KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2019

KubeCon | CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

Co-Location of Workloads with High Resource Efficiency

Penghao Cen (@ScorpioCPH) - Ant Financial
Jian He (@jian-he) - Alibaba Cloud

About Us

- Penghao Cen
    - Ant Financial - Infra & Data - System Engineering
    - Scheduling & Cluster Resource Management & Workloads Colocation
- Jian He
    - Alibaba Cloud Container Service

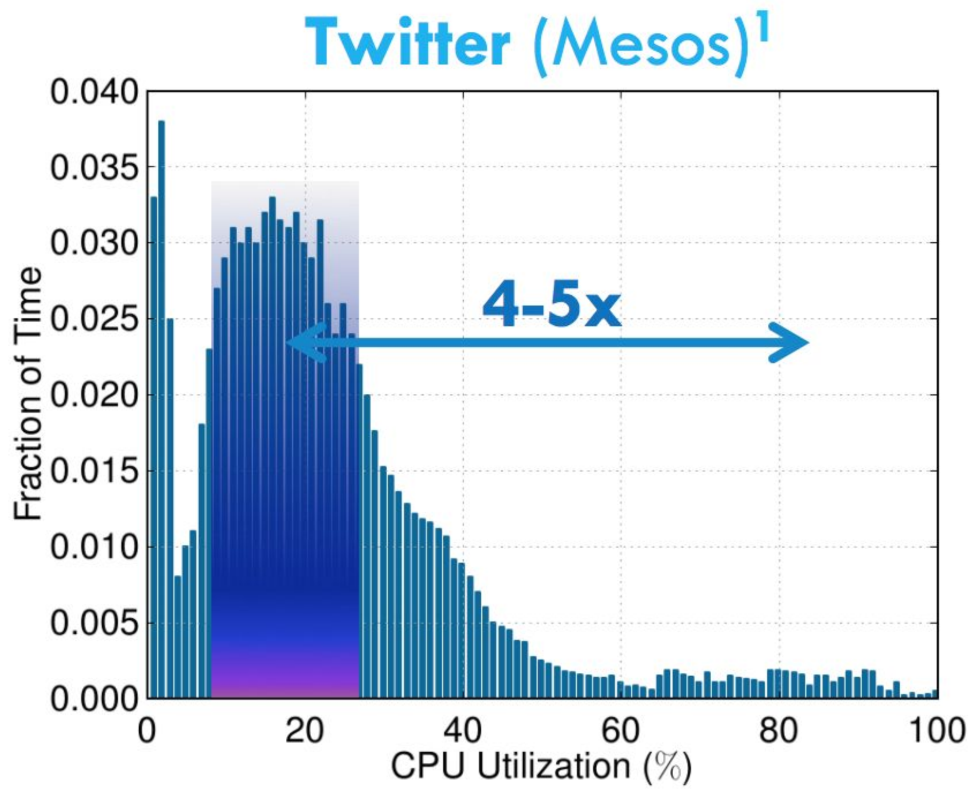
Agenda

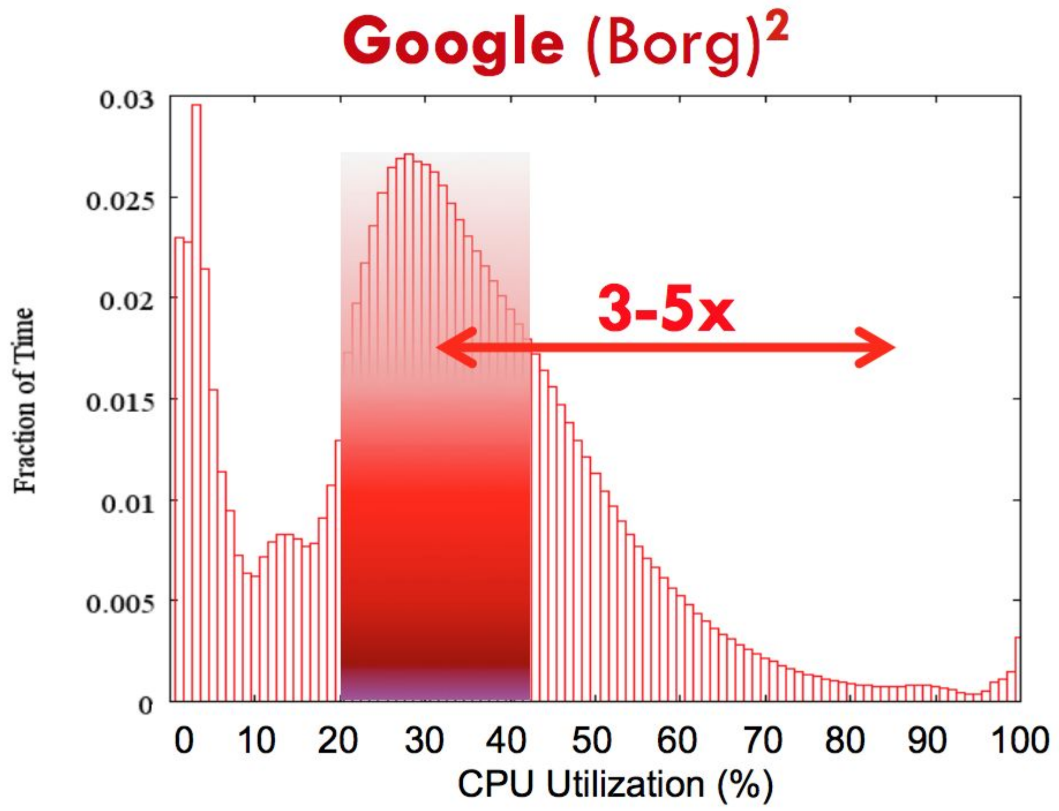
- Resource Utilization in Large Scale Cluster
- Workloads
- Colocation on Kubernetes
- Results

Cluster Scale

- Tens of clusters
  - Tens of thousands of nodes in one cluster
  - Hundreds of GPU nodes in the same cluster with CPU nodes
- Hundreds of thousands of pods
  - Tens of thousands of jobs
- Resource cost is huge

Should We Care About Utilization?





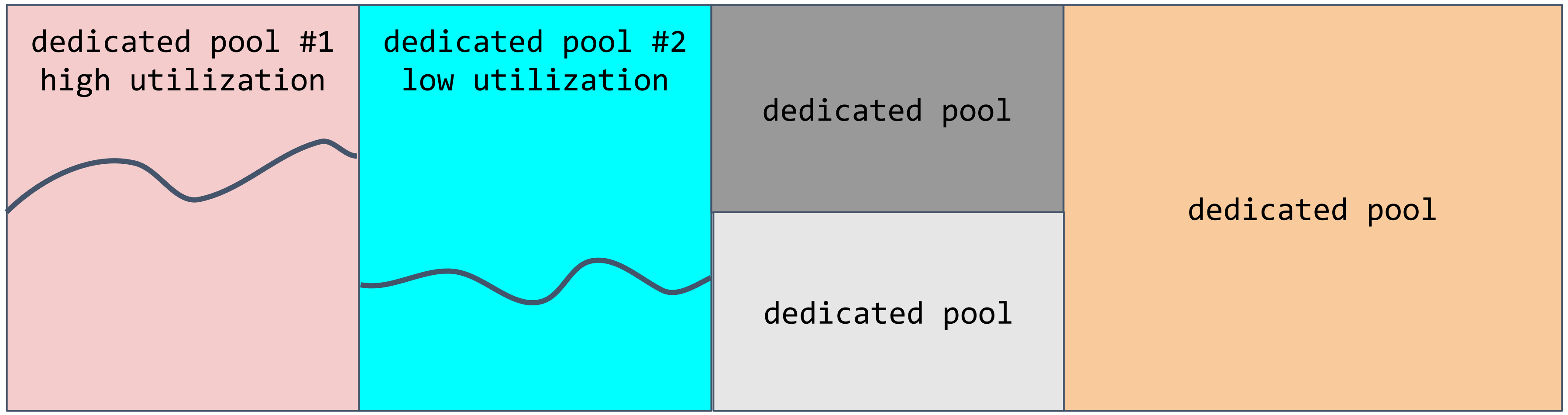
Ref: http://csl.stanford.edu/~christos/publications/2015.christina_delimitrou.phd_thesis.slides.pdf

[1] C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management, ASPLOS 2014.

[2] L. A. Barroso, U. Holzle. The Datacenter as a Computer, 2013.

Why Low Utilization?

- Dedicated nodes for latency sensitive workloads
- Gap between reserved (allocated) and used
- Utilization varies over time
- Nodes are heterogeneous (size, type, performance...)



Increasing Utilization Brings Significant Cost Saving

Workloads

	Long Running Services	**Jobs**
Category	e-commerce website, payment system	Spark, Flink, XGBoost, TensorFlow Training
Latency	Sensitive	Insensitive
Priority	High	Low
Traffic Pattern	Peak during daytime and low during night	Peak when running
Fault Tolerance	Should not fail, high availability	Fail and retry

Workloads - How to increase utilization?

- Overcommit?
  - Uncontrollable overcommit is dangerous
  - Overcommit should follow with reacting to dynamic load changes
  - Some resources are "compressible" (CPU) and some are not (RAM)
    - Container will be killed if they exceed their memory limit

Out of Memory

Workloads - Put them together

- Different workloads need different resource priority level
  - High level resource for services (Production)
  - Low level resource for jobs (Preemptible)
  - Isolation is the key point - node level cgroup

Workloads - Put them together

- Different workloads need different resource priority level
  - Production and Preemptible

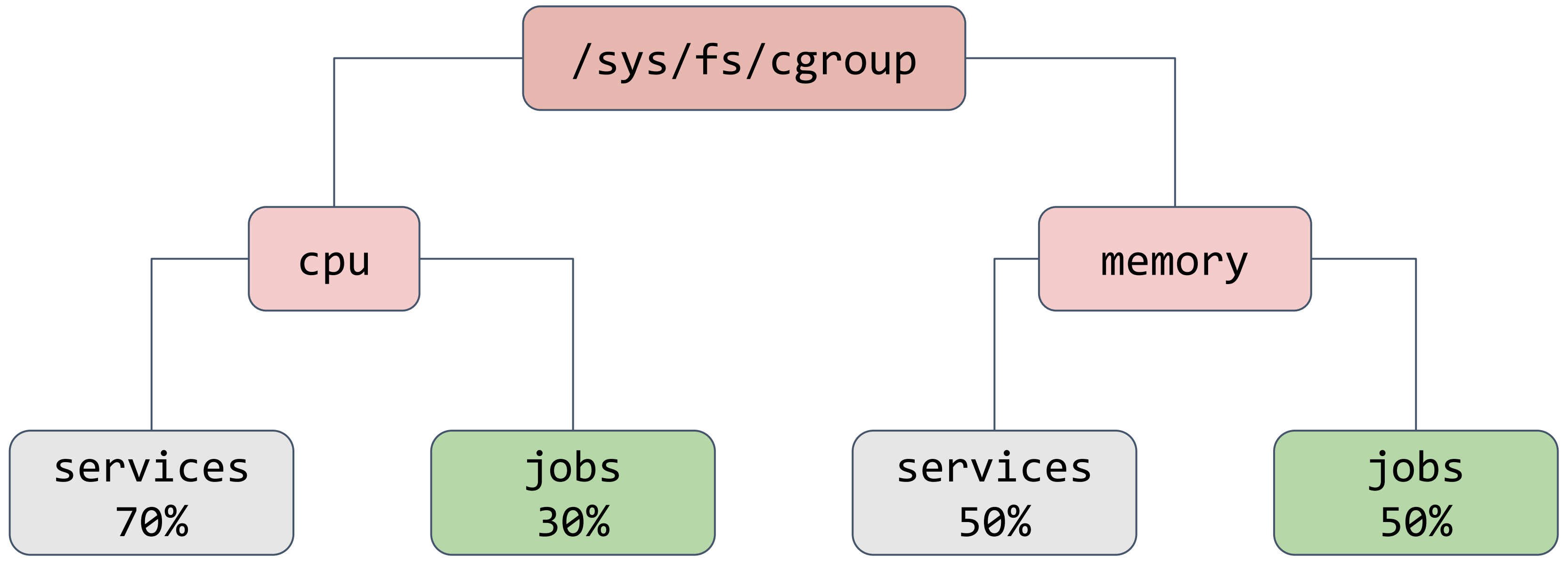
Production	Preemptible
SLO Guaranteed	No SLO
Not Preemptible	Be killed at anytime
High Priority	Low Priority

Workloads - Put them together

- How tasks are isolated from each other?
    - CGroups
        - CPU - shares/quota/cores
        - Memory - memory.limit_in_bytes
        - Disk - blkio.throttle.xxx
        - Network - priority and rate

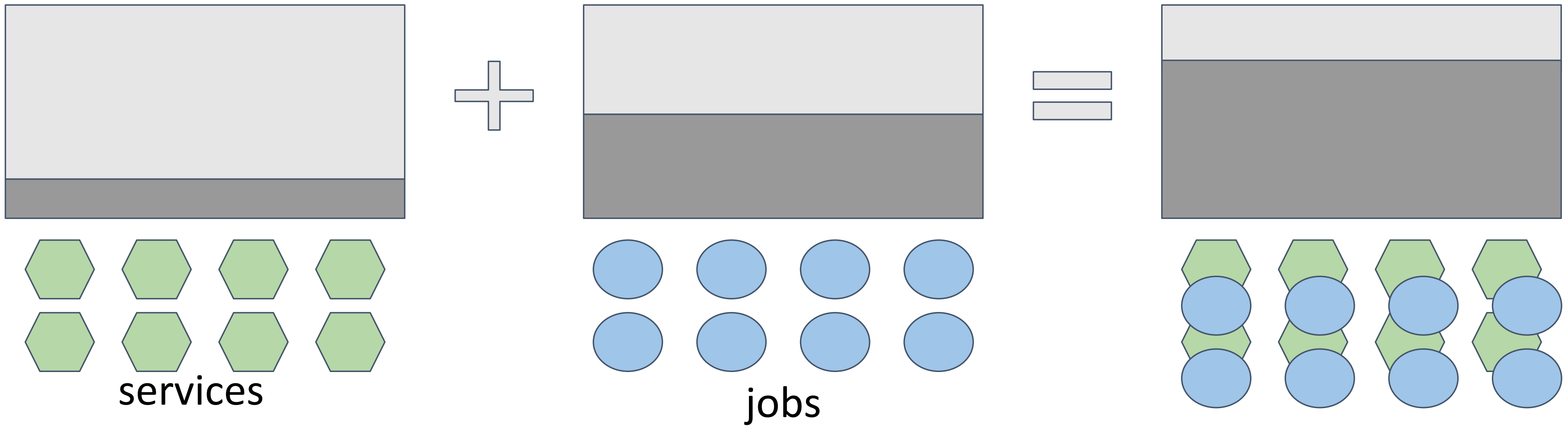
Workloads - Put them together

- CGroup is a good approach
  - Separated node level cgroup for services and jobs
  - Custom defined resource isolation between services and jobs



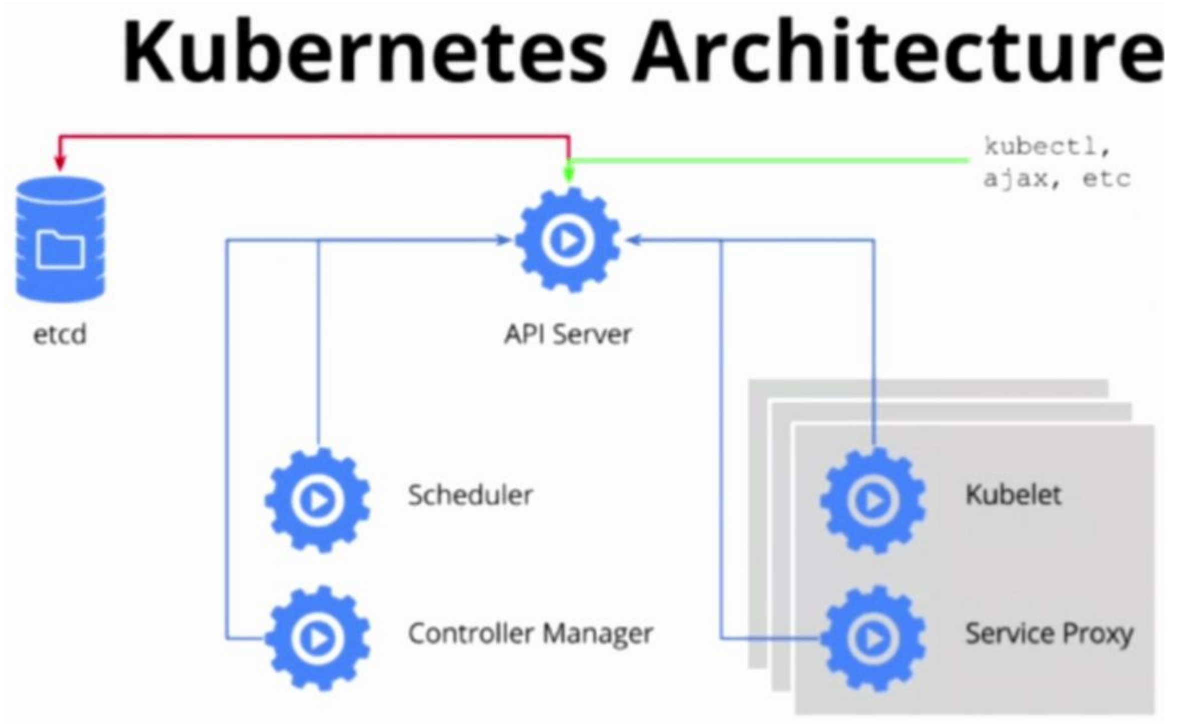
Workloads - Put them together

- Jobs should not impact services
- Services get guaranteed resources and jobs get best effort resources
- Never over commit services resources
- Jobs are not happy if they starved to death



Current Features in Kubernetes

- Kubelet
  - CPU Manager (CFS shares/CFS quota/CPU affinity)
  - Device Manager
- API-Server
  - Admission (mutating)
- Scheduler
  - Extended resource scheduling
- QoS Class



Problems - QoS Class

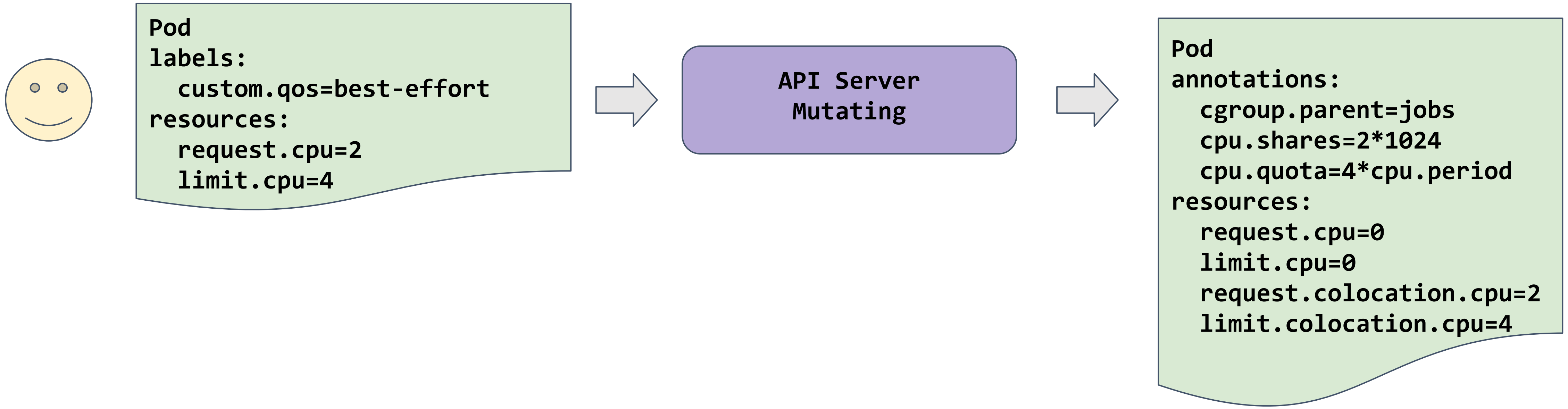
- Implicit QoS with request/limit
  - request & limit is zero means Best Effort
- Problems
  - Rogue best effort pods can take over all resources due to no limit
  - Request is zero meaning scheduler cannot do resource accounting and schedule based on request size
  - Can’t define custom cgroup parameters (cpu shares and quota)

Problems - QoS Class

- Explicit QoS with label
  - `custom.qos=best-effort`
- Solutions
  - Create new resource type "colocation CPU" for jobs (extended resource)
    - Make CPU as infinite resource because it is compressible
    - So we only care about memory/disk resource
  - Auto mutate request.cpu to "colocation CPU"
  - Scheduler do resource accounting based on this extended resource
  - Define custom cgroup parameters in admission mutating

Build Colocation with Native Feature - API Server

- Admission - mutating
  - Change request CPU to "colocation CPU"
  - Set cgroup parameters in pod's annotation

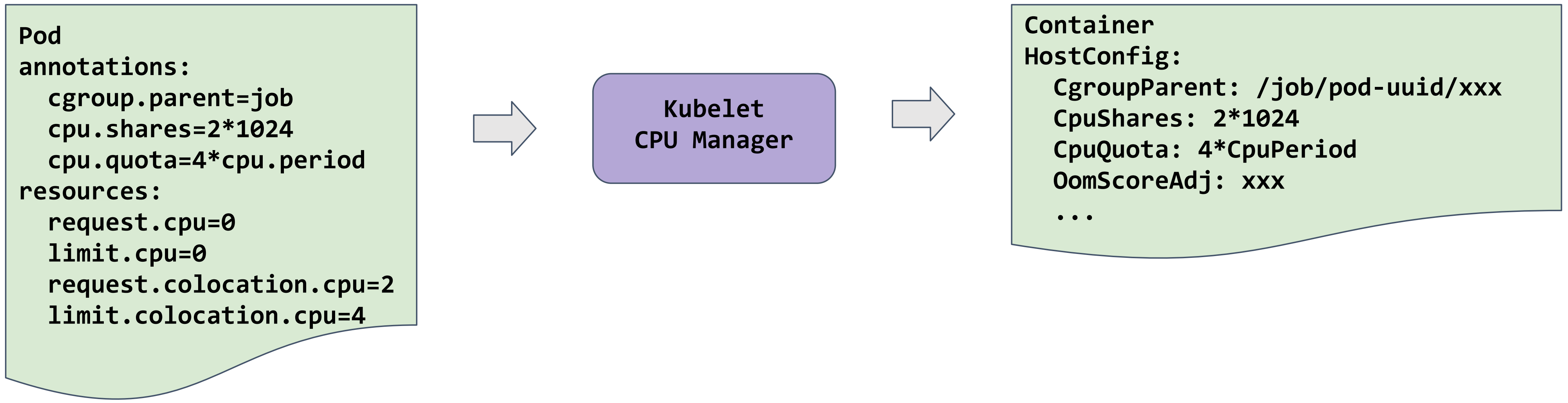


```
Pod
labels:
  custom.qos=best-effort
resources:
  request.cpu=2
  limit.cpu=4
```

```
Pod
annotations:
  cgroup.parent=jobs
  cpu.shares=2*1024
  cpu.quota=4*cpu.period
resources:
  request.cpu=0
  limit.cpu=0
  request.colocation.cpu=2
  limit.colocation.cpu=4
```

Build Colocation with Native Feature - Kubelet

- Extend CPU manager policy
  - Set pod level cgroup by annotation
  - Isolation in many dimensions
    - CPU CFS/memory/blkio/oom score/network priority

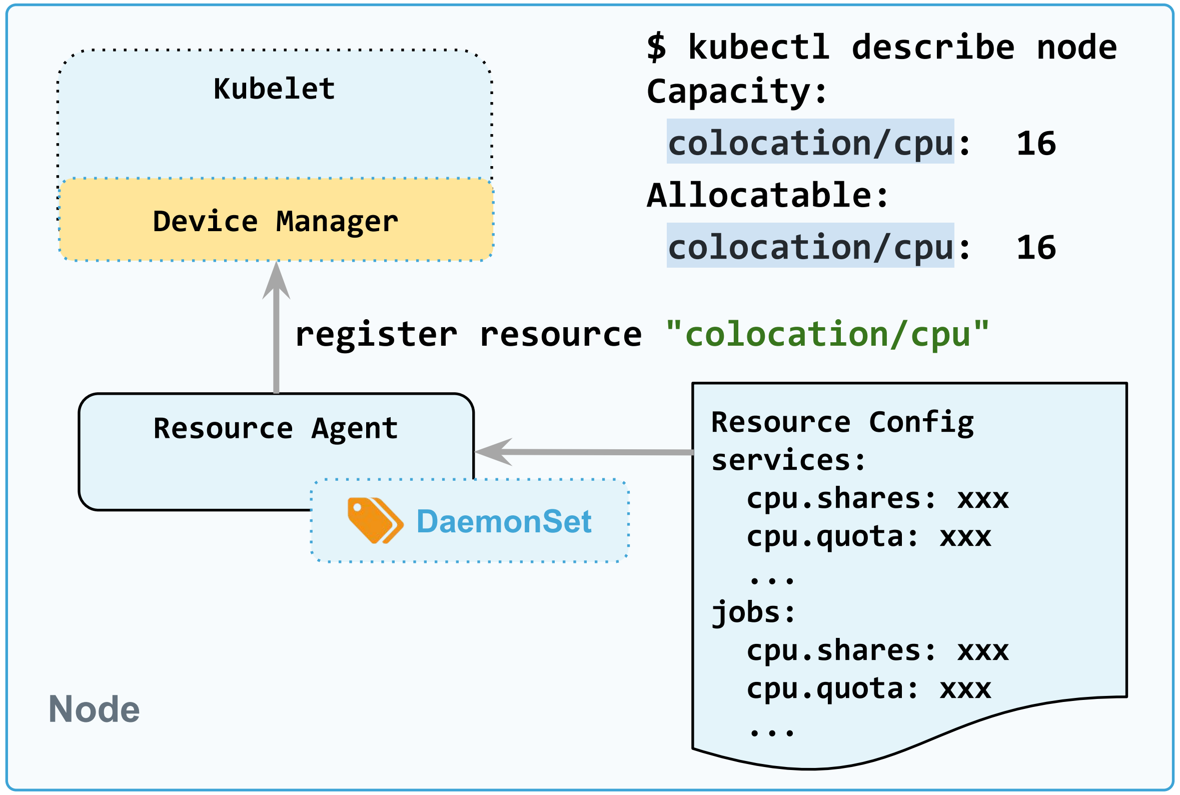


```
Pod
annotations:
  cgroup.parent=job
  cpu.shares=2*1024
  cpu.quota=4*cpu.period
resources:
  request.cpu=0
  limit.cpu=0
  request.colocation.cpu=2
  limit.colocation.cpu=4
```

```
Container
HostConfig:
  CgroupParent: /job/pod-uuid/xxx
  CpuShares: 2*1024
  CpuQuota: 4*CpuPeriod
  OomScoreAdj: xxx
  ...
```

Build Colocation with Native Feature - Kubelet

- Resource Agent
  - Advertise dynamic "colocation CPU" according to node level utilization
  - Set node level cgroup by config
  - DaemonSet
  - Resource name: "colocation/cpu"



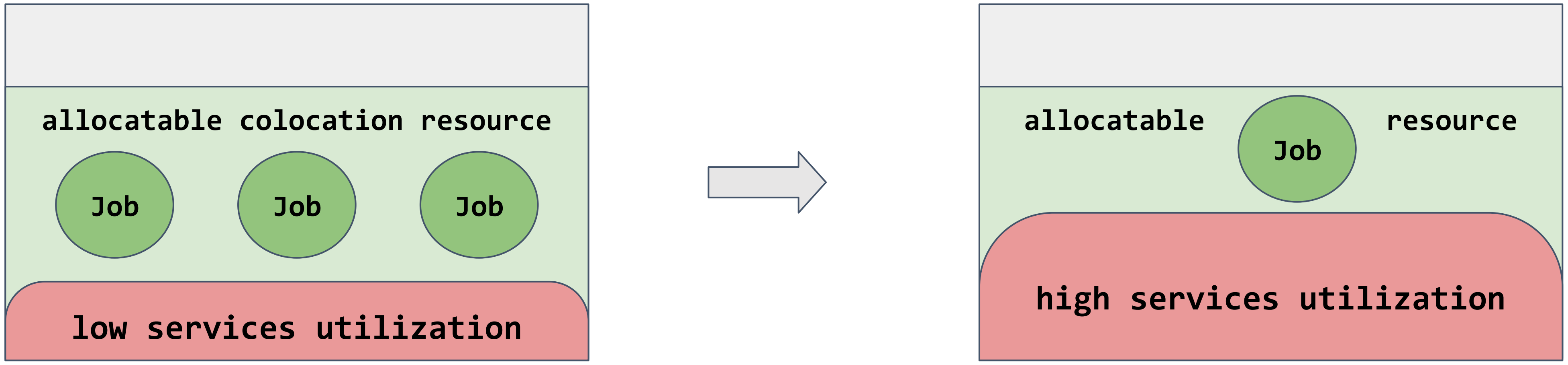
```
$ kubectl describe node
Capacity:
 colocation/cpu:   16
Allocatable:
 colocation/cpu:   16
```

```
Resource Config
services:
  cpu.shares: xxx
  cpu.quota: xxx
  ...
jobs:
  cpu.shares: xxx
  cpu.quota: xxx
  ...
```

Colocation on Kubernetes - Resource Agent

Dynamic colocation resource

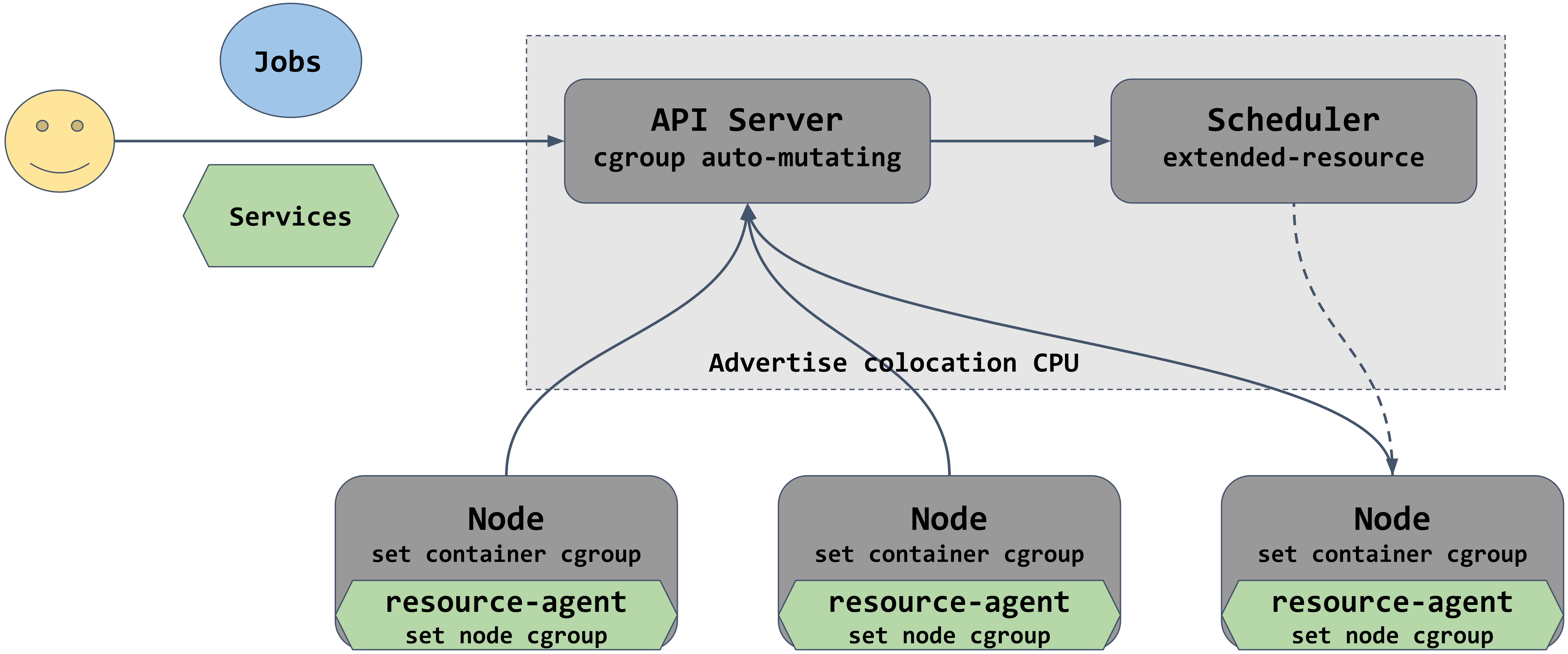
- Resource auto profiling
- More colocation resource means more jobs



More related works

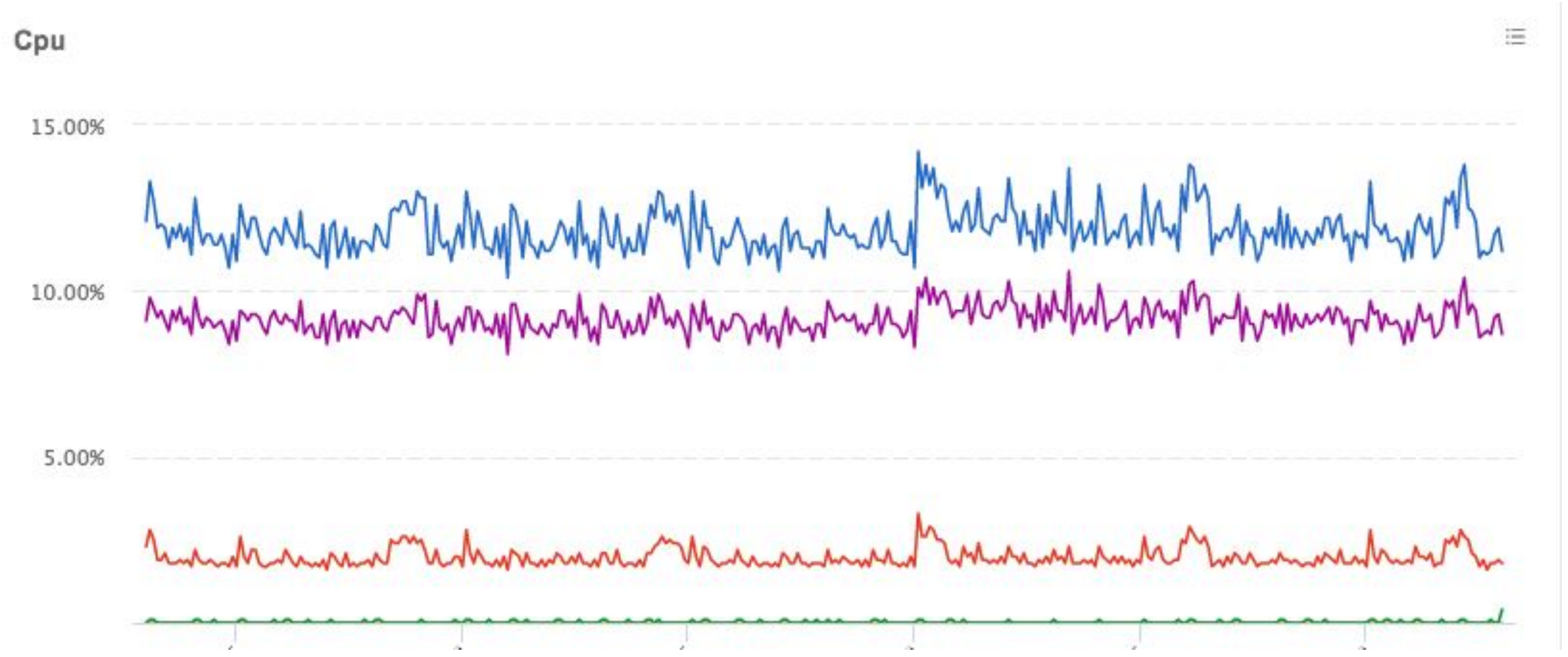
- CRD
  - Quota - cluster level
  - PodGroup - gang scheduling
- Resource Auto-Profiling
  - VPA
  - HPA
- Unified-Scheduler
  - Priority and Preemption

Architecture



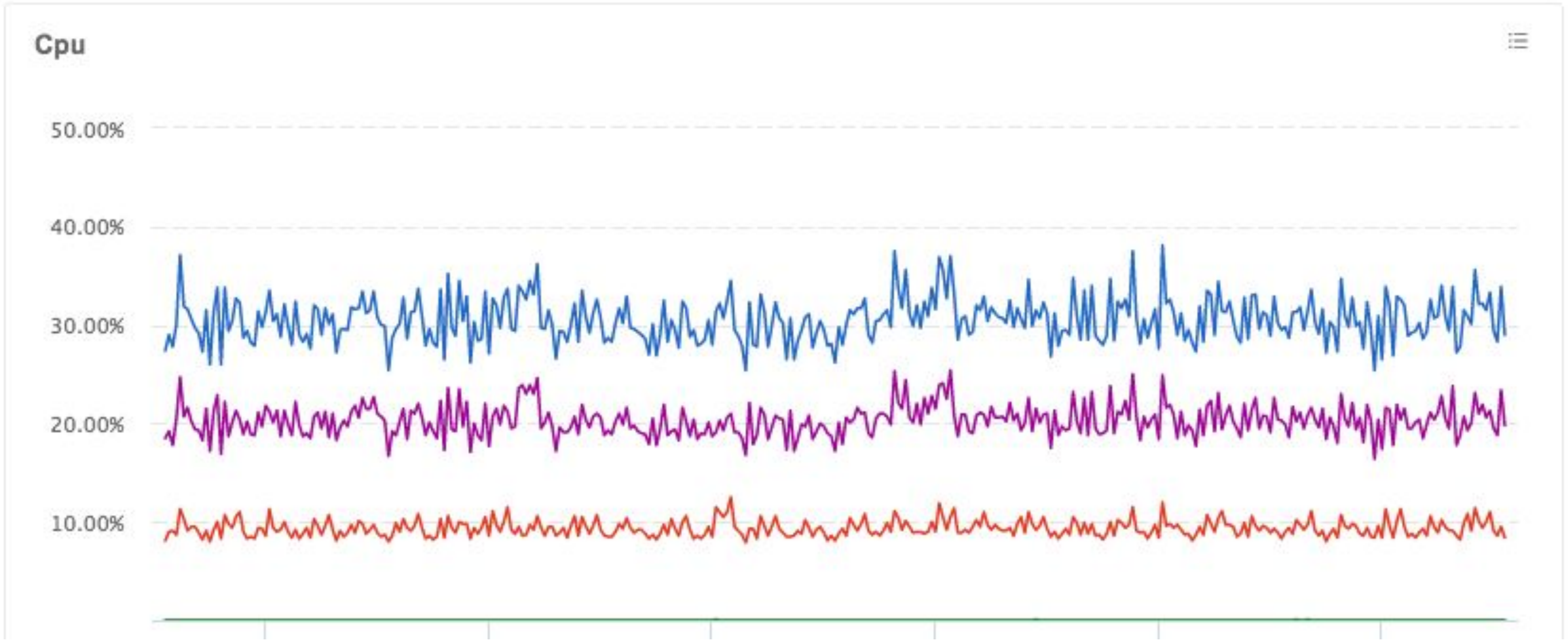
Results - Services

- CPU utilization 10%-15%



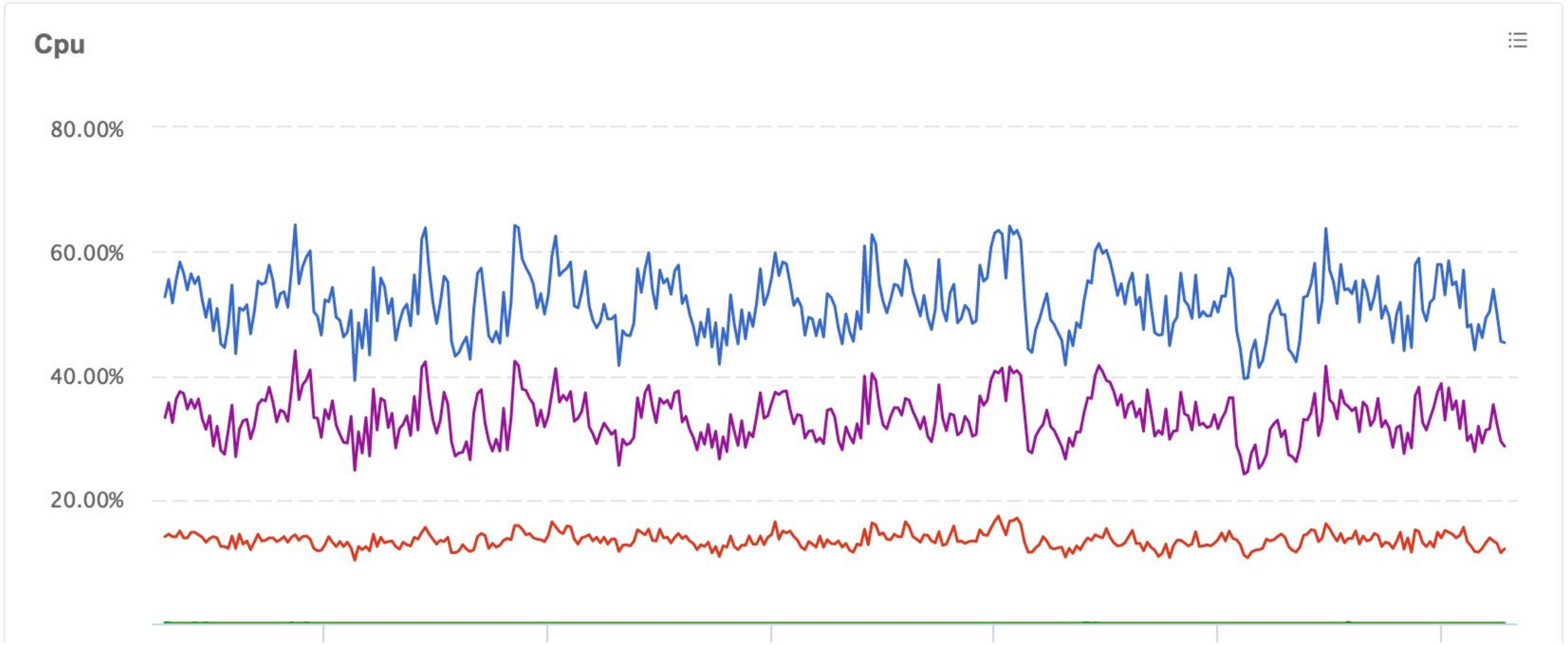
Results - Jobs

- CPU utilization 20%-30%



Results - Services + Jobs

- CPU utilization 35%-50%



OpenKruise - Automate everything!

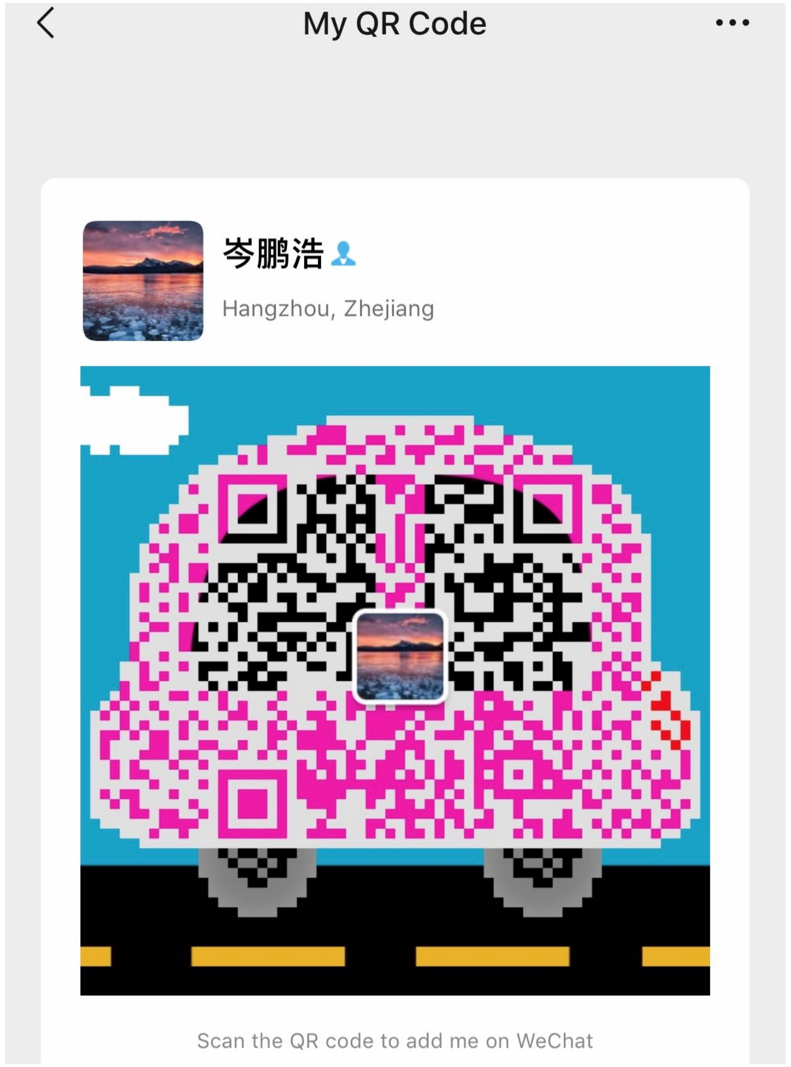
https://github.com/openkruise/kruise

OpenKruise

Automate everything on Kubernetes

Q & A

Thanks


My QR Code

岑鹏浩

Hangzhou, Zhejiang

Scan the QR code to add me on WeChat