Enhancing Kubernetes Scheduler for Diverse Workloads in Large Clusters

Yuan Chen, Yan Xu @Apple

Agenda

- Introduction
- Case studies
  - Scheduling for stateful apps.
  - Gang scheduling for batch jobs
  - Scalable scheduling in large clusters
- Summary

Vanilla Scheduler is Becoming Insufficient

K8s Scheduler

- Stateless applications
- Pod by pod scheduling
- Simple scheduling logic
- “Optimal” strategy

Diverse workloads in large clusters

- Stateful apps, batch jobs, ML/DL, HPC
- Advanced scheduling
- Scalable scheduling
- Custom scheduling

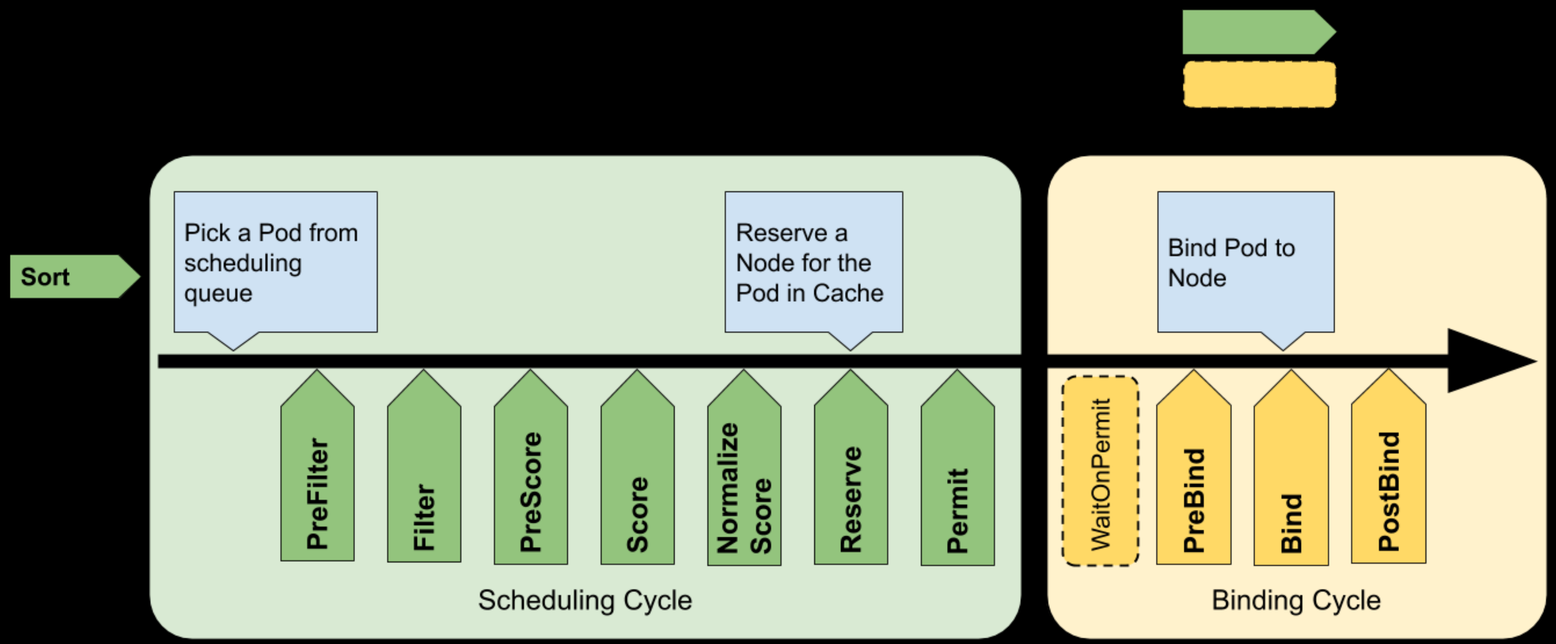
Emerging Scheduling Requirements

- Support for stateful apps
- Gang scheduling
- Custom preemption
- Topology-awareness
- GPU bin-packing
- Resource interference/contention-aware
- Scalable scheduling
- Adaptive scheduling

...

Scheduling Framework

"Provides a uniform and configurable mechanism and APIs for extending the default scheduler through multiple extension points. The default scheduler and all plugins are compiled into a single scheduler, which allows many new and custom scheduling features to be implemented while keeping the scheduling "core" simple and maintainable."



- **Highly extensible and customizable**
- **Better performance and scalability**
- **Better handle errors**
- **No conflicts and race conditions**

Comparison of Different Methods

Extending the Vanilla scheduler

Methods	Implemenation	Extension Points	Compatibility	Overhead	Conflict	Performance
Ad-hoc	Modify the scheduler code	None	Low	Medium	No	High
Scheduler Extenders	A single scheduler with webhook extensions	Very limited: PostFilter, PostScore	High	Medium	No	Low
Multiple Schedulers	Multiple independent schedulers	Separate schedulers	Medium	High	Yes	Medium
Scheduling Framework	A single scheduler with lightweight plugins	**Before and after each stage of a scheduling cycle**	High	Low	No	High

Agenda

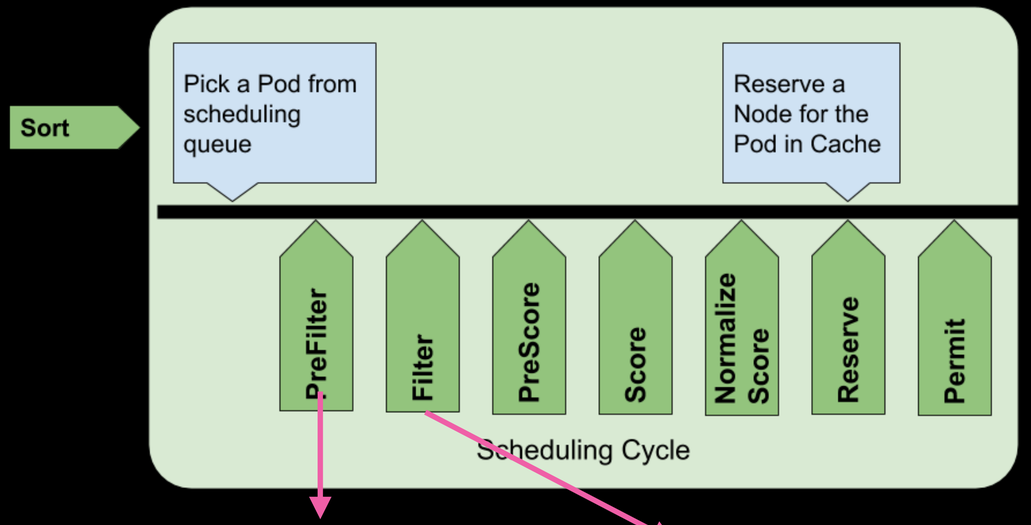
- Introduction
- Case studies
  - Scheduling for stateful apps.
  - Gang scheduling for batch jobs
  - Scalable scheduling in large clusters
- Summary

Case Study
Scheduling for StatefulPod

- Fixed IP address for a Stateful Pod
- Scheduling
  - Track IP information
  - Check IP availability on a Node or Rack
  - Reschedule the Pod on the same Node or Rack with the assigned IP address

StaticIP Scheduler Plugin

Custom PreFilter and Filter plugins



Sync up the IP reservation information and available IP count with the Node/Pod informer.

Check if the Pod is a stateful Pod and has an IP reservation or not.

For a regular Pod or a stateful Pod without reservation, check if the node has free IP addresses available.

For a stateful Pod with an IP reservation, check if the node has the reservation.

```go
// Filter is a plugin that checks if the candidate Node has an IP
// reservation or free IPs for the Pod to be scheduled.
func (ss *staticIPScheduling) Filter(ctx context.Context, _ *framework.
CycleState, pod *v1.Pod, nodeInfo *nodeinfo.NodeInfo) *framework.Status
 {
    si := ss.schedulerInfo
    nodeName := nodeInfo.Node().Name
    reservedNodeName, found := si.reservations[podToKey(pod)]

    // If the pod is a stateful pod with an IP reservation.
    if found {
        if reservedNodeName != nodeName {
            return framework.NewStatus(framework.UnschedulableAndUnreso
lvable, reservationFailureReason)
        }
        return framework.NewStatus(framework.Success, "")
    }

    // If the pod is a regular pod or a new stateful pod without reserv
ation.
    if si.ipsFree[nodeName].FreeIPs <= 0 {
        return framework.NewStatus(framework.UnschedulableAndUnresolvab
le, ipFailureReason)
    }
    return framework.NewStatus(framework.Success, "")
}
```

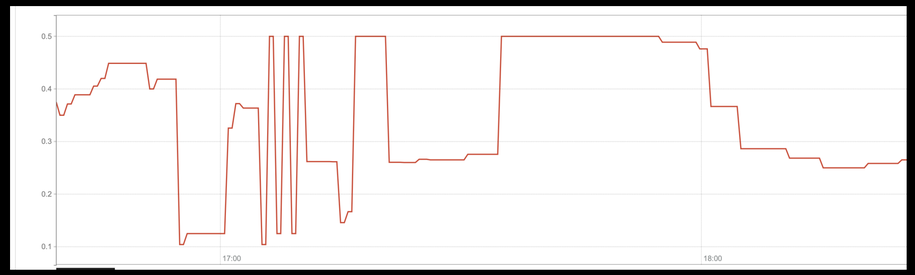
Compared with Scheduler Webhook Extenders

- Simplified implementation
- More robust and stable
- Easier to maintain and manage
- Better performance

Microbenchmark Performance Results

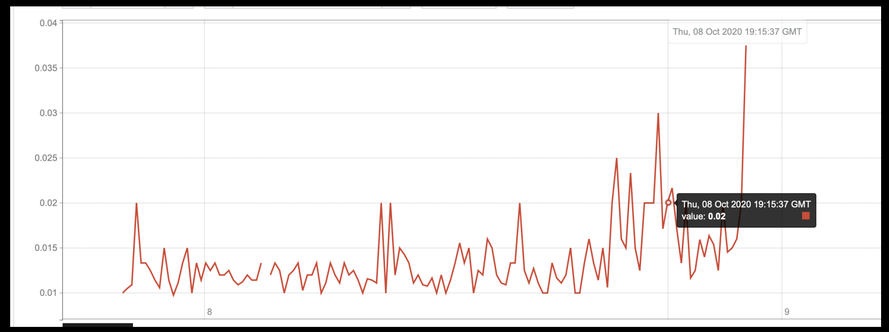
Up to 50% of scheduling algorithm duration

Predicate Extender



Up to 4% of the scheduling algorithm duration

FilterPlugin



Case Study

Gang scheduling for batch jobs

- All or nothing scheduling
- Important for batch applications
  - Big data, e.g., Spark
  - Machine learning/deep learning

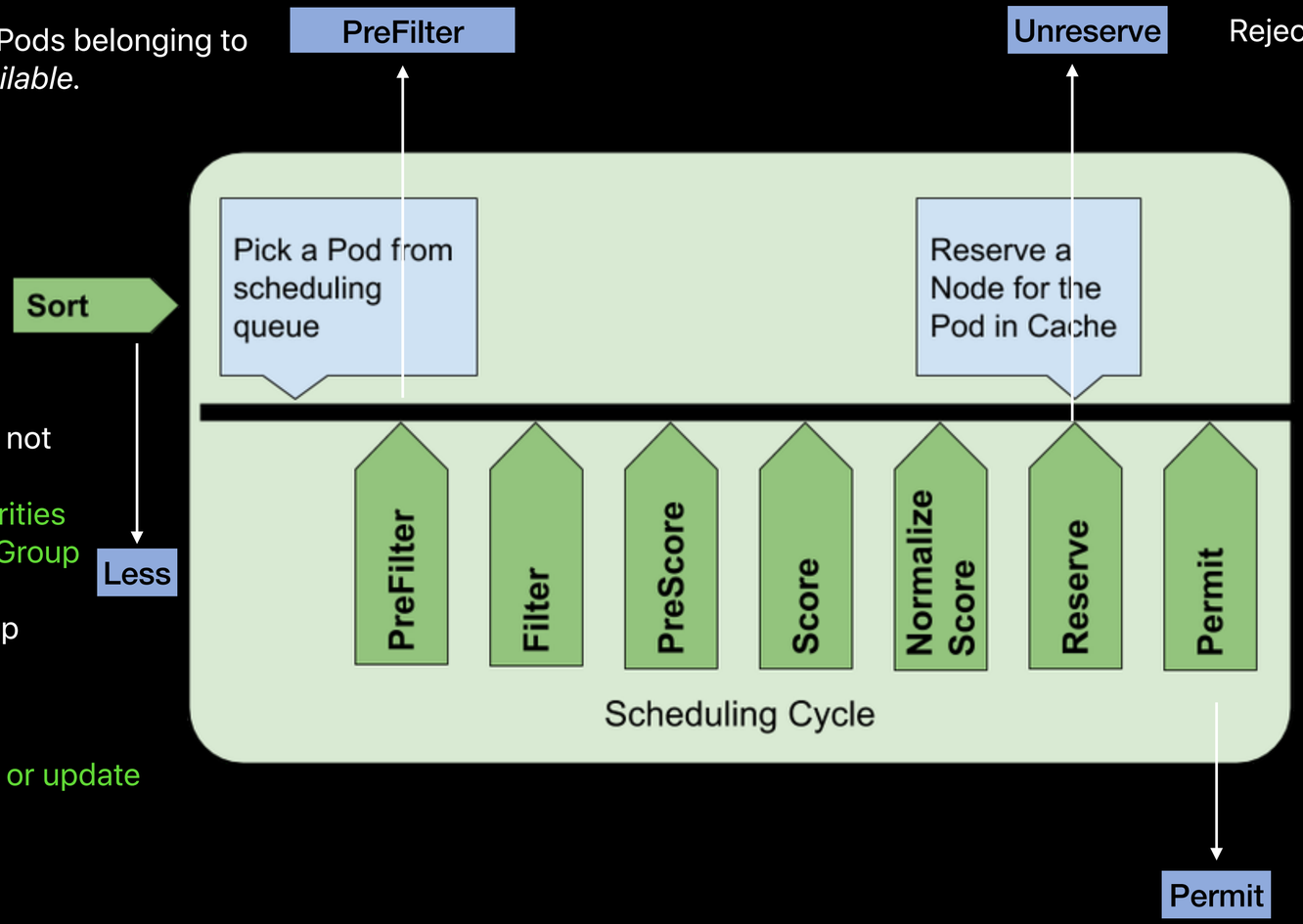
A lightweight coscheduling plugin

- Proposed by Qingcan Wang, et. al.
- Community collaboration: contributors from Alibaba, Apple, IBM, Tencent ...

Lightweight Coscheduling Plugins

labels:
pod-group.scheduling.sigs.k8s.io/name: my-batch-job
pod-group.scheduling.sigs.k8s.io/min-available: 10

1. Create or update a PodGroup if not present.
2. Validate minAvailable and priorities of all the pods in the same PodGroup
3. Validate if the total number of Pods belonging to the PodGroup reaches *minAvailable*.



1. Create or update a PodGroup if not present.
2. Validate minAvailables and priorities of all the pods in the same PodGroup
3. Validate if the total number of Pods belonging to the PodGroup reaches minAvailable.

PodInformer: PodGroup cleanup or update

- Deletion
- Timeout in Waiting state
- Custom parameters

1. If the PodGroup meets minAvailable, allow the Pod and all waiting Pods of the PodGroup
2. Otherwise, put the Pod in Waiting state

Coscheduling Plugins

- Simple
- Support across jobs/deployments
- PodGroup update and cleanup
  - Monitor and update PodGroup status
  - Periodic cleanup
  - Customizable parameters
- Error check

https://github.com/kubernetes-sigs/scheduler-plugins/tree/master/pkg/coscheduling

```yaml
apiVersion: kubescheduler.config.k8s.io/v1beta1
kind: KubeSchedulerConfiguration
leaderElection:
  leaderElect: false
clientConnection:
  kubeconfig: "REPLACE_ME_WITH_KUBE_CONFIG_PATH"
profiles:
- schedulerName: default-scheduler
  plugins:
    queueSort:
      enabled:
        - name: Coscheduling
      disabled:
        - name: "*"
    preFilter:
      enabled:
        - name: Coscheduling
    permit:
      enabled:
        - name: Coscheduling
    reserve:
      enabled:
        - name: Coscheduling
# optional plugin configs
  pluginConfig:
  - name: Coscheduling
    args:
      permitWaitingTimeSeconds: 10
      podGroupGCIntervalSeconds: 30
      podGroupExpirationTimeSeconds: 600
```

Coscheduling: Ongoing Work and Next Steps

- Coscheduling based on PodGroup CRD
- Custom preemption
- Reservation with backfill
- Rescheduling
- Generic sorting plugin

Refs

1. *https://github.com/kubernetes-sigs/scheduler-plugins/tree/master/kep/42-podgroup-coscheduling*
2. *https://github.com/kubernetes-sigs/scheduler-plugins/issues/13*

Case Study
Scalable scheduling

- Large clusters
- Large jobs or services
- Auto-scaling

Performance limited by

- Pod by pod scheduling
- Choose the “optimal” placement

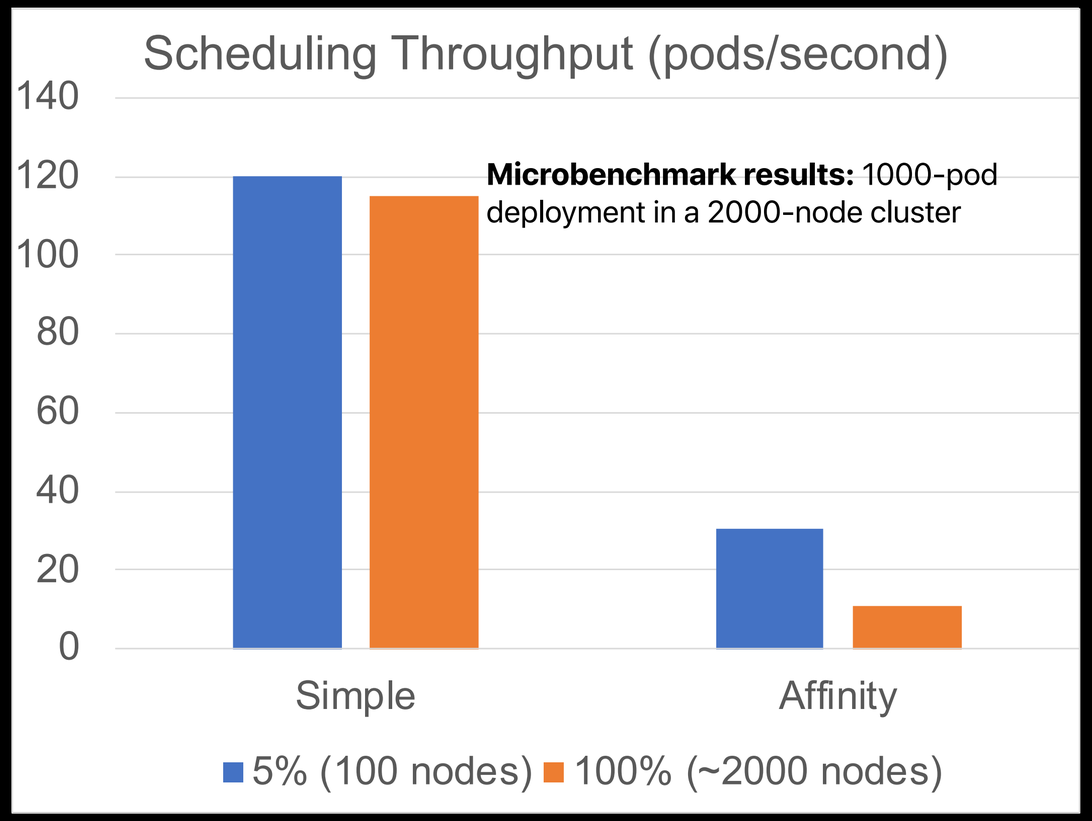
Proposals

- Custom parameters: balancing the scheduling quality and performance
- Group scoring: reuse scoring results and score a group of pods at a time

Custom Scheduler Parameters

percentageOfNodesToScore

Impact on scheduling performance



Proposal for K8s 1.20: per-profile parameter

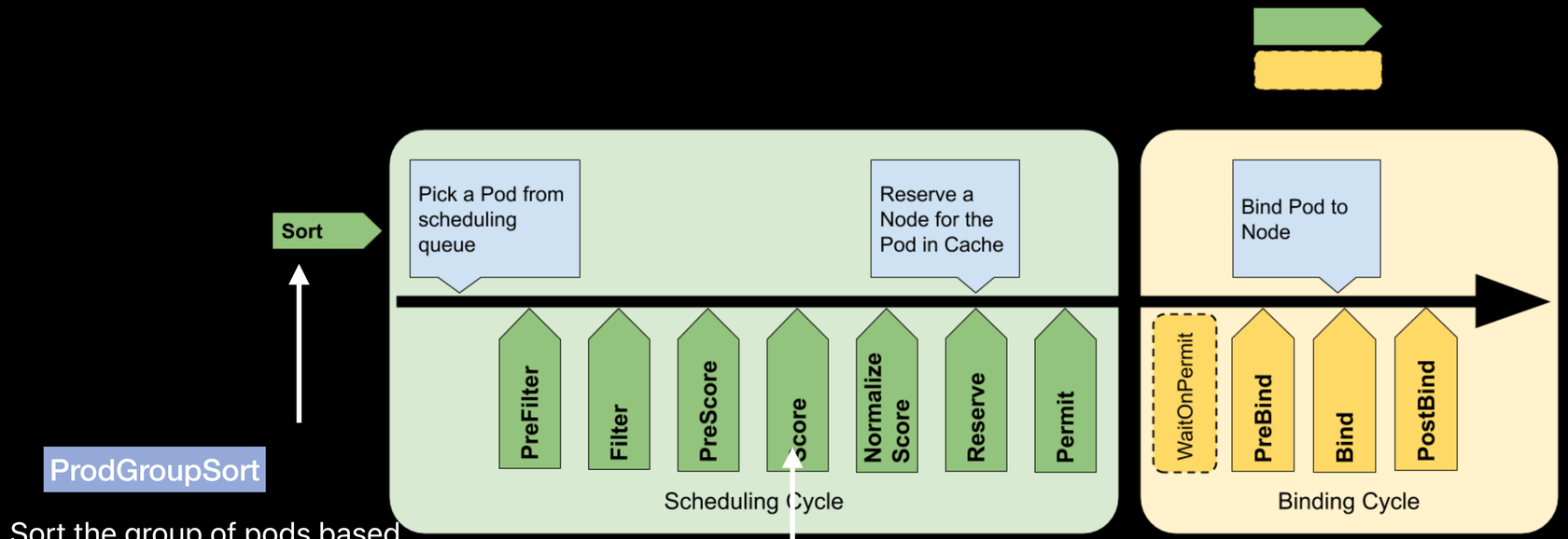
```
ApiVersion: kubescheduler.config.k8s.io/v1alpha2
kind: KubeSchedulerConfiguration
...

profiles
- schedulerName: batch-scheduler
   plugins:
   ...
   percentageOfNodesToScore: 10

- schedulerName: service-scheduler
   plugins:
   ...
   percentageOfNodesToScore: 50
```

- https://github.com/kubernetes/kubernetes/issues/93270
- https://github.com/kubernetes/kubernetes/pull/95823

Group Scoring

Score a group of pods with identical resource requirements at the same time.

- Rank feasible Nodes against a single Pod.
- Assign top k scoring Nodes to k Pods.



Scalable Scheduling Challenges

- Pod group as a scheduling unit
- Customize the scheduling flow logic
  - Filter: shortcut (e.g., the power of two choices)
  - Score: reuse the previous scoring results
- Tradeoff between simplicity and performance/features

Custom Scheduling with Multiple Profiles and Plugins

An example policy file

```yaml
profiles:
- schedulerName: default-scheduler
  plugins:
    queueSort:
      enabled:
        - name: Coscheduling
      disabled:
        - name: "*"

- schedulerName: stateful-scheduler
  plugins:
    queueSort:
      enabled:
        - name: Coscheduling
      disabled:
        - name: "*"
    preFilter:
      enabled:
        - name: StaticIPScheduling
    filter:
      enabled:
        - name: StaticIPScheduling
```

```yaml
- schedulerName: gang-scheduler
  plugins:
    queueSort:
      enabled:
        - name: Coscheduling
      disabled:
        - name: "*"
    preFilter:
      enabled:
        - name: Coscheduling
    permit:
      enabled:
        - name: Coscheduling
    unreserve:
      enabled:
        - name: Coscheduling
  pluginConfig:
    - name: CoScheduling
      args:
        permitWaitingTimeSeconds: 10
        podGroupGCIntervalSeconds: 30
        podGroupExpirationTimeSeconds: 600
  percentageOfNodesToScore: 10
```

```yaml
- schedulerName: integrated-scheduler
  plugins:
    queueSort:
      enabled:
        - name: Coscheduling
      disabled:
        - name: "*"
    preFilter:
      enabled:
        - name: Coscheduling
        - name: StaticIPScheduling
    filter:
      enabled:
        - name: StaticIPScheduling
    permit:
      enabled:
        - name: Coscheduling
    unreserve:
      enabled:
        - name: Coscheduling
  percentageOfNodesToScore: 50
```

Agenda

- Introduction
- Case studies
  - Scheduling for stateful apps.
  - Gang scheduling for batch jobs
  - Scalable scheduling in large clusters
- Summary

Summary

- There are increasing needs for improved and new features in Kubernetes scheduler.
- Kubernetes scheduling framework provides a flexible mechanism for developing new scheduling features.
- New scheduling features are becoming available and under development.
- **Community collaboration is the key**!

Scheduler-plugins: *https://github.com/kubernetes-sigs/scheduler-plugins*

Acknowledgements

- Wei Huang (IBM)
- Qingcan Wang (Alibaba)
- Kai Zhang (Alibaba)
- Abdullah Gharaibeh (Google)
- Weidong Cai (Tencent)

Apple Booth

cncf@group.apple.com

We’re Hiring