Kuberentes 中的资源调度与管理

Da Ma (@k82cn)

Introduction

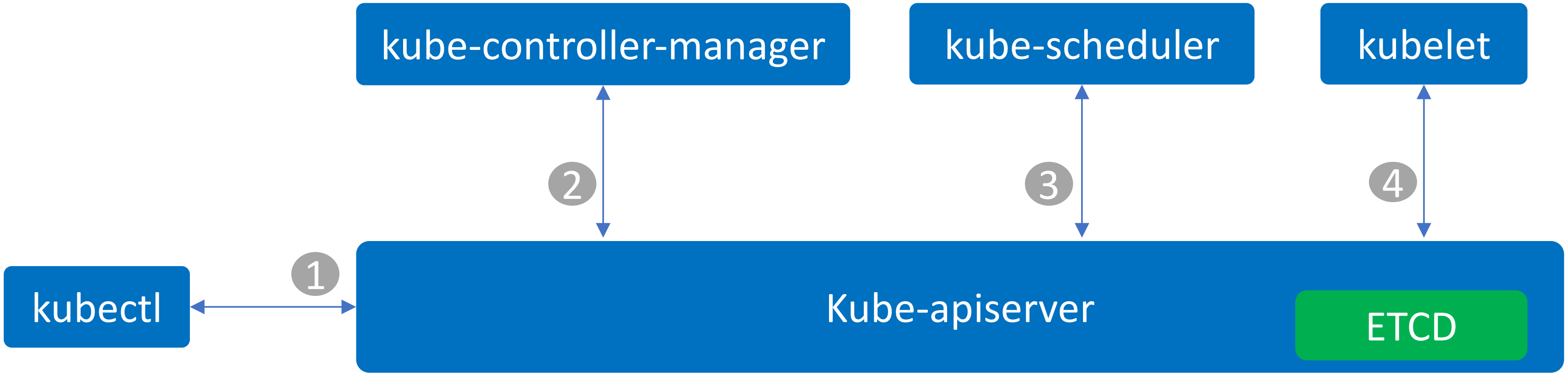
马达.IBM

北京 海淀

Da Ma (Klaus, @k82cn)

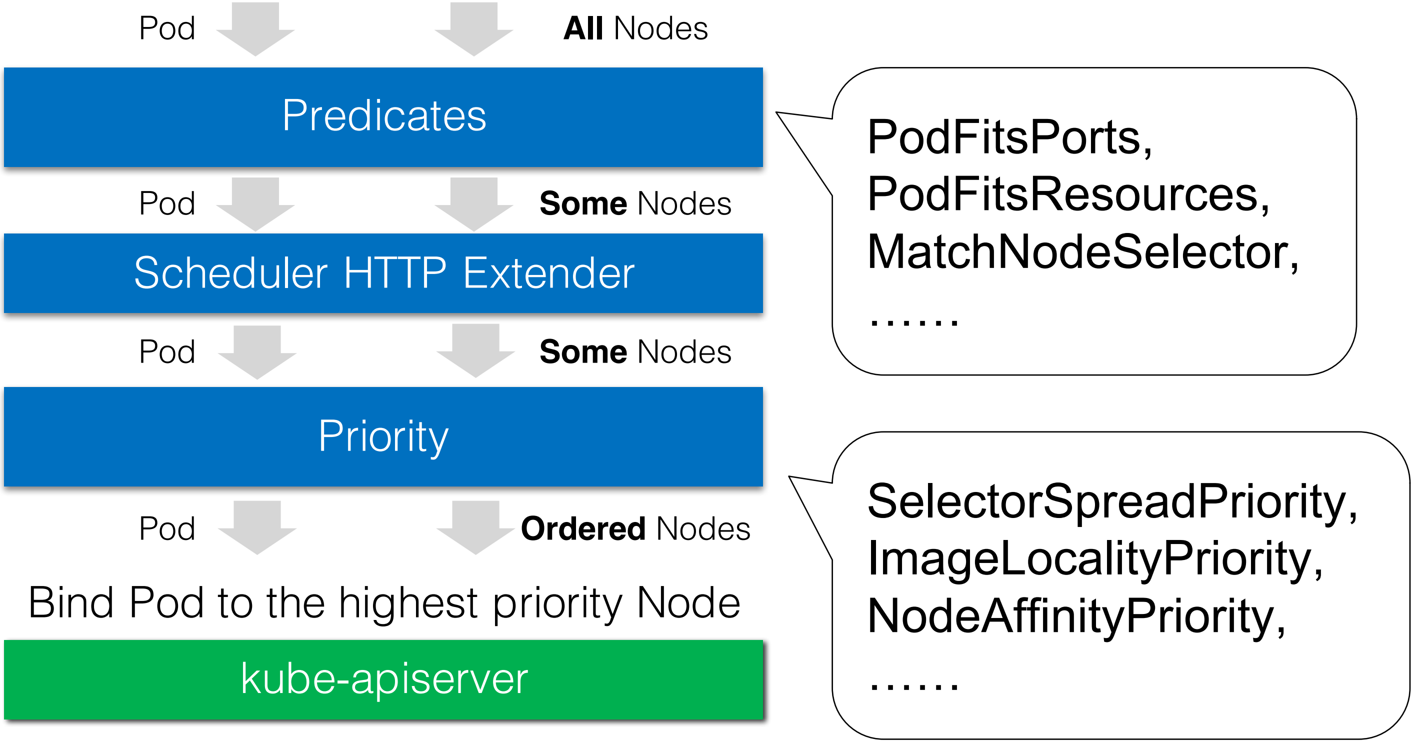
- Distributed Computing since 2005
- Kubernetes Maintainer
- Mesos Contributor

Kubernetes Overview



1. Kubectl creates an object (e.g. RC) in apiserver if all admission passed
2. In kube-controller-manager, related controler create Pods based on its replicas
3. kube-scheduler get the “notification” of Pod from apiserver; kube-scheduler choose one host for the Pod one by one based on its policy
4. kubelet gets the notification of Pod from apiserver; and then start the container

kube-scheduler overview



Pod **All** Nodes

Predicates

Pod **Some** Nodes

Scheduler HTTP Extender

Pod **Some** Nodes

Priority

Pod **Ordered** Nodes

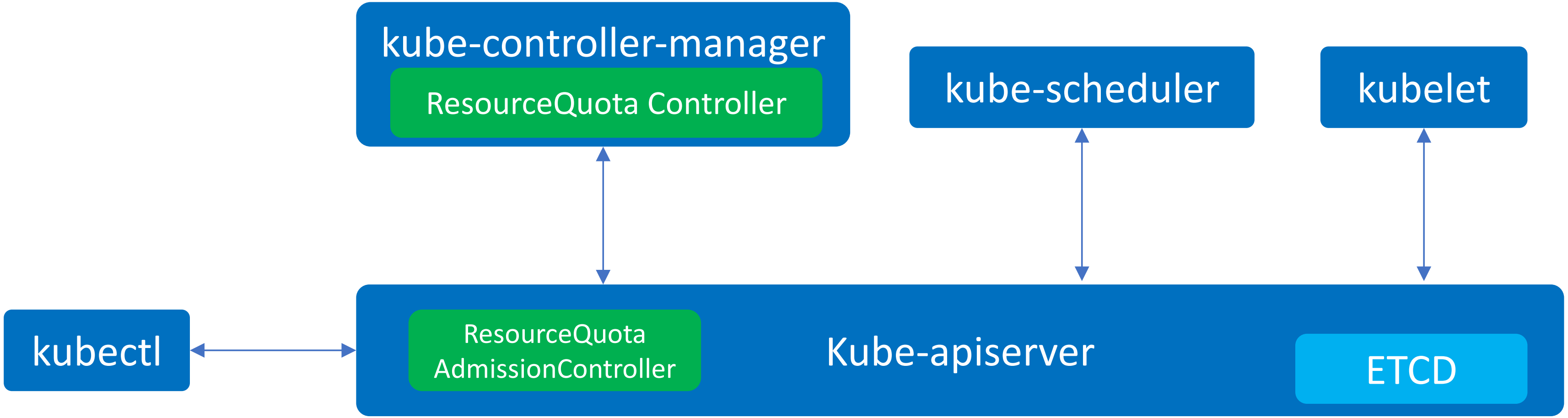
Bind Pod to the highest priority Node

kube-apiserver

PodFitsPorts,
PodFitsResources,
MatchNodeSelector,
……

SelectorSpreadPriority,
ImageLocalityPriority,
NodeAffinityPriority,
……

Resource Quota



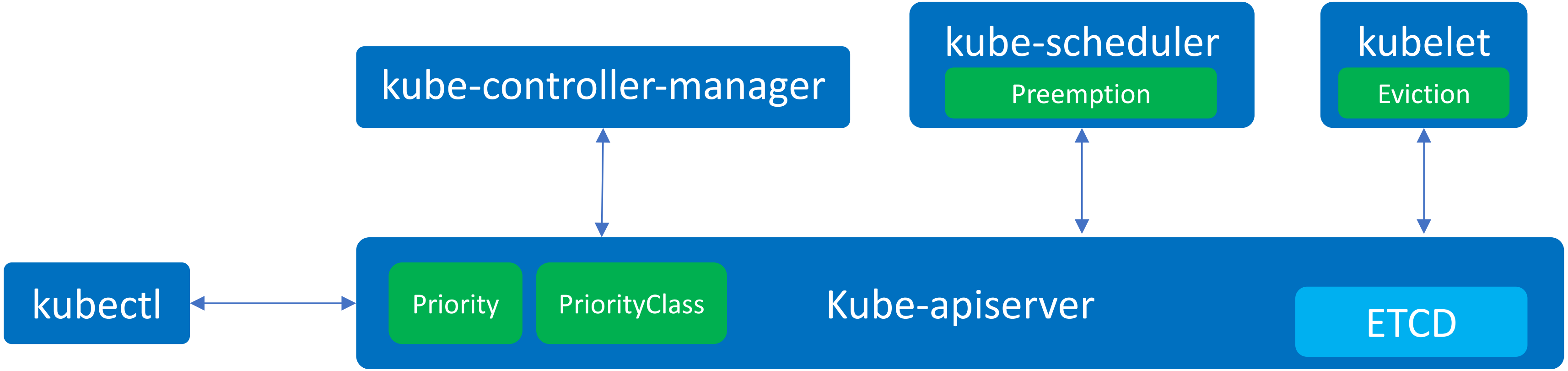
- ResourceQuota AdmissionController reject creation if exceed Quota
- ResourceQuotaController update *Quota.Usage* accordingly
- Static allocation of cluster, no sharing between namespace: namespace resource management, multi-tenants (resource sharing part)
- Defines Pending Pods in cluster for scheduling: job level scheduler

Resource sharing architecture for batch and serving workloads in Kubernetes

@davidopp, @erictune, @foxish, @k82cn

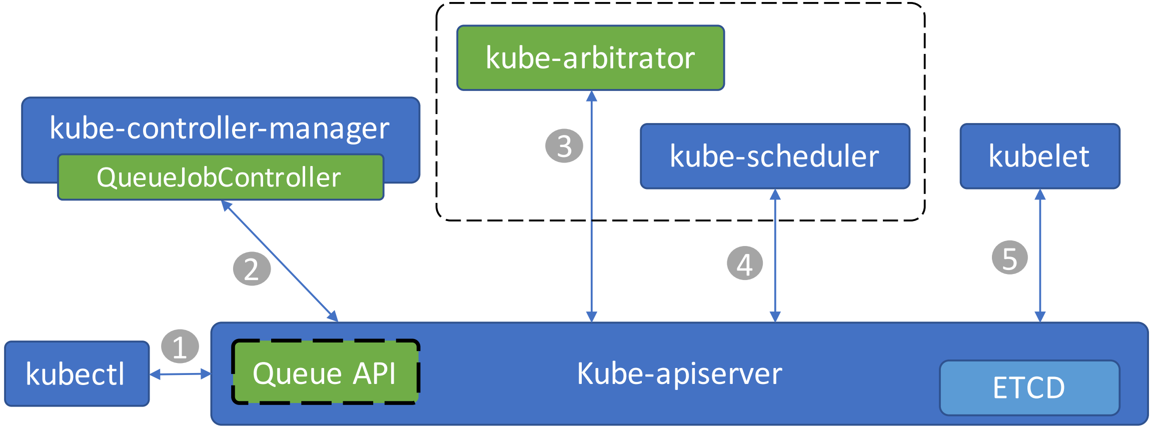
https://docs.google.com/document/d/1-H2hnZap7gQivcSU-9j4ZrJ8wE_WwcfOkTeAGjzUyLA/edit#

Priority/Preemption (online service)



- Priority is an integer, identify by PriorityClass (ProrityClassName)
- kube-scheduler select a victim Pod to preempt (Pod level eviction), PDB is considered
- kube-scheduler will re-use preempted resources
- Kubelet will evict Pods by priority, then by QoS

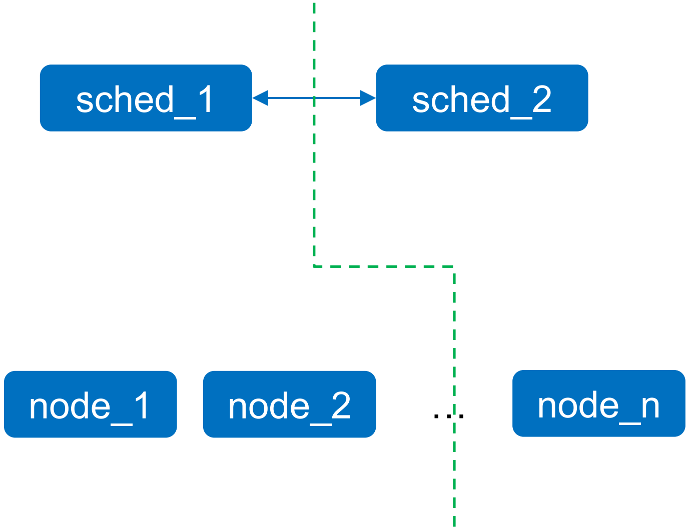
kube-arbitrator (batch job, offline service)



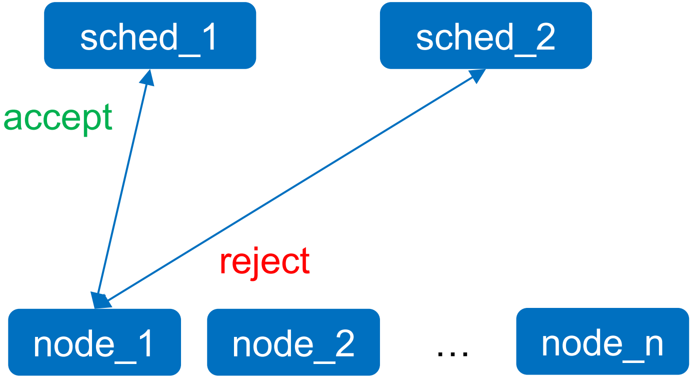
1. Kubectl creates a ***QueueJob*** object in apiserver if all admission passed
2. In kube-controller-manager, ***QueueJobController*** create Pods based on its replicas
3. kube-arbitrator get the “notification” of Pod from apiserver
4. kube-arbitrator choose one host for the Pod of ***QueueJob*** based on its policy
5. kubelet gets the notification of Pod from apiserver; and then start the container

1. The policy in kube-arbitrator is pluggable, DRF by default
2. kube-arbitrator handles **batch job**, kube-scheduler handles **long-running workload** (right now; by multi-scheduler feature)
3. The first framework will be **Tensorflow**, and then other framework

Multi-Scheduler

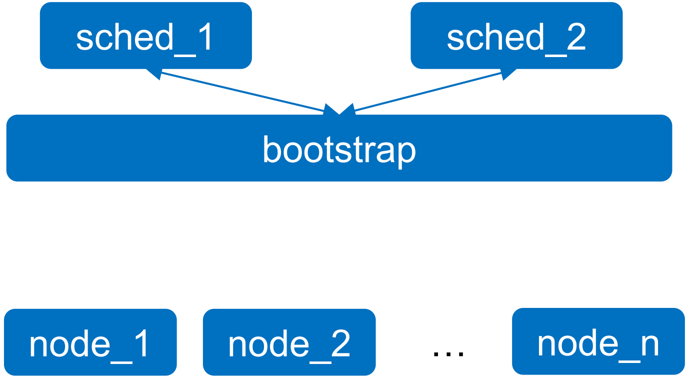


Option 1

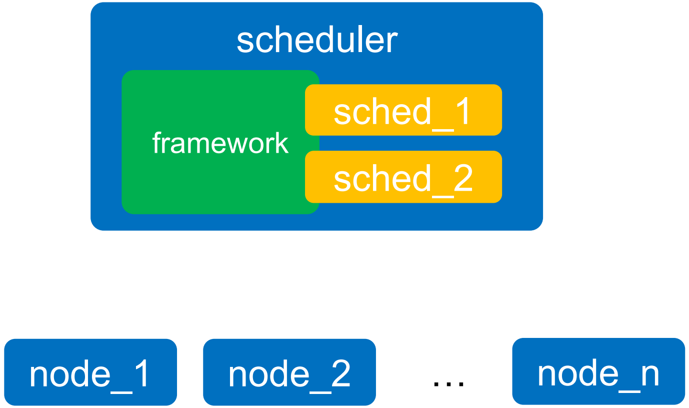


Option 2

Multi-Scheduler



Option 3



Option 4

descheduler

As Kubernetes clusters are very dynamic and their state change over time, there may be desired to move already running pods to some other nodes for various reasons:

- Some nodes are under or over utilized.
- The original scheduling decision does not hold true any more, as taints or labels are added to or removed from nodes, pod/node affinity requirements are not satisfied any more.
- Some nodes failed and their pods moved to other nodes.
- New nodes are added to clusters.

Policy and Strategies:

- RemoveDuplicates
- LowNodeUtilization
- RemovePodsViolatingInterPodAntiAffinity

Long Running Service

or

Online Service

On-going

- Priority/Preemption to beta
- Schedule DaemonSet Pods in kube-scheduler
  - Multi-Scheduler solution
  - Policy lib, modularity
- Scheduler as a Framework
  - SDK
  - gRPC extenders
- Performance Enhancement

Join community

SIG Scheduling

- Meeting on-demand
- #sig-scheduling Slack channel
- kubernetes-sig-scheduling@ mailing group

☺ Thanks