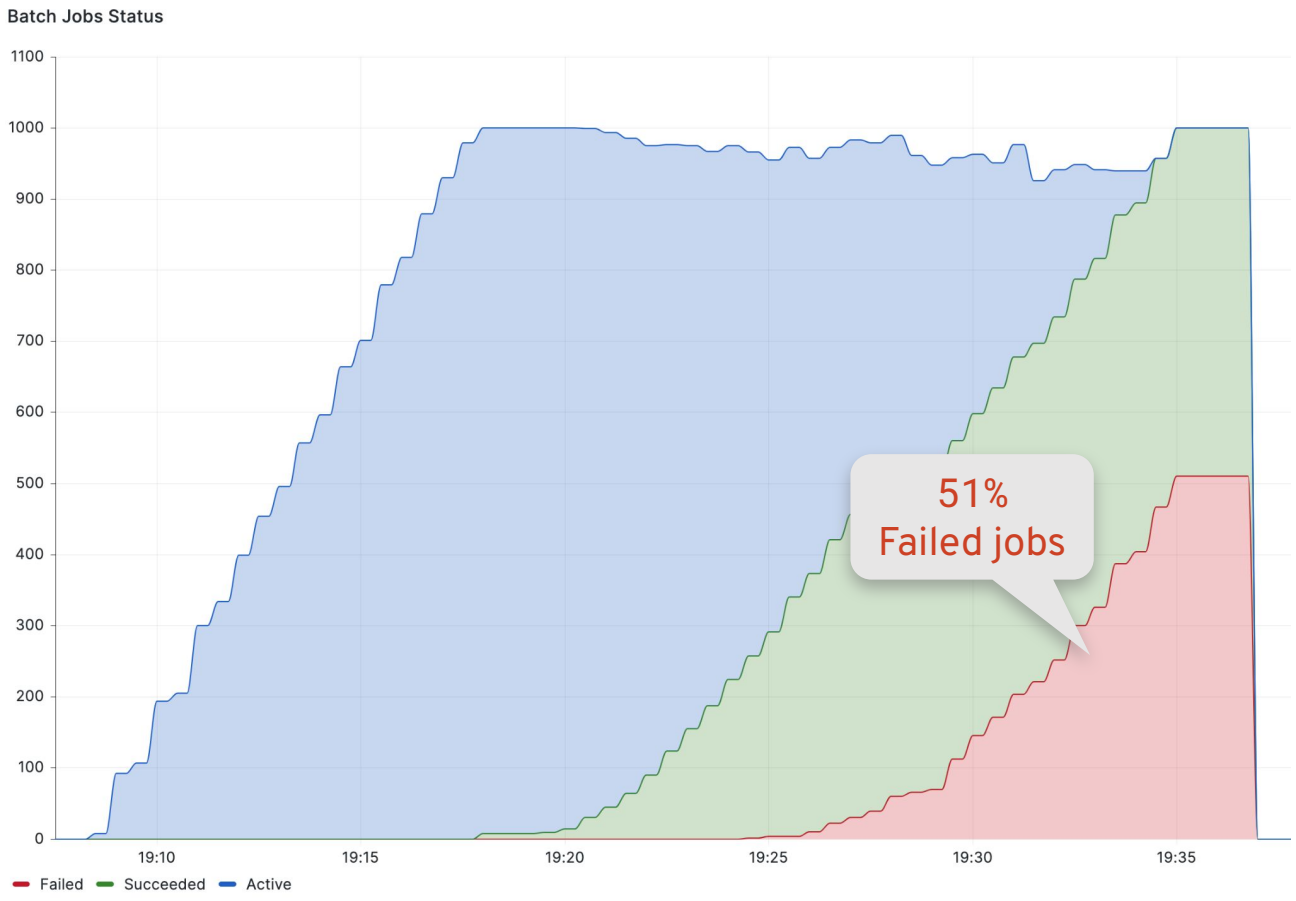
- **1000 batch Jobs**:
  - Fixed completion count:
    - 10 parallel Pods
  - Completion duration: 3 min + 50% jitter
    - AFTER all Pods ready
  - Completion deadline: 15 min
- **100 Nodes**, simulated with KWOK:
  - Total allocatable resources:
    - 1000 CPUs
    - 1000Gi memory
- **Max throughput***: ~1000 Jobs / 10 min

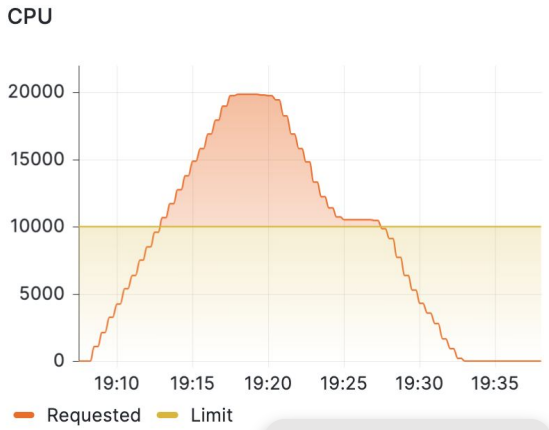
*theoretical

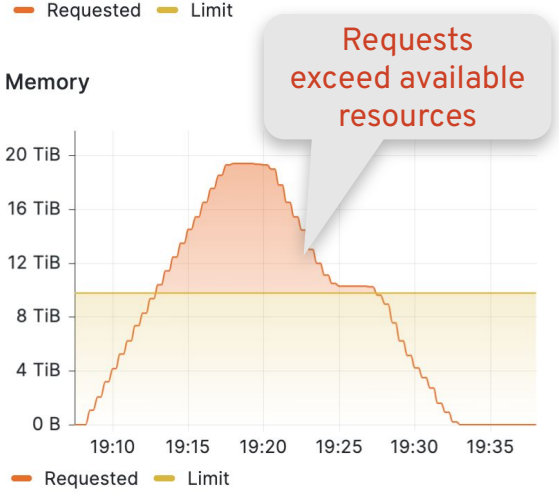
```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: sample
spec:
 completions: 10
 parallelism: 10
 backoffLimit: 0
 activeDeadlineSeconds: 900
 template:
   spec:
     restartPolicy: Never
     containers:
     - name: job
       resources:
         limits:
           cpu: "2"
           memory: 2Gi
         requests:
           cpu: "2"
           memory: 2Gi
```

The Kubernetes Scheduler

CLOUD NATIVE AI DAY EUROPE



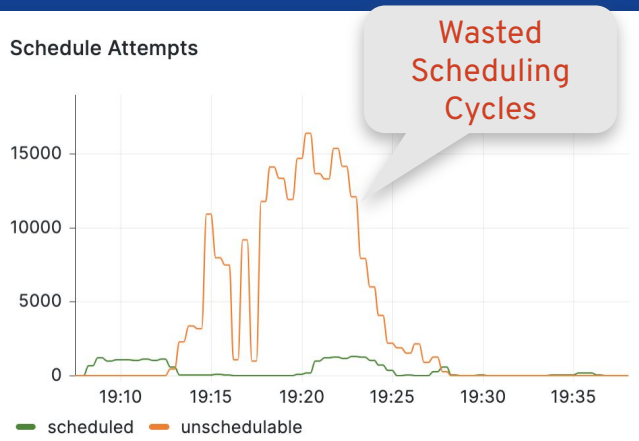




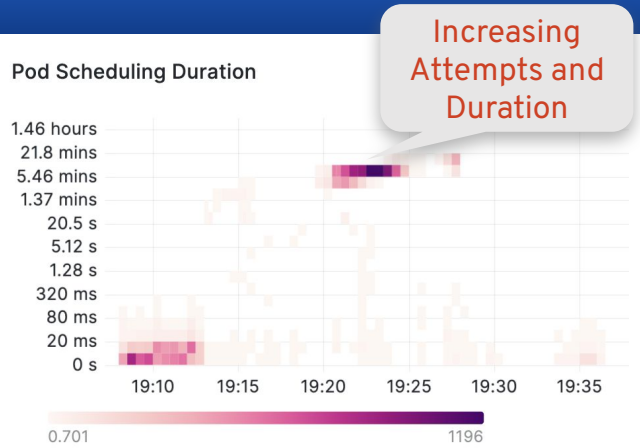
The Kubernetes Scheduler

CLOUD NATIVE AI DAY EUROPE

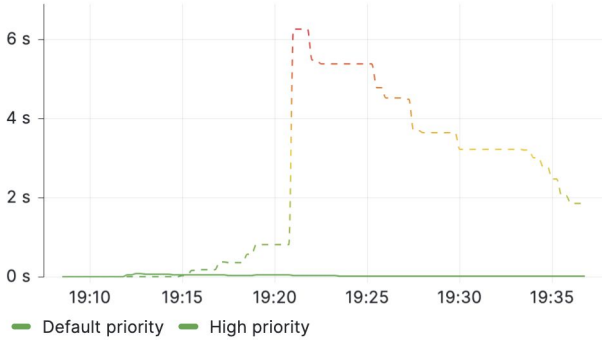
Schedule Attempts



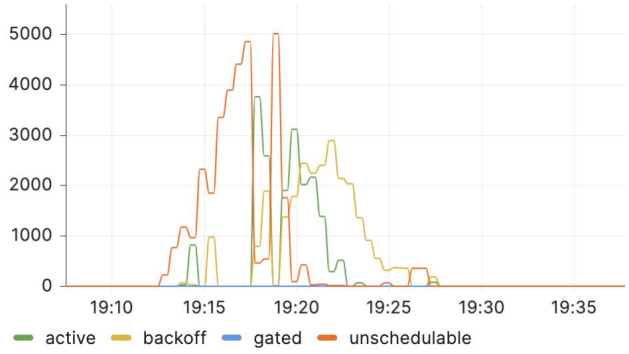
Pod Scheduling Duration



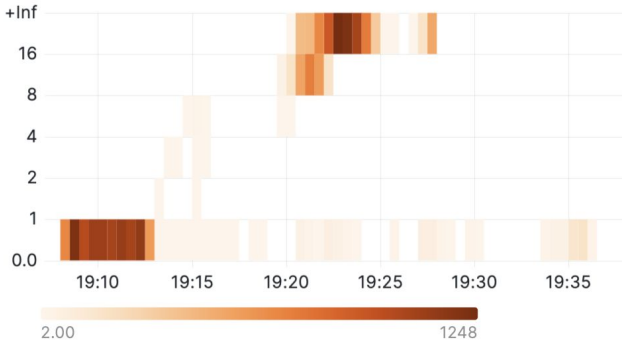
Average Sample Pods Scheduling Time



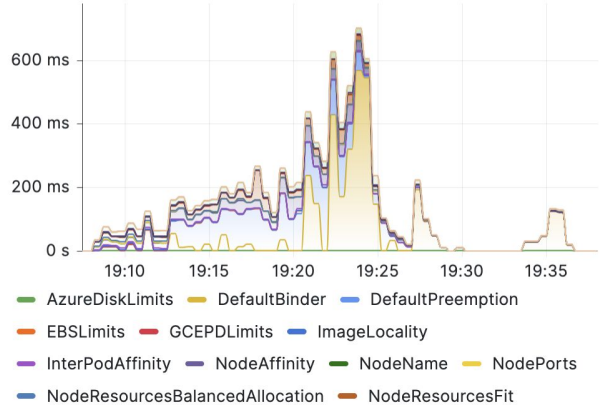
Pending Pods (by queue)



Pod Scheduling Attempts



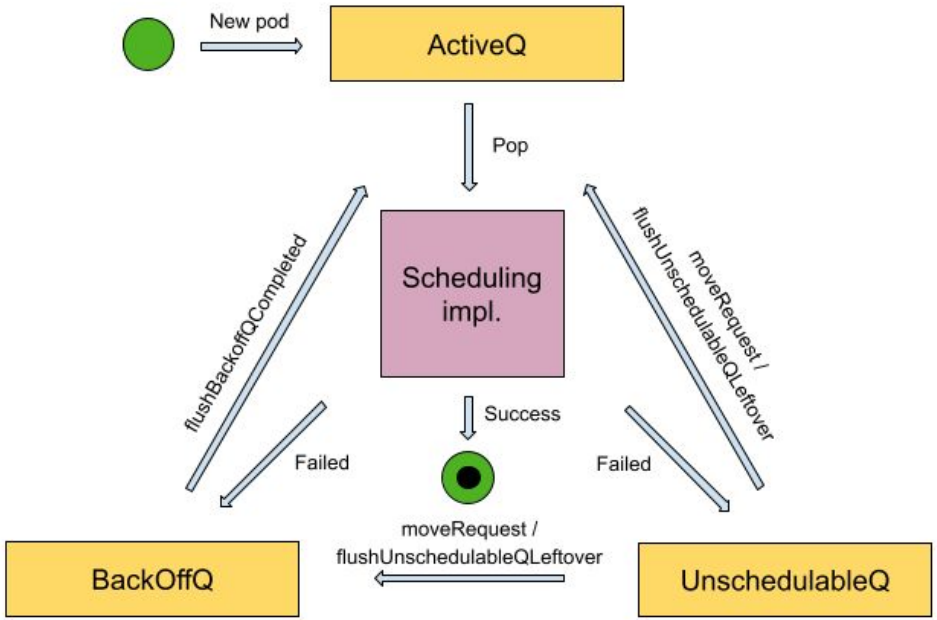
Plugins Execution Duration



The Kubernetes Scheduler

Scheduling queues in kube-builder:

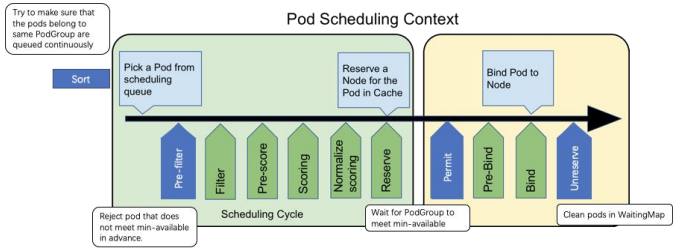
[sig-scheduling/scheduler_queues.md]

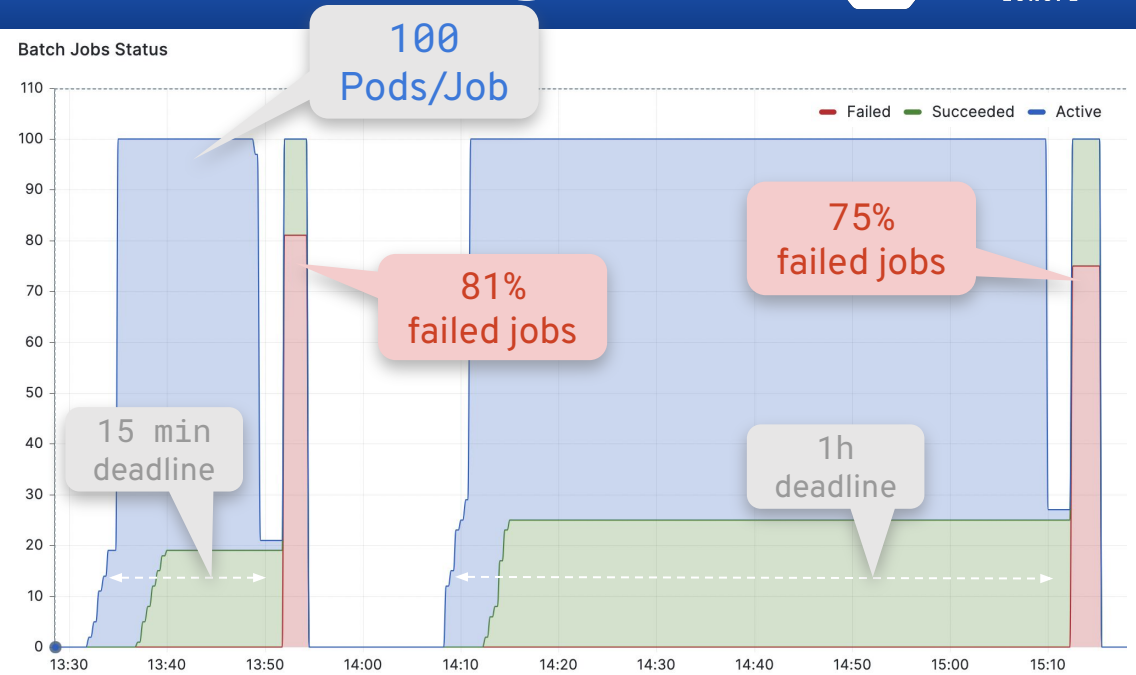


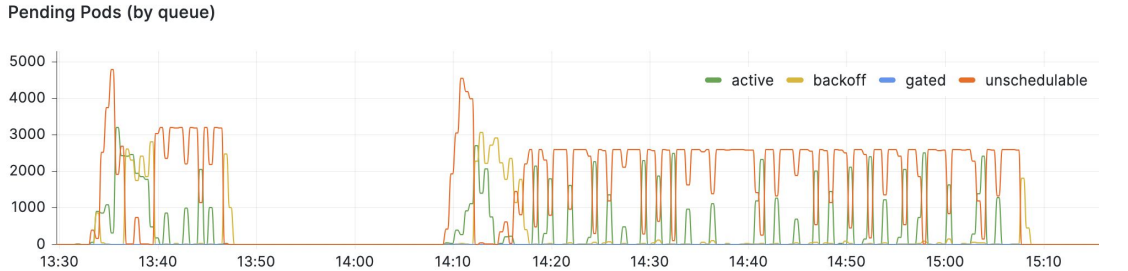
Coscheduling Scheduler Plugin

- Job failures ratio increases with **parallelism**
- Too many concurrent Pods leads to **resources fragmentation** and **deadlock** ultimately

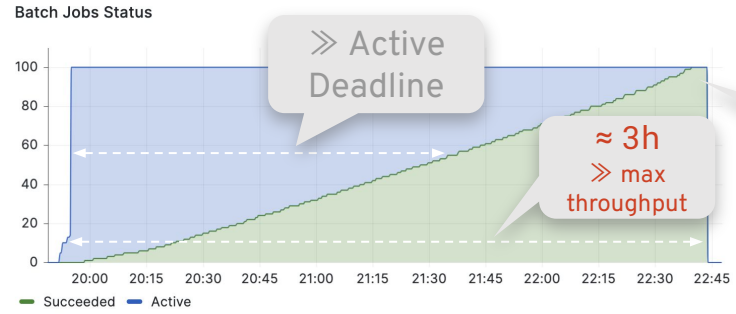
- Gang-scheduling with the **coscheduling** kube-scheduler plugin:

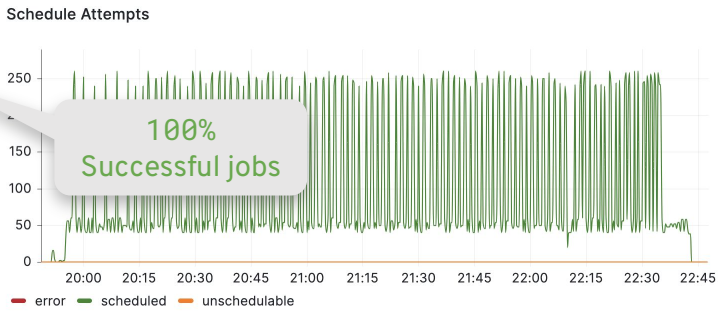


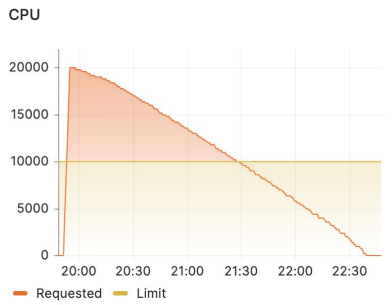


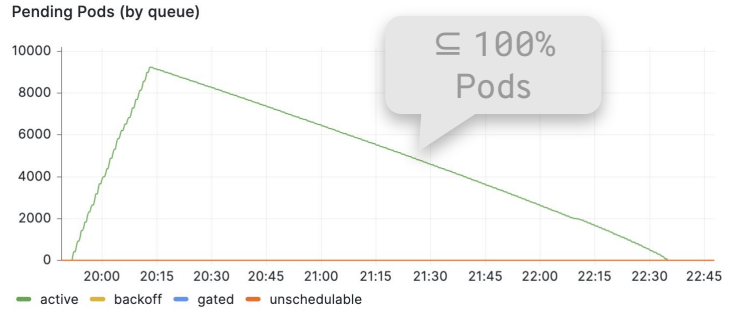


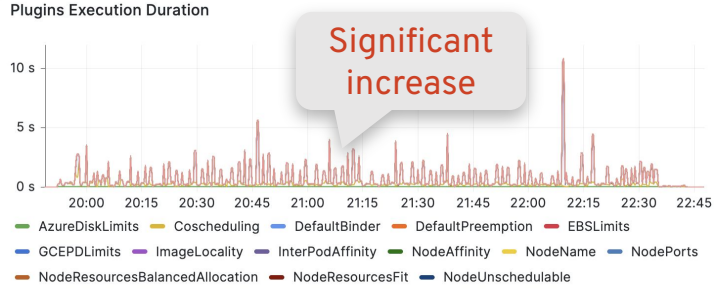
Coscheduling Scheduler Plugin

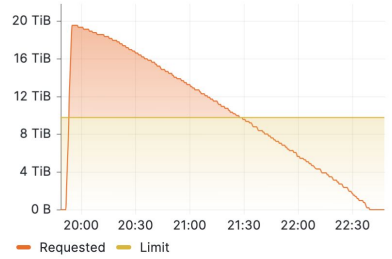


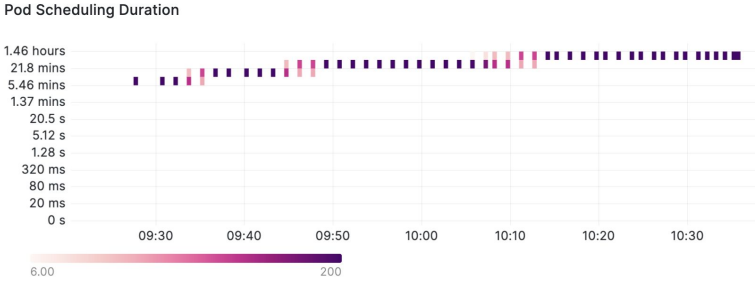


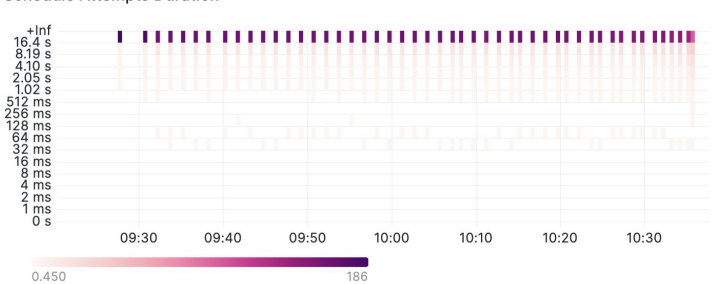


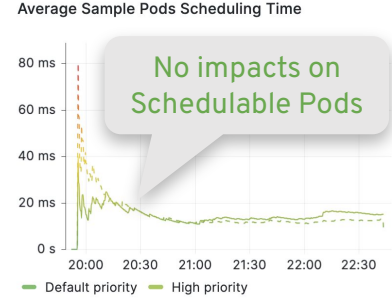






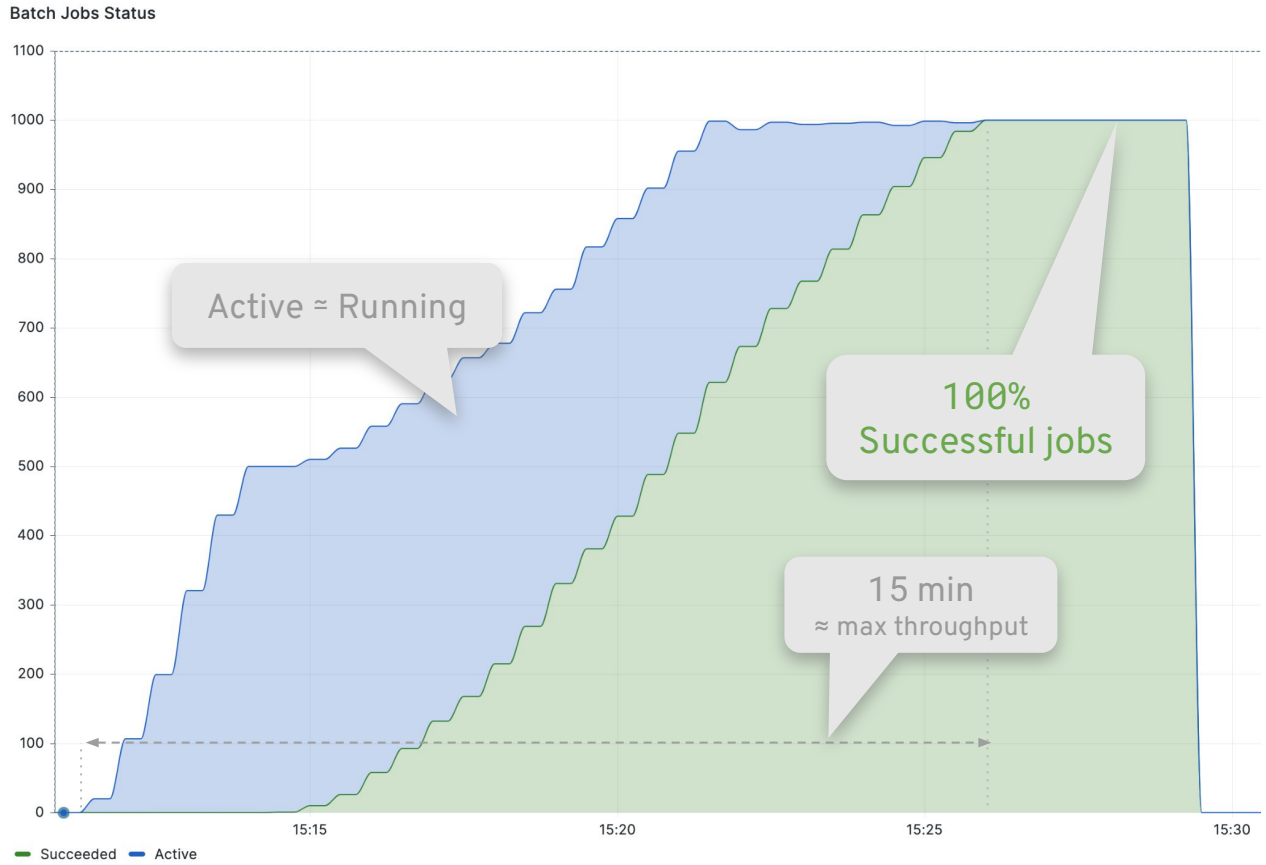


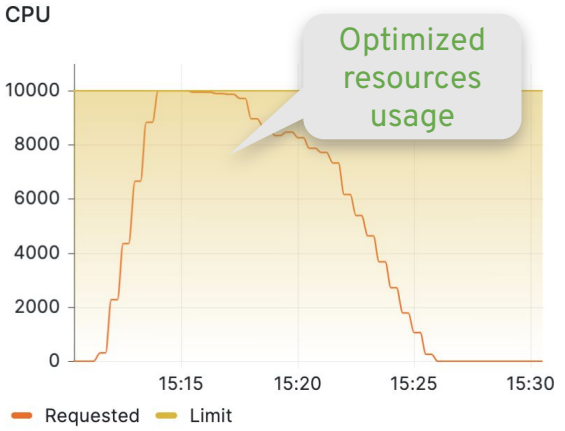


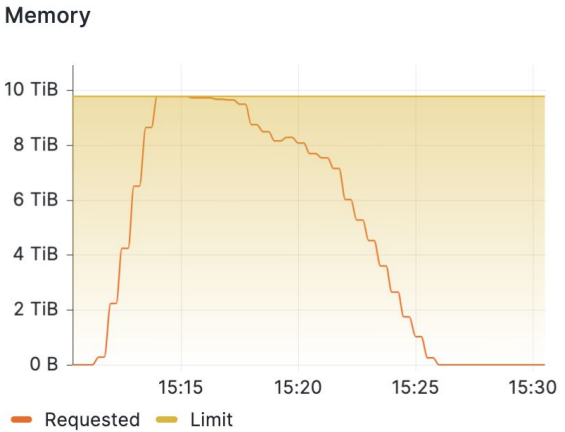


Example with a Job Scheduler (Kueue)

CLOUD NATIVE AI DAY EUROPE



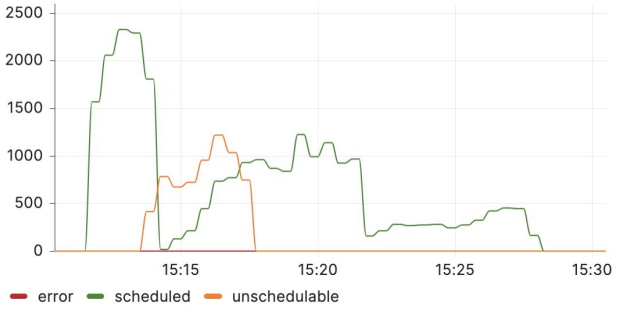




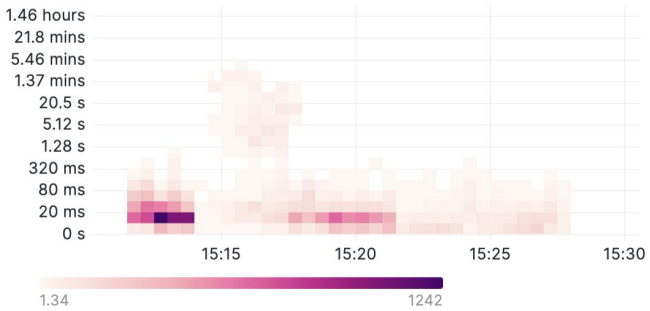
Example with a Job Scheduler (Kueue)

CLOUD NATIVE AI DAY EUROPE

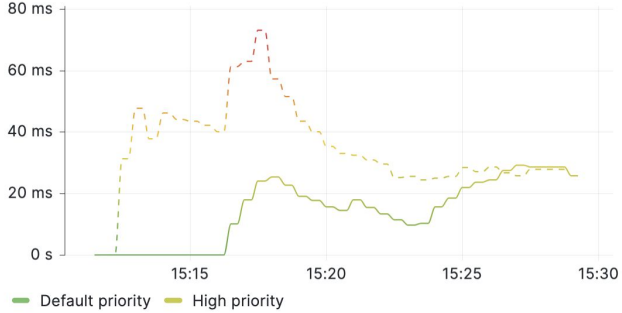
Schedule Attempts



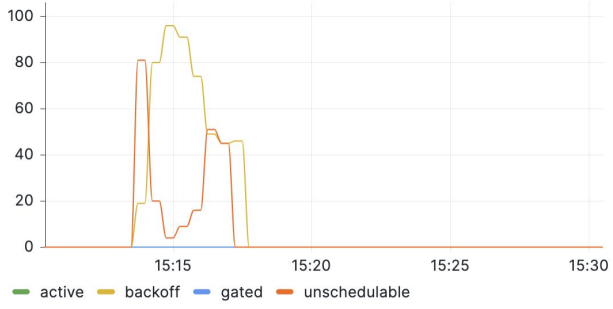
Pod Scheduling Duration



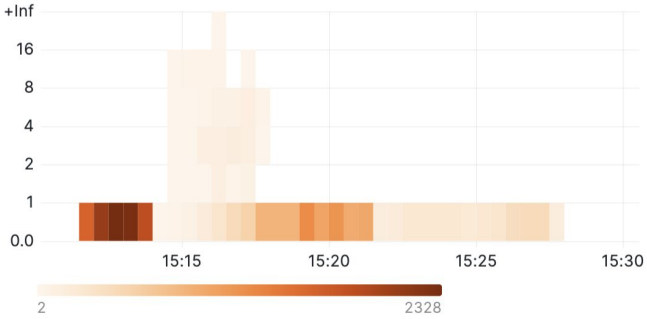
Average Sample Pods Scheduling Time



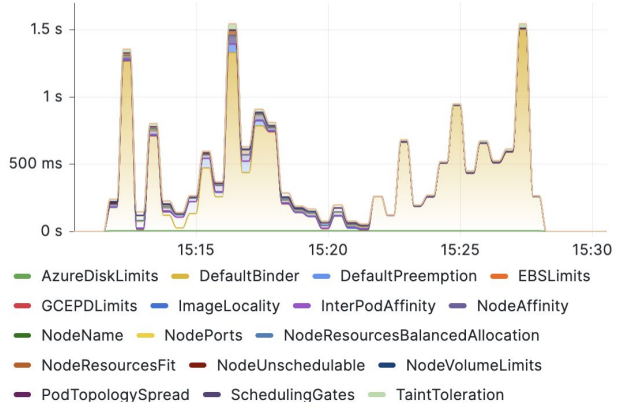
Pending Pods (by queue)



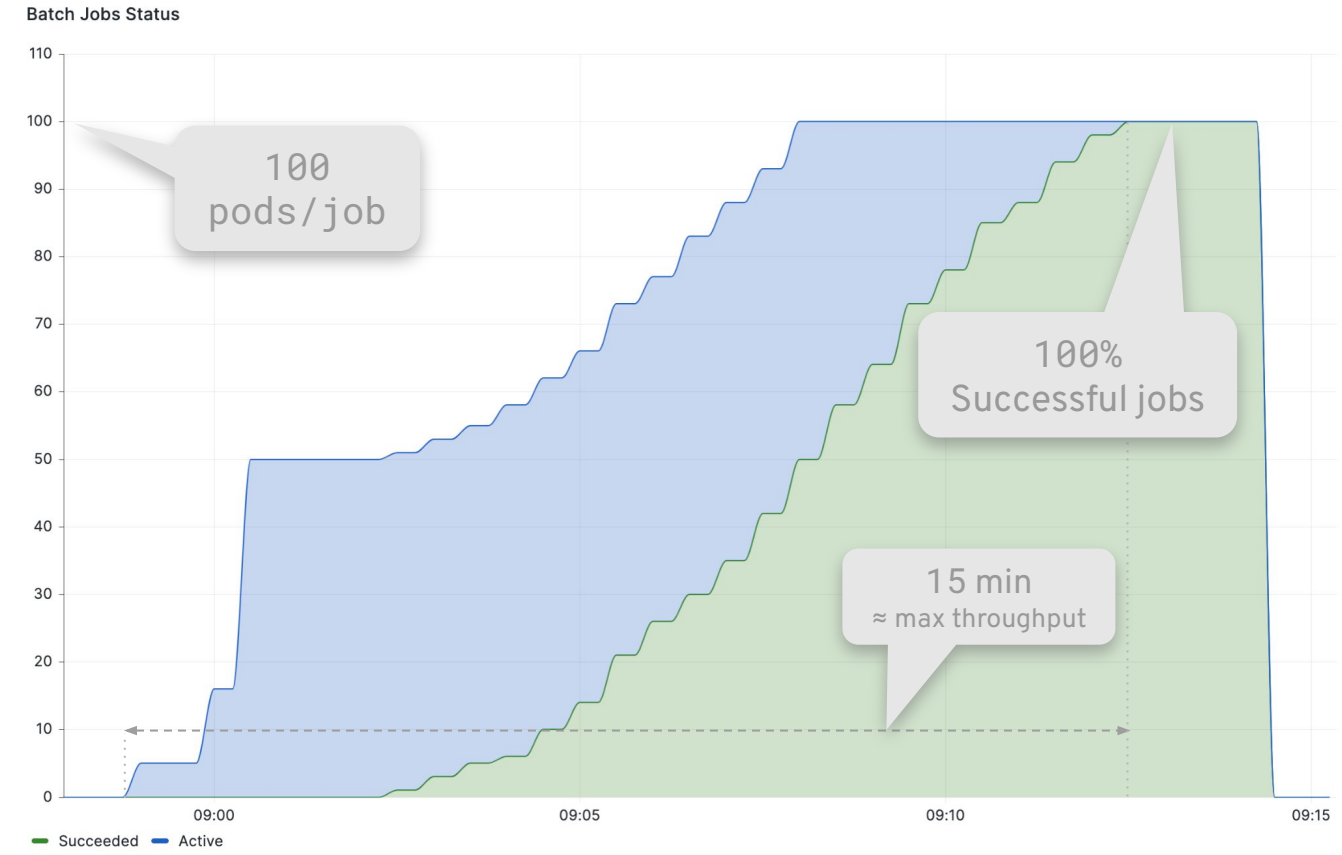
Pod Scheduling Attempts

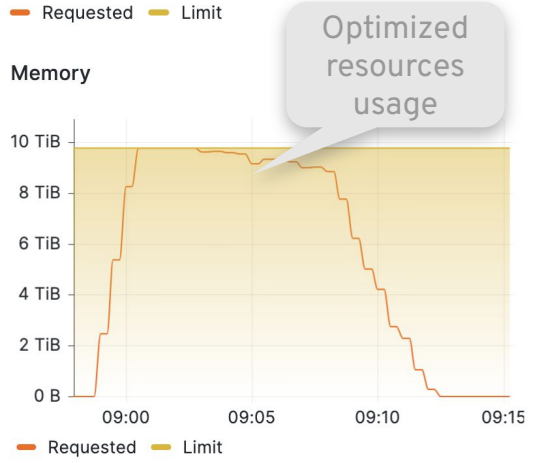


Plugins Execution Duration



Example with a Job Scheduler (Kueue)





Job Schedulers

	Custom Schedulers	Queue Managers
Design	Replace the default Scheduler	“Front” the default Scheduler
Implementation	Kubernetes Scheduler Framework / Standalone	Controllers
Responsibilities	All-in-One Queuing + Scheduling	Single Responsibility Principle Queuing Only
Logical Unit	Group of Pods	High-level Jobs
Scheduling	Guaranteed	Independent
Projects	Koordinator Volcano YuniKorn	Kueue MCAD

Koordinator

koordinator.sh

Architecture	Custom Scheduler Controller and Admission Webhooks DaemonSet for Nodes profiling (Koordlet)	Custom Kubernetes scheduling framework plugins
Workload Types	Any kind of workloads Operates at Pods level	Requires `schedulerName` field if deployed as a secondary scheduler
Queuing / Scheduling	Gang Scheduling via the `PodGroup` API Load aware via Koordlet deployed as a DaemonSet Fine-grained Scheduling via custom SLO / QoS	Custom kube-scheduler plugins
Quota Management	Via the scheduler-plugins `ElasticQuota` API Multi-Hierarchical Elastic Quota Management Implementation	Extensions of the kube-scheduler plugins via custom annotations (e.g. for hierarchy)
Heterogeneous Workloads / Clusters	Provides the `Device` API Fine-grained Device Scheduling	
Auto-scaling	Via Cluster Autoscaler	
Multi-Cluster Support	No	Based on the scheduler framework

MCAD

codeflare.dev

Architecture	Controller	Queues the creation of the wrapped resources so it delays the creation of Pods by the underlying controllers
Workload Types	Provides the AppWrapper API that can encapsulate any kind of resources	
Queuing / Scheduling	Priority, Preemption, Borrowing Gang scheduling via the `PodGroup` API	Takes nodes allocatable resources into account to adjust the size of the queue
Quota Management	Provides the `QuotaSubtree` API	Multiple level of hierarchy
Heterogeneous Clusters / Devices	Relies on extended resources	
Auto-scaling	Yes, via the MachineSet / MachinePool APIs	OpenShift specific Implemented by InstaScale
Multi-Cluster Support	Yes	Early phase

Volcano

volcano.sh

Architecture	Custom Scheduler Controller and Admission Webhooks	
Workload Types	Provides the `VolcanoJob` API Numerous clients integration (Flink, KubeFlow, Ray, Spark, TorchX, …)	Set of `PodTemplateSpec`
Queuing / Scheduling	Priority, Preemption, Borrowing FIFO, Dominant Resource Fairness sorting policies Gang scheduling via the provided `PodGroup` API	Custom `PodGroup` API
Quota Management	Provides the `Queue` API Weighted / Proportional resources	No hierarchy Proportional queue sizes
Heterogeneous Clusters / Devices	Relies on extended resources NUMA-aware plugin	Support topology manager scope and policies
Auto-scaling	Via Cluster Autoscaler	
Multi-Cluster Support	Delegates to external projects	Custom scheduler

Apache YuniKorn

yunikorn.apache.org

Architecture	Custom Scheduler (standard or scheduler plugins) Controller and Admission Webhooks Standalone core scheduler + k8s shim	Admission webhook ensures only one scheduler is active (excluding YuniKorn itself)
Workload Types	Any kind of workloads Operates at Pods level	`schedulerName` field added by the admission webhook
Queuing / Scheduling	Priority, Preemption FIFO and fair-share sorting policies Gang scheduling via custom `TaskGroup` annotations	Resources reservation with placeholder Pods Application id label must be set manually in Pod templates
Quota Management	Custom ACL and hierarchical queues configuration RBAC via custom user info annotation	User info annotation is enforced by the admission webhook
Heterogeneous Clusters / Devices	Via extended resources	
Auto-scaling	Via Cluster Autoscaler	Optimised with resources reservation
Multi-Cluster Support	No	Based on the scheduler framework

Kueue

CLOUD NATIVE AI DAY EUROPE

kueue.sigs.k8s.io

Architecture	Controller and Admission Webhooks	Gates the creation of Pods by the underlying job controller
Workload Types	Pods, Jobs, JobSets, Kubeflow Jobs, RayClusters, RayJobs	Requires *suspend* semantic Transparent to the clients
Queuing / Scheduling	Priority, Preemption, Borrowing, Partial Admission Gang scheduling via serialised admission based on readiness Provides the `AdmissionCheck` API as extension points	Strict FIFO,Best Effort FIFO (fair-share coming soon) strategies Independent from the current cluster load Can work with external PodGroup
Quota Management	Provides the `ClusterQueue` and `LocalQueue` APIs Elastic quota within cohorts and lending limits	Only 2 levels of hierarchy at the moment Batch admin users only, RBAC enforced
Heterogeneous Clusters / Devices	Provides the `ResourceFlavor` API	Checks node affinity and inject node selectors and tolerations to Pod templates
Auto-scaling	Via the CA `ProvisioningRequest` API	Uses the `AdmissionCheck` API
Multi-Cluster Support	Yes (alpha)	Alpha feature Uses the `AdmissionCheck` API

CLOUD NATIVE AI DAY EUROPE

Thank You!

Feedback