
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Unleash the Full Potential of Kubernetes Scheduler: Configuration, Extension and Operation in Production

Yuan Chen, Wei Huang, Yibo Zhuang | Apple

Chen Wang | IBM

About Us

KubeCon | CloudNativeCon North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Yuan Chen
Software Engineer
Apple
Interest: Kubernetes Scheduling, Scalability, Observability
- GitHub: yuanchen8911
- LinkedIn: yuanchen
- Twitter: YuanChenByte
- WeChat: yuan_gt
- Email: yuanchen97@gmail.com

Wei Huang
Software Engineer
Apple

Yibo Zhuang
Software Engineer
Apple

Chen Wang
Staff Research Scientist
IBM Research
Interest: Kubernetes Scheduler Plugins, Kubernetes Autoscalers
- GitHub: wangchen615
- LinkedIn: chenw615
- Twitter: wangchen615
- Email: chen.Wang1@ibm.com

Agenda

- Kubernetes scheduling overview (Wei Huang)
- Scheduler configuration (Yibo Zhuang)
- Scheduler operation (Yuan Chen)
- Scheduler extensions (Wei Huang/Chen Wang)
- Q/A

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Introduction and Scheduling Overview

Wei Huang

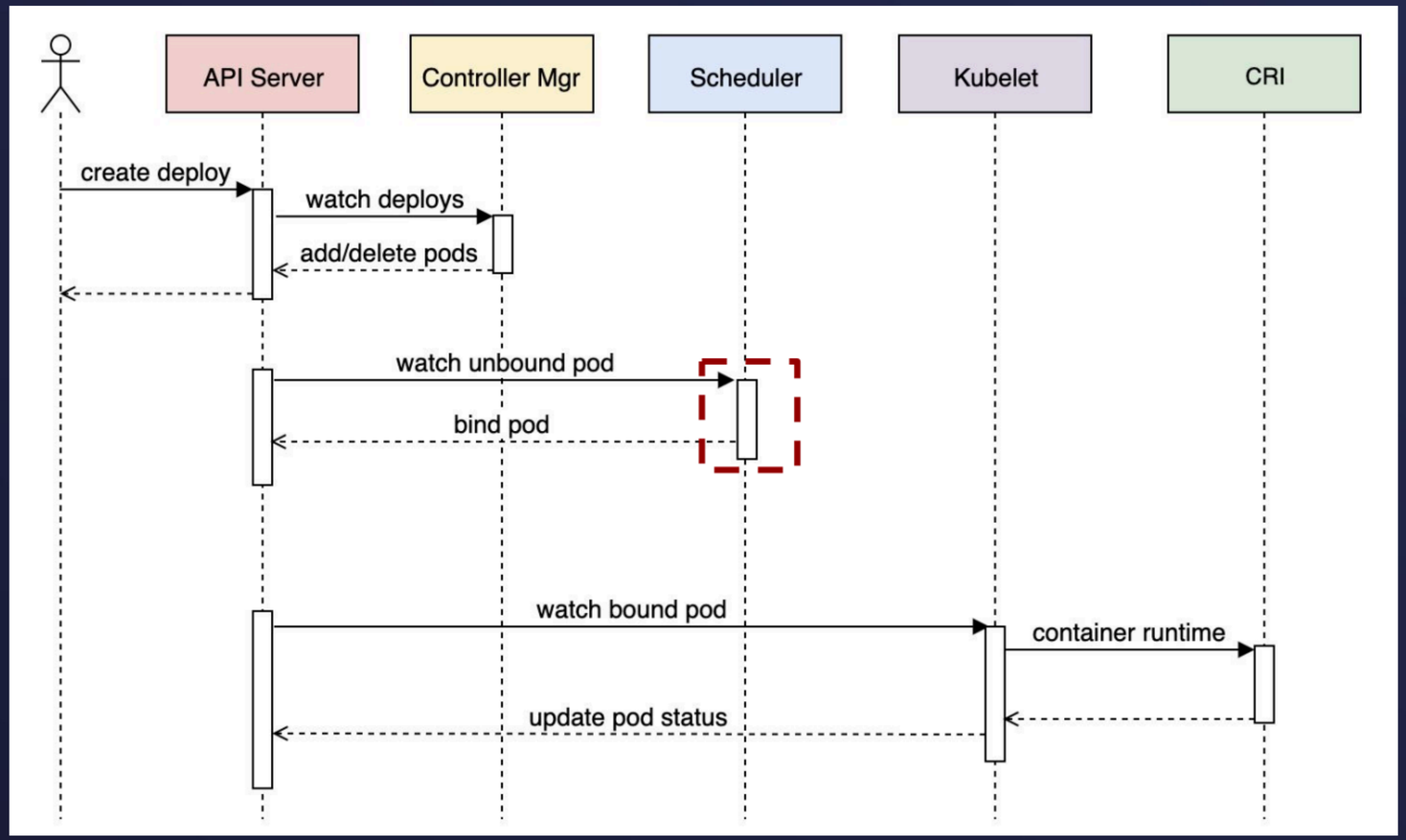
Kubernetes Scheduling Overview

KubeCon | CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

- Pod lifecycle
- Scheduler workflow internals
- Scheduling framework
- Scheduler profiles and plugins

Pod Lifecycle

- Pod created
- Pod scheduled
- Pod running
- Pod completed



Scheduler Workflow Internals

Informers

Feed Pods

Keep cache up-to-date

Queues

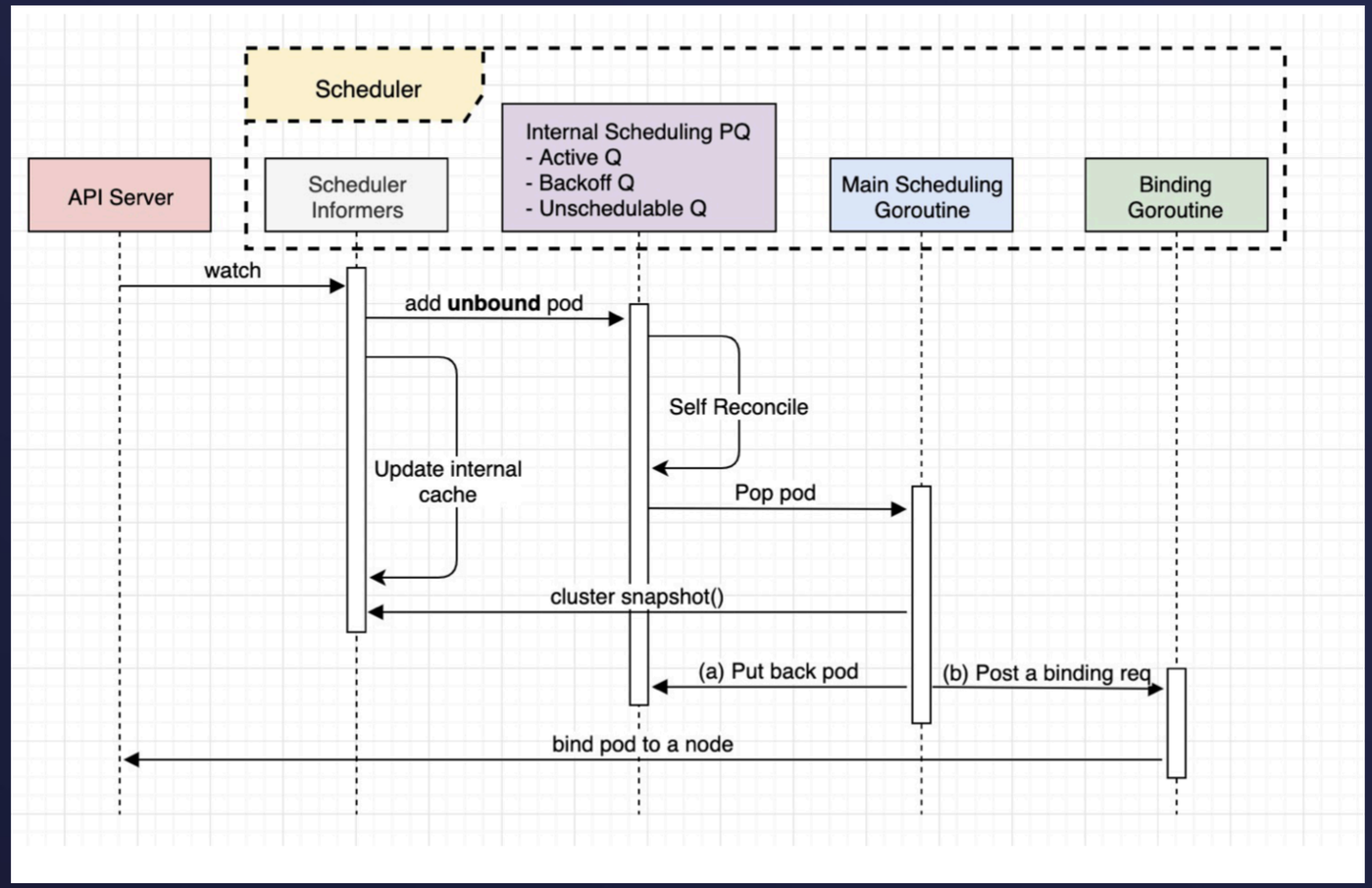
Pop Pods

Faireness

Core Scheduling

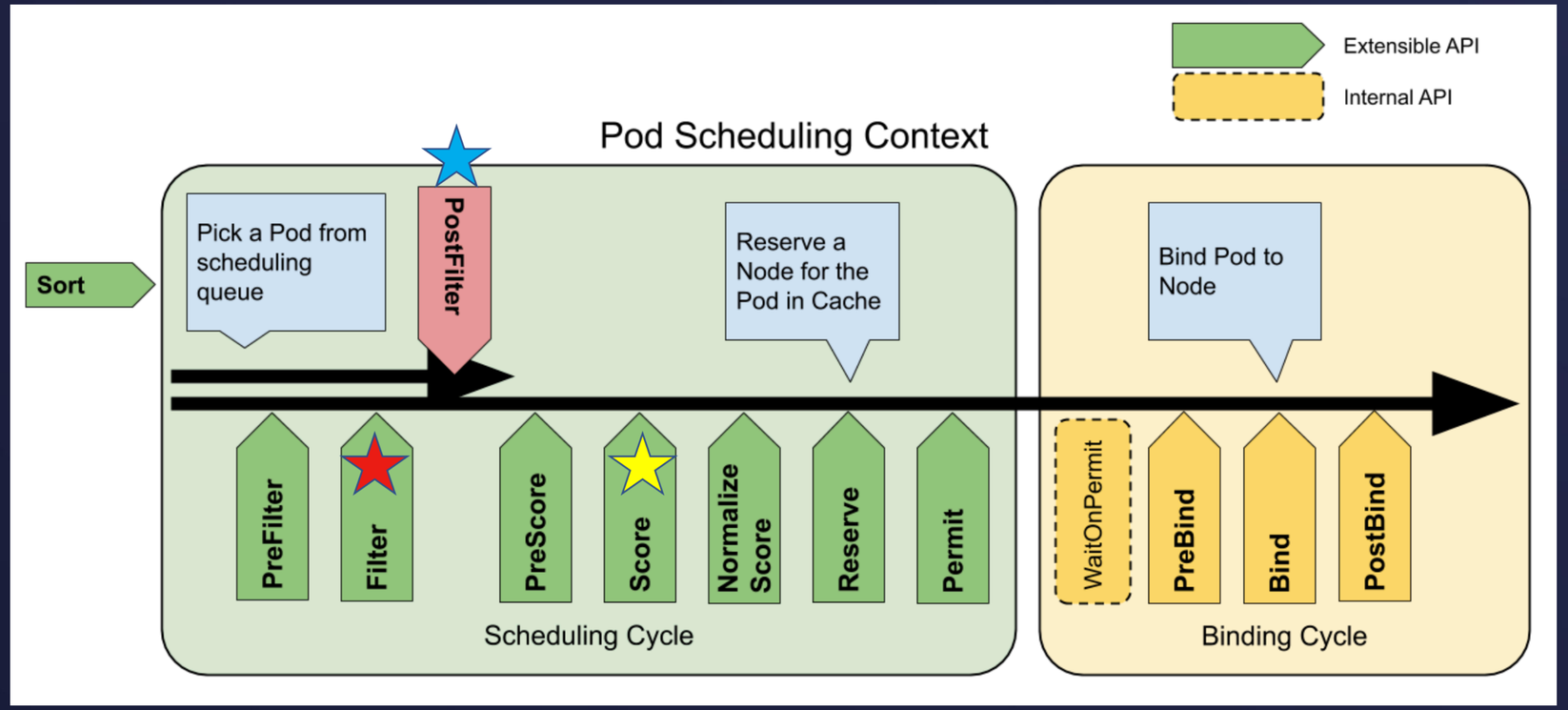
a) queue pod back

b) binding Cycle



Scheduler Framework

- Hard constraints
  - Filter
  - PostFilter (a.k.a. preemption)
- Soft constraints
  - Score: pick the *best* node



Scheduler Profiles and Plugins

- Scheduler profiles
- Scheduler plugins

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Scheduler Configuration

Yibo Zhuang

Kubernetes Scheduler Configuration

Overview

- Customization of scheduling behavior
- Configurations
  - Global parameters
  - Per plugin configuration
  - Multiple profiles

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: /etc/srv/kubernetes/kube-scheduler/kubeconfig
leaderElection:
  leaderElect: true
  leaderElectLeaseDuration: 45s
  leaderElectRenewDeadline: 30s
podInitialBackoffSeconds: 10
podMaxBackoffSeconds: 100
profiles:
- schedulerName: default-scheduler
  plugins:
    multiPoint:
      enabled:
      - name: Foo
```

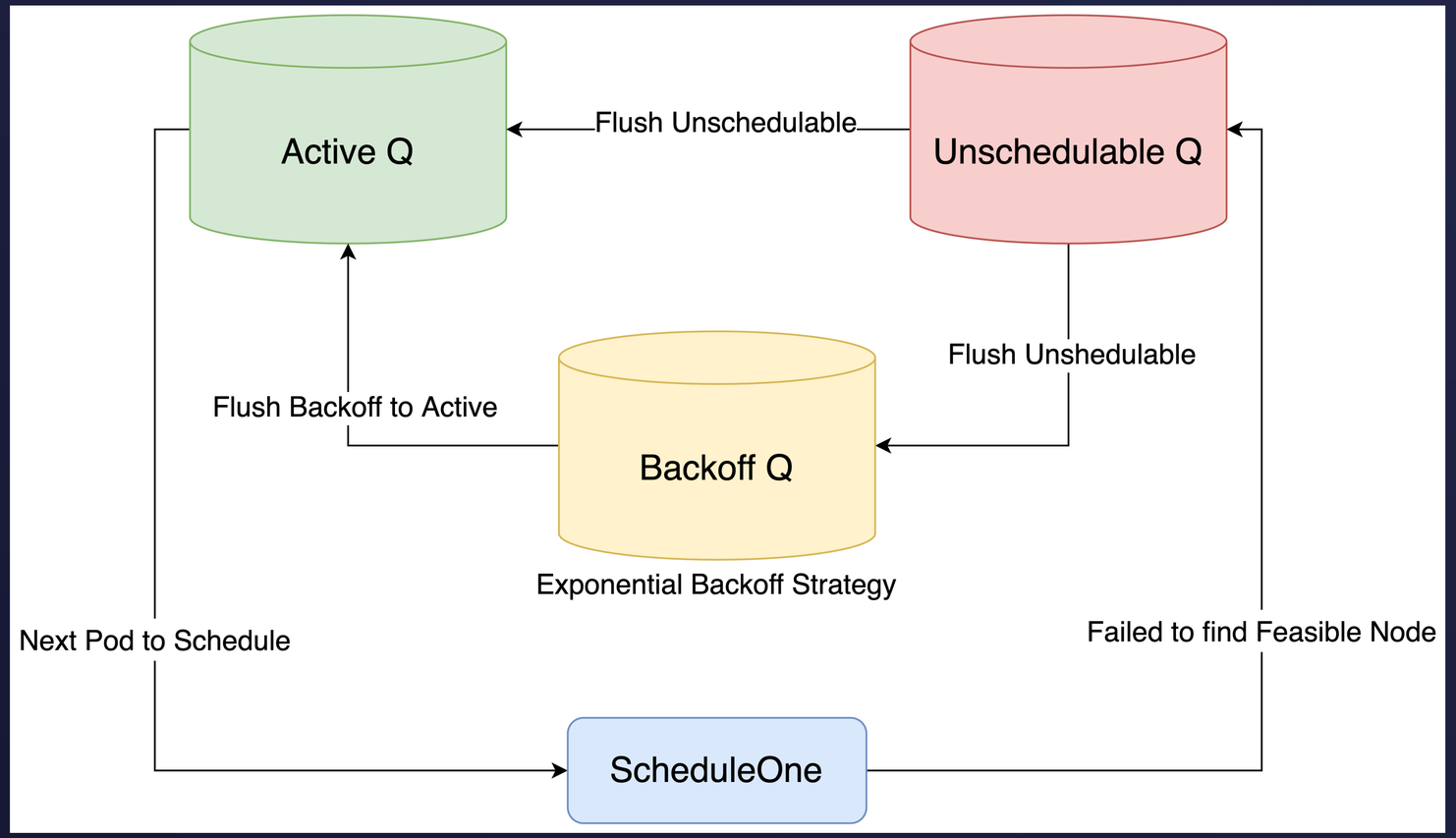
Configuration: Global Parameters

Configuring Scheduler Global Parameters

- *percentageOfNodesToScore*: % of all nodes used for initial search for feasible nodes (default: adaptive between 5 to 50%)
- *leaderElection*: enable scheduler to run in leader-election mode
- *podInitialBackoffSeconds*: initial backoff for unschedulable pods
- *podMaxBackoffSeconds*: maximum backoff for unschedulable pods

Kubernetes Scheduler: Queues

Simplified View of Scheduler Internal Queues



Scheduler Profile Configuration

- Scheduling profile allows for granular control of extension points
  - Extension points: *queueSort, preFilter, filter, postFilter, preScore, score, reserve, permit, preBind, bind, postBind*
- pluginConfig allows for setting plugin-specific arguments
- weight allows for favoring plugin score during normalization

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- name: default-scheduler
  plugins:
    queueSort:
      enabled:
      - name: MyAwesomeSort
      disabled:
      - name: "*"
    score:
      disabled:
      - name: PodTopologySpread
      enabled:
      - name: MyCustomPluginA
        weight: 2
      - name: MyCustomPluginB
        weight: 1
  pluginConfig:
  - name: MyCustomPluginA
    args:
      percentageNodeReserved: 50
      learningStrategy: SimpleRegression
```

Profile Configuration: multiPoint

Starting from v1beta3, profile config added *multiPoint* for simplified enablement/ disablement across several extension points

Example

Plugin	Extension Points
DefaultQueueSort	QueueSort
CustomQueueSort	QueueSort
DefaultPlugin1	PreFilter, Filter, PreScore, Score
DefaultPlugin2	PreScore, Score
CustomPlugin1	PreFilter, Filter, PreScore, Score
CustomPlugin2	PreFilter, Filter, PreScore, Score

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: default-scheduler
  plugins:
    multiPoint:
      enabled:
      - name: CustomQueueSort
      - name: CustomPlugin1
        weight: 3
      - name: CustomPlugin2
        weight: 2
      disabled:
      - name: DefaultQueueSort
    filter:
      disabled:
      - name: DefaultPlugin1
    score:
      enabled:
      - name: DefaultPlugin2
```

Scheduler Configuration: Multi-Profile

- A single instance of kube-scheduler can run multiple profiles
  - scheduler name
  - set of plugins with extension points
- Pods must provide the scheduler name for the specific profile in *.spec.schedulerName*
  - default profile is *default-scheduler*
- All profiles must use the same plugin for *queueSort* with same parameters

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- name: default-scheduler
  plugins:
    multiPoint:
      enabled:
      - name: MyCustomPluginA
    queueSort:
      enabled:
      - name: MyAwesomeSort
      disabled:
      - name: "*"
  pluginConfig:
  - name: MyAwesomeSort
    args:
      sortingAlgorithm: BucketSort
- name: second-scheduler
  plugins:
    queueSort:
      enabled:
      - name: MyAwesomeSort
      disabled:
      - name: "*"
    preFilter:
      enabled:
      - name: MyCustomPluginB
      disabled:
      - name: "*"
    filter:
      enabled:
      - name: MyCustomPluginB
      disabled:
      - name: "*"
  pluginConfig:
  - name: MyAwesomeSort
    args:
      sortingAlgorithm: BucketSort
```

Scheduler: Default Plugin Parameters

- *NodeResourcesFit*
  - plugin weight: **1**
  - scoring strategy: **LeastAllocated** with cpu weight: **1**, memory weight: **1**
- *InterPodAffinity*
  - plugin weight: **2**
  - hardPodAffinityWeight: **1**
- *NodeResourcesBalancedAllocation*
  - plugin weight: **1**
  - resources for calculation: **cpu, memory**

Scheduler: Resource Bin Packing

- Default *NodeResourcesFit* uses least allocated preferring nodes with lower utilization
- Using *MostAllocated*
  - opposite of *LeastAllocated*, preferring nodes with higher utilization
  - final score determined by weighted sum of configured resource

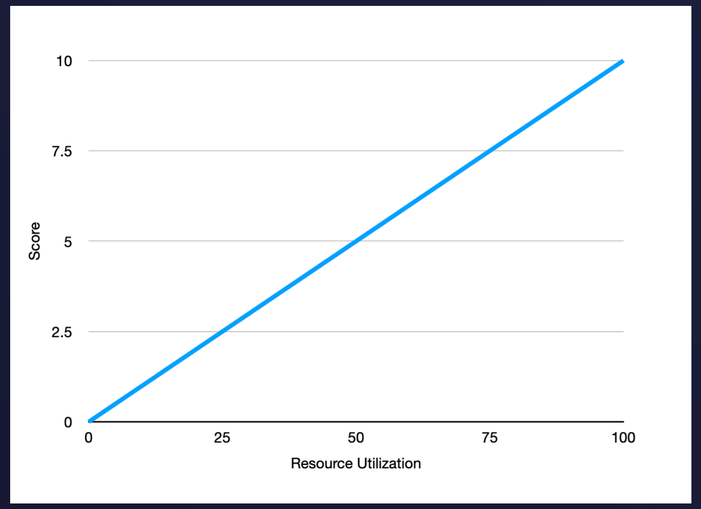
$$\frac{\sum((\frac{resourceRequested}{resourceCapacity} \times 100) \times weight)}{weightSum}$$

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- name: default-scheduler
  pluginConfig:
  - args:
      scoringStrategy:
        resources:
        - name: cpu
          weight: 2
        - name: memory
          weight: 1
        - name: nvidia.com/gpu
          weight: 3
        type: MostAllocated
    name: NodeResourcesFit
```

Scheduler: Resource Bin Packing

- Using *RequestedToCapacityRatio*
  - Allow for customized “scoring shape” with configurable utilization percentage to score mapping

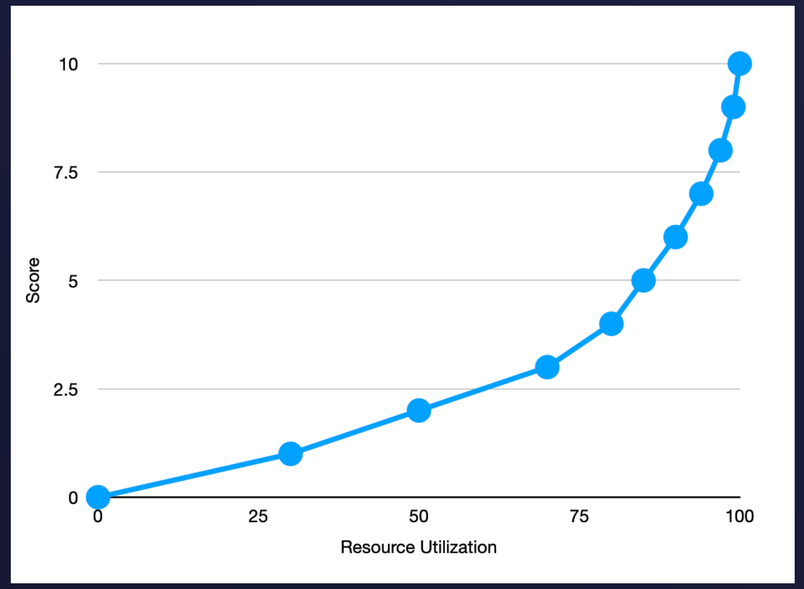
$$\frac{\sum((scoringShape(resourceRequested, resourceCapacity) \times 100) \times weight)}{weightSum}$$




```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- name: default-scheduler
  pluginConfig:
  - args:
      scoringStrategy:
        resources:
        - name: cpu
          weight: 2
        - name: memory
          weight: 1
        - name: nvidia.com/gpu
          weight: 3
        requestedToCapacityRatio:
          shape:
          - utilization: 0
            score: 0
          - utilization: 100
            score: 10
        type: RequestedToCapacityRatio
    name: NodeResourcesFit
```

Scheduler: Resource Bin Packing

- RequestedToCapacityRatio non-linear example




```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- name: default-scheduler
  pluginConfig:
  - args:
      scoringStrategy:
        resources:
        - name: cpu
          weight: 2
        - name: memory
          weight: 1
        - name: nvidia.com/gpu
          weight: 3
        requestedToCapacityRatio:
          shape:
          - utilization: 0
            score: 0
          - utilization: 30
            score: 1
          - utilization: 50
            score: 2
          - utilization: 70
            score: 3
          - utilization: 80
            score: 4
          - utilization: 85
            score: 5
          - utilization: 90
            score: 6
          - utilization: 94
            score: 7
          - utilization: 97
            score: 8
          - utilization: 99
            score: 9
          - utilization: 100
            score: 10
        type: RequestedToCapacityRatio
    name: NodeResourcesFit
```

Demo

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Scheduler Operation

Yuan Chen

Kubernetes Scheduler Operation

- Build and Deployment
- Events/Logs
- Troubleshooting
- Metrics and Dashboards

Testbed

https://github.com/Huang-Wei/2022-kubecon/tree/main/demos/operation

-go1.19: https://go.dev/doc/install

-Kubernetes: https://github.com/kubernetes/kubernetes

-Minikube: https://minikube.sigs.k8s.io/docs/start/

-Manifests: https://github.com/Huang-Wei/2022-kubecon/tree/main/demos/operation

Scheduler: Build and Deployment

Build

- make kube-scheduler

Deployment

- Option 1: run on control plane or master node

  kube-scheduler –config=./scheduler-config.yaml

- Option 2: run as deployment on worker nodes
  - Put into kube-system
  - Set priorityClassName: system-cluster-critical

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
leaderElection:
  leaderElect: false
clientConnection:
  kubeconfig: "/root/.kube/config"
  qps: 300
  burst: 600
profiles:
- schedulerName: default-kube-scheduler
  plugins:
- schedulerName: best-fit-scheduler
  plugins:
  pluginConfig:
    - name: NodeResourcesFit
      args:
        scoringStrategy:
          type: MostAllocated
podInitialBackoffSeconds: 1
podMaxBackoffSeconds: 10
percentageOfNodesToScore: 100
```

Scheduler Startup

- Path
- kubeconfig
- Command line arguments
- Scheduler config
  - `--write-config-to`
- RBAC/ServiceAccount

```
/home/yuan/kubernetes/_output/bin/kube-scheduler --config=./scheduler-config.yaml --v=5
I1007 17:11:27.918411  309408 flags.go:64] FLAG: --authentication-skip-lookup="false"
I1007 17:11:27.918414  309408 flags.go:64] FLAG: --authentication-token-webhook-cache-ttl="10s"
I1007 17:11:27.918782  309408 flags.go:64] FLAG: --write-config-to=""
...
I1007 17:11:28.571284  309408 configfile.go:105] "Using component config" config=<
        apiVersion: kubescheduler.config.k8s.io/v1
        clientConnection:
          burst: 4000
          kubeconfig: /root/.kube/config
          qps: 2000
        enableProfiling: true
        kind: KubeSchedulerConfiguration
        leaderElection:
          leaderElect: false
          leaseDuration: 15s
          renewDeadline: 10s
          resourceName: kube-scheduler
          resourceNamespace: kube-system
          retryPeriod: 2s
        parallelism: 16
        percentageOfNodesToScore: 100
        podInitialBackoffSeconds: 1
        podMaxBackoffSeconds: 10
        profiles:
        - pluginConfig:
          - args:
              apiVersion: kubescheduler.config.k8s.io/v1
              kind: DefaultPreemptionArgs
              minCandidateNodesAbsolute: 100
              minCandidateNodesPercentage: 10
            name: DefaultPreemption
          - args:
              apiVersion: kubescheduler.config.k8s.io/v1
              kind: NodeResourcesFitArgs
              scoringStrategy:
                resources:
                - name: cpu
                  weight: 1
                - name: memory
                  weight: 1
                type: LeastAllocated
            name: NodeResourcesFit
...
I1007 17:11:28.572155  309408 server.go:149] "Starting Kubernetes Scheduler" version="v1.26.0-alpha.1.295+8836d51e433192"
I1007 17:11:28.572180  309408 server.go:151] "Golang settings" GOGC="" GOMAXPROCS="" GOTRACEBACK=""
I1007 17:11:28.573622  309408 secure_serving.go:210] Serving securely on [::]:10259
I1007 17:11:28.573831  309408 reflector.go:221] Starting reflector *v1.PersistentVolume (0s) from vendor/k8s.io/client-go/informers/factory.go:149
I1007 17:11:28.573970  309408 reflector.go:221] Starting reflector *v1.Node (0s) from vendor/k8s.io/client-go/informers/factory.go:149
I1007 17:11:28.574353  309408 reflector.go:257] Listing and watching *v1.Node from vendor/k8s.io/client-go/informers/factory.go:149
I1007 17:11:28.574035  309408 reflector.go:257] Listing and watching *v1.ReplicaSet from vendor/k8s.io/client-go/informers/factory.go:149
I1007 17:11:28.589562  309408 node_tree.go:65] "Added node in listed group to NodeTree" node="minikube" zone=""
I1007 17:11:28.589613  309408 eventhandlers.go:69] "Add event for node" node="minikube"
I1007 17:11:28.593533  309408 eventhandlers.go:184] "Add event for scheduled pod" pod="kubernetes-dashboard/kubernetes-dashboard-54596f475f-fmwtr"
I1007 17:11:28.593911  309408 eventhandlers.go:184] "Add event for scheduled pod" pod="kube-system/kube-controller-manager-minikube"
...
I1007 17:11:28.674783  309408 shared_informer.go:303] caches populated
```

Events/Logs

/var/log/kubernetes/kube-scheduler.log

Scheduled pod

```
I1005 23:45:45.742605  868235 eventhandlers.go:172] add event for unscheduled pod yuan/test-pod
I1005 23:45:45.742697  868235 scheduling_queue.go:915] About to try and schedule pod yuan/test-pod
I1005 23:45:45.742703  868235 scheduler.go:483] Attempting to schedule pod: yuan/test-pod
I1005 23:45:45.745077  868235 default_binder.go:51] Attempting to bind yuan/testpod to node013.kube.cloud.com
I1005 23:45:45.749509  868235 scheduler.go:655] "Successfully bound pod to node" pod="yuan/yuan-pod" node="node013.kube.cloud.
com" evaluatedNodes=132 feasibleNodes=87
I1005 23:45:45.749516  868235 eventhandlers.go:216] delete event for unscheduled pod yuan/test-pod
I1005 23:45:45.749536  868235 eventhandlers.go:236] add event for scheduled pod yuan/test-pod
```

Unscheduled pod

```
I0910 14:00:39.988895  656292 factory.go:327] "Unable to schedule pod; no fit; waiting" pod="yuan/unschedulable-p
od" err="0/132 nodes are available: 72 node(s) didn't match pod affinity/anti-affinity, 72 node(s) didn't match p
od anti-affinity rules, 19 node(s) were unschedulable, 16 node(s) had untolerated taint {kube.com/new-node: pendi
ng}, 11 node(s) had untolerated taint {kube.com/cordon-qa: true}, 8 node(s) had untolerated taint {node.kubernete
s.io/unreachable: }, 2 node(s) had untolerated taint {kube.com/identity-unhealthy: }, 2 node(s) had untolerated t
aint {yuan: gpu-test}, 1 node(s) had untolerated taint {kube.com/unhealthy: }
```

Troubleshooting: Unscheduled Pods

- Misconfiguration
  - I0925 16:00:58.439197 890649 factory.go:327] "Unable to schedule pod; no fit; waiting" pod="yuan-dev/nginxstp8" err="0/100 nodes are available: 100 persistentvolumeclaim \"yuan-pvc72\" not found."
- Insufficient resources
  - 3 Insufficient cpu, 20 Insufficient memory, 10 Insufficient ephemeral-storage
- Taint, affinity/anti-affinity conflicts
  - I0910 14:00:39.988895 656292 factory.go:327] "Unable to schedule pod; no fit; waiting" pod="yuan/unschedulable-pod" err="0/132 nodes are available: 72 node(s) didn't match pod affinity/anti-affinity, 72 node(s) didn't match pod anti-affinity rules, 19 node(s) were unschedulable, 16 node(s) had untolerated taint {kube.com/new-node: pending}, 11 node(s) had untolerated taint {kube.com/cordon-qa: true}, 8 node(s) had untolerated taint {node.kubernetes.io/unreachable: }, 2 node(s) had untolerated taint {kube.com/identity-unhealthy: }, 2 node(s) had untolerated taint {yuan: gpu-test}, 1 node(s) had untolerated taint {kube.com/unhealthy: }

Troubleshooting: Poor Performance

- Network connection timeout: apiserver, webhook, etc.

```
10/6/22 8:19:56.057 PM  E1007 03:19:56.057674 1547194 scheduler.go:362] Error updating pod yuan/test-pod: Internal error
occurred: failed calling webhook "opa.kube.io": Post "https://opa.svc.kube.com/?timeout=5s": context deadline exceeded
~
```

- Slow rescheduling: large backoff periods
  - Large podInitialBackoffSeconds, podMaxBackoffSeconds

```
I1007 17:49:00.057787  311771 schedule_one.go:872] "Unable to schedule pod; no fit; waiting" pod="default/large-cpu-pod" err="0/1 no
des are available: 1 Insufficient cpu. preemption: 0/1 nodes are available: 1 Insufficient cpu.."
I1007 17:49:00.058000  311771 scheduling_queue.go:430] "Pod moved to an internal scheduling queue" pod="default/large-cpu-pod" event
="ScheduleAttemptFailure" queue="Unschedulable"
I1007 17:49:00.058069  311771 schedule_one.go:946] "Updating pod condition" pod="default/large-cpu-pod" conditionType=PodScheduled c
onditionStatus=False conditionReason="Unschedulable"

I1007 17:52:55.776059  311771 scheduling_queue.go:321] "Pod moved to an internal scheduling queue" pod="default/pod" event="PodAdd"
queue="Active"
...
I1007 17:53:06.385458  311771 scheduling_queue.go:973] "About to try and schedule pod" pod="default/large-cpu-pod"
I1007 17:53:06.385478  311771 schedule_one.go:81] "Attempting to schedule pod" pod="default/large-cpu-pod"
```

- HOL blocking： frequent rescheduling of unschedulable pods
  - Small podInitialBackoffSeconds, podMaxBackoffSeconds

```
I1007 17:52:55.776013  311771 eventhandlers.go:116] "Add event for unscheduled pod" pod="default/pod"
I1007 17:52:55.776059  311771 scheduling_queue.go:321] "Pod moved to an internal scheduling queue" pod="default/pod" event="PodAdd"
queue="Active"
I1007 17:59:55.776095  311771 scheduling_queue.go:973] "About to try and schedule pod" pod="default/pod"
I1007 17:59:55.776109  311771 schedule_one.go:81] "Attempting to schedule pod" pod="default/pod"
~
```

Troubleshooting: Slow Scheduling

- PercentageOfNodesToScore
  - Global
  - Per profile: https://github.com/kubernetes/kubernetes/pull/112521

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
percentageOfNodesToScore: 50
profiles:
- schedulerName: default-kube-scheduler
- schedulerName: best-quality-kube-scheduler
  percentageOfNodesToScore: 100
- schedulerName: best-performance-scheduler
  percentageOfNodesToScore: 5
```

- Example: a 250-node cluster
  - `percentageOfNodesToScore: 60`

```
I1007 18:15:55.497217       1 scheduler.go:655] "Successfully bound pod to node" pod="yuan/pod-try-more-nodes" node="no
kube.com" evaluatedNodes=232 feasibleNodes=150
```

  - default: adaptive
    - `percentageOfNodesToScore = int32(50) - numAllNodes/125 = 28`
    - `minFeasibleNodesToFind = 100`

```
I1007 18:15:55.497217       1 scheduler.go:655] "Successfully bound pod to node" pod="yuan/pod-try-fewer-nodes" node="node0110.cloud
.kube.com" evaluatedNodes=172 feasibleNodes=100
```

Monitoring: Metrics

-Performance

- -scheduler_pod_scheduling_duration_seconds (from submission to scheduled)
- -scheduler_scheduling_attempt_duration_seconds (algorithm+ binding)
- -scheduler_scheduling_algorithm_duration_seconds (algorithm)
- -scheduler_framework_extension_point_duration_seconds (per extension point)
- -scheduler_plugin_execution_duration_seconds (per extension point & plugin)
- -scheduler_scheduling_algorithm_filter_evaluation_seconds
- -scheduler_scheduling_algorithm_priority_evaluation_seconds
- -scheduler_scheduling_algorithm_preemption_evaluation_seconds (preemption time)
- -scheduler_binding_duration_seconds (binding time)

-Scheduling status and results

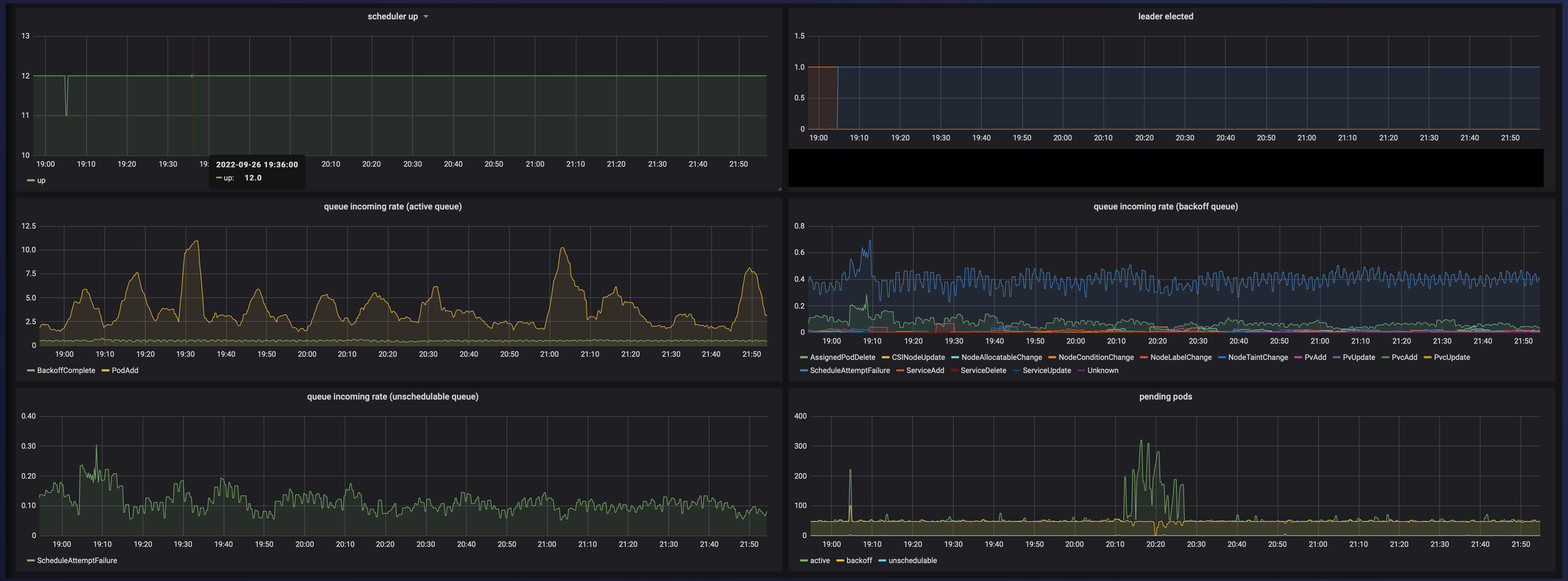
- -scheduler_schedule_attempts_total (total)
- -scheduler_pod_scheduling_attempts (successful attempts)
- -scheduler_pending_pods

-Preemption

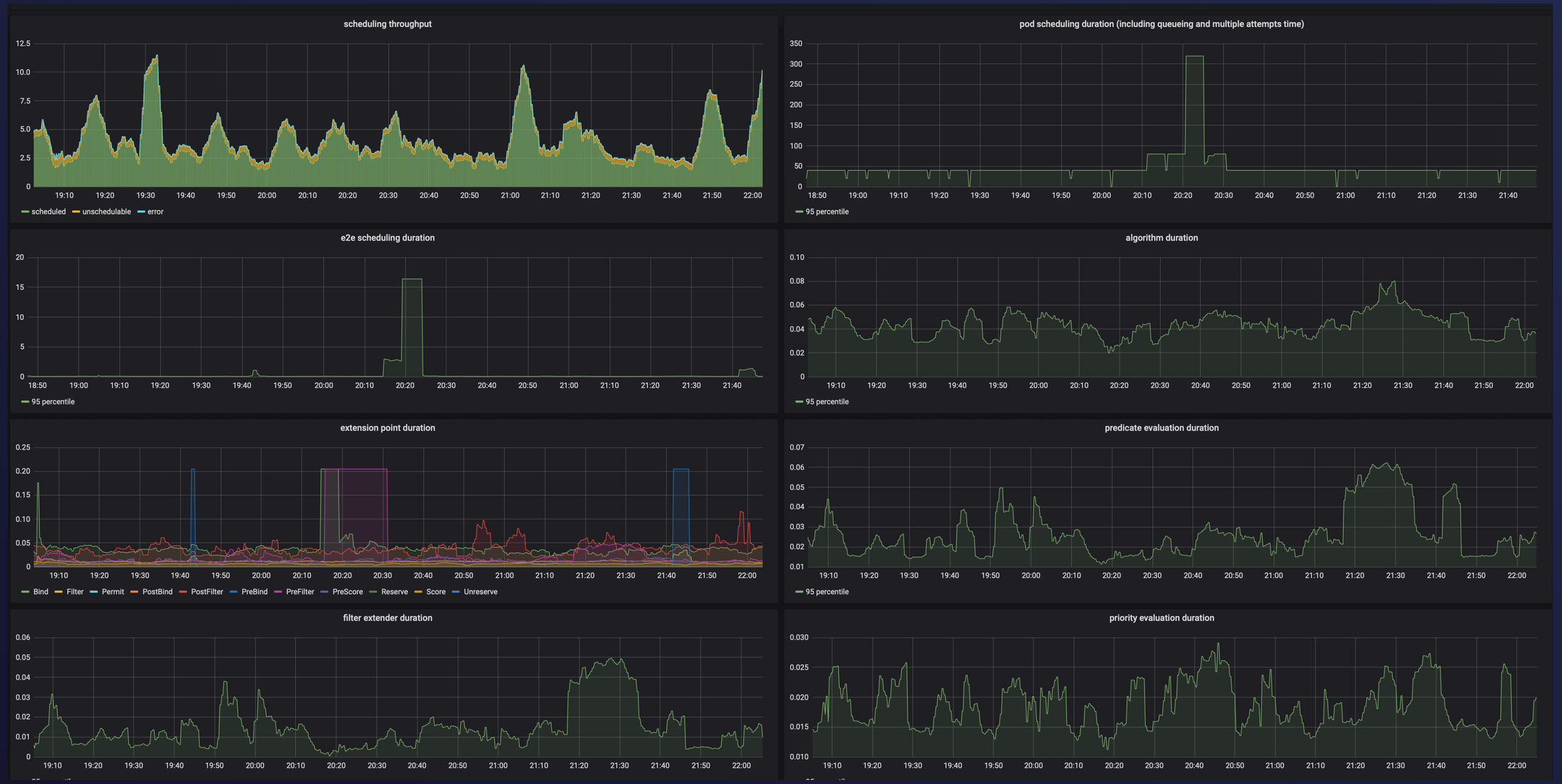
- -scheduler_preemption_victims_sum
- -scheduler_preemption_victims_count
- -scheduler_preemption_attempts_total
- -scheduler_total_preemption_attempt

https://github.com/kubernetes/kubernetes/blob/master/pkg/scheduler/metrics/metrics.go

Monitoring: Dashboard



Monitoring: Dashboard (con't)



Monitoring: Dashboard (con't)

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Scheduler Extension

Wei Huang, Chen Wang

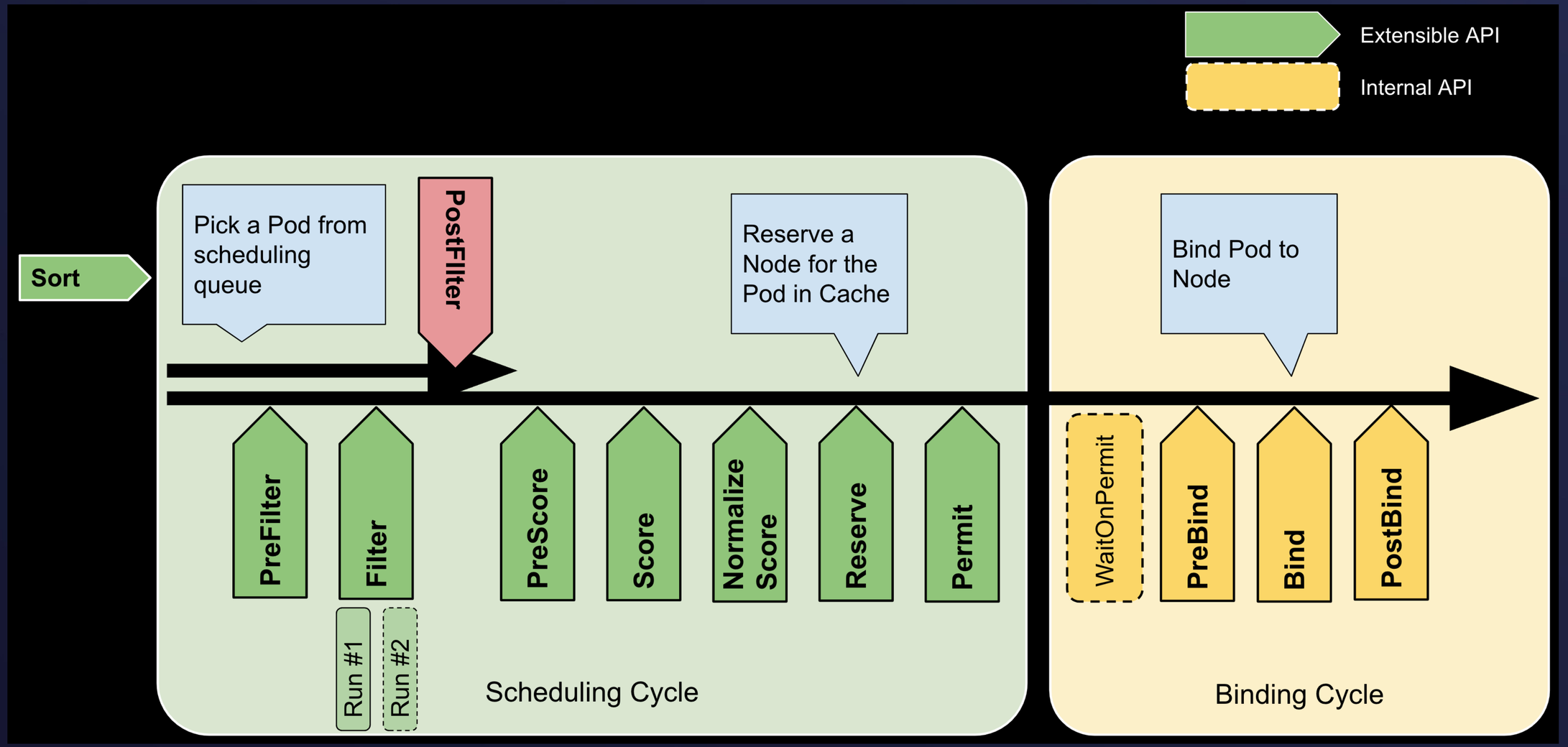
Kubernetes Scheduler Extension

- Overview
  - Extend Kubernetes scheduler: why, how?
  - Scheduler Framework Anatomy
- Use cases
  - Coscheduling
  - Load-awareness Pod lifecycle and workflow
- Demo (a toy example)

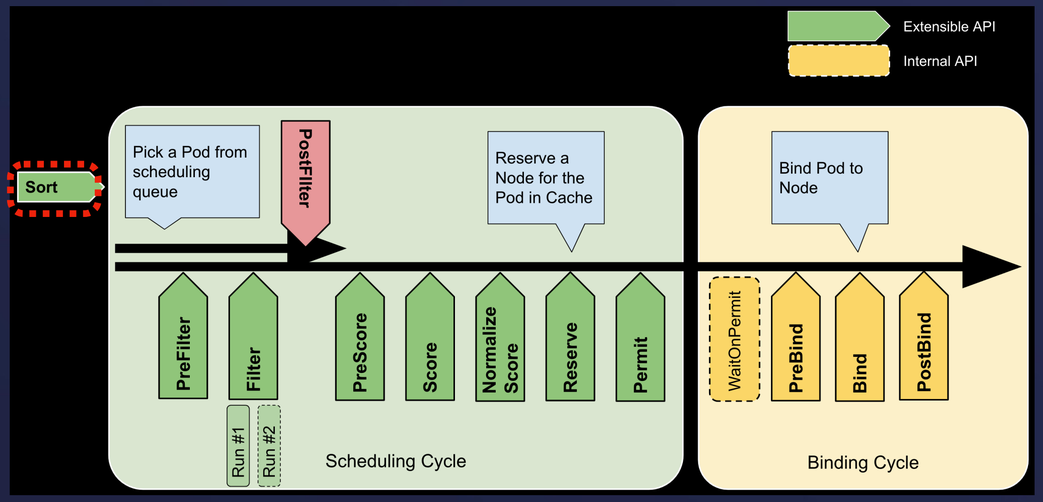
How to Extend Kubernetes Scheduler

- Scheduler extenders
- Scheduler plugins
- Write your own scheduler

Scheduler Framework Anatomy



QueueSort

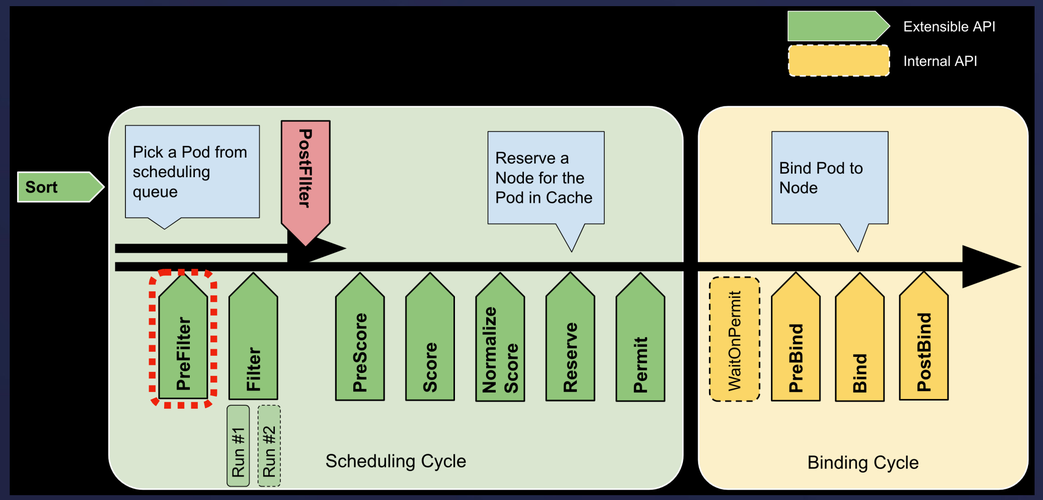


- Sort Pods in internal activeQ
- Stateless implementation

```
Less(*QueuedPodInfo, *QueuedPodInfo) bool
```

- ⚠ Enable only one global QueueSort plugin
- Default QueueSort (to honor .spec.priority)
- Co-scheduling (to group Pods back-to-back)

PreFilter



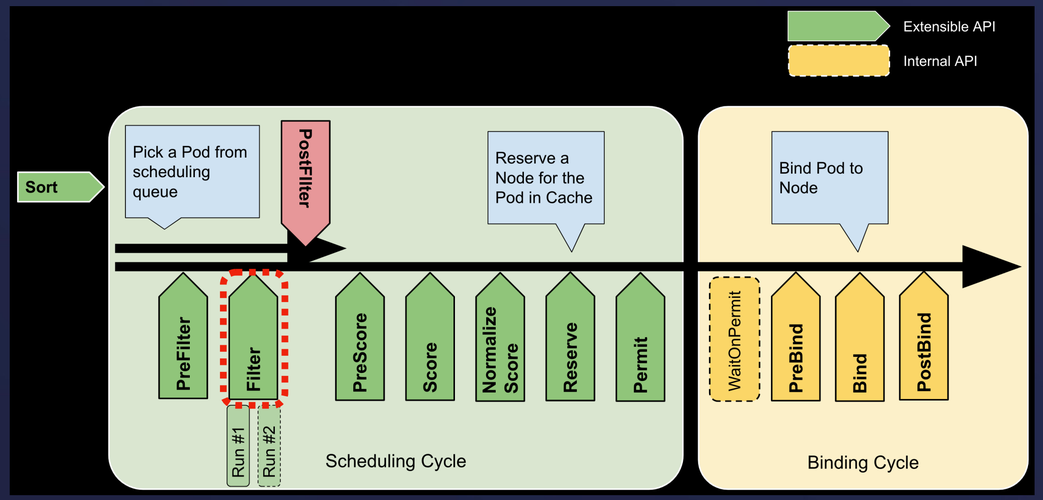
- Pre-check if the Pod should proceed scheduling
- Pre-calculation of `cycleState` (for Filter / PostFilter)

```
PreFilter(ctx context.Context, state *CycleState, p *v1.Pod) (*PreFilterResult, *Status)
```

- Return as early as you can
- Leverage latest interface to return filtered nodes list
- Parallelize processing if needed
- Implement `PreFilterExtensions` (for preemption)

⚠ Think twice on the returned Status (preemption)

Filter

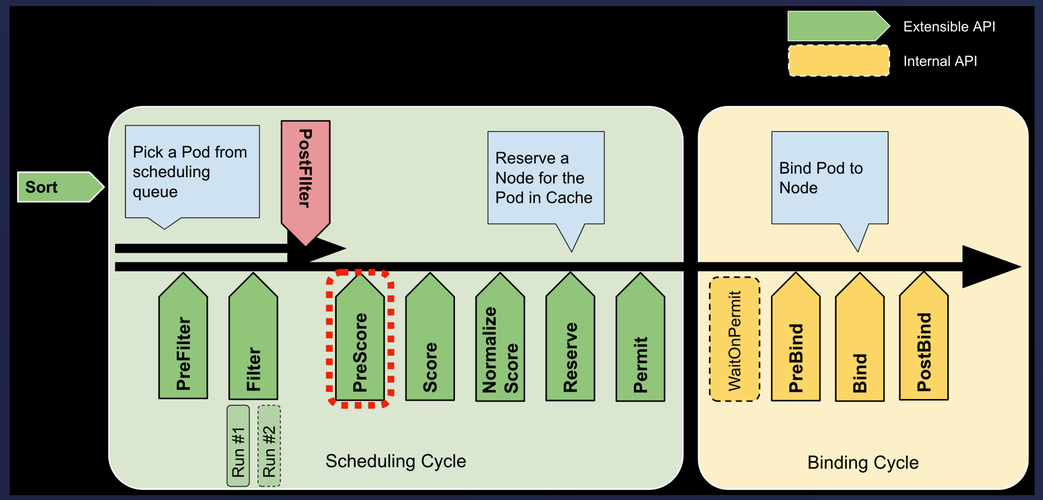


✍ Given a Node, Check if Pod satisfies a particular *hard* scheduling constraint

- Be efficient - avoid O(n) operation
- Build plugin-specific cache if needed (in a separate goroutine)
- Leverage pre-calculated `cycleState` if needed

```
Filter(ctx context.Context, state *CycleState, pod *v1.Pod, nodeInfo *NodeInfo) *Status
```

PreScore

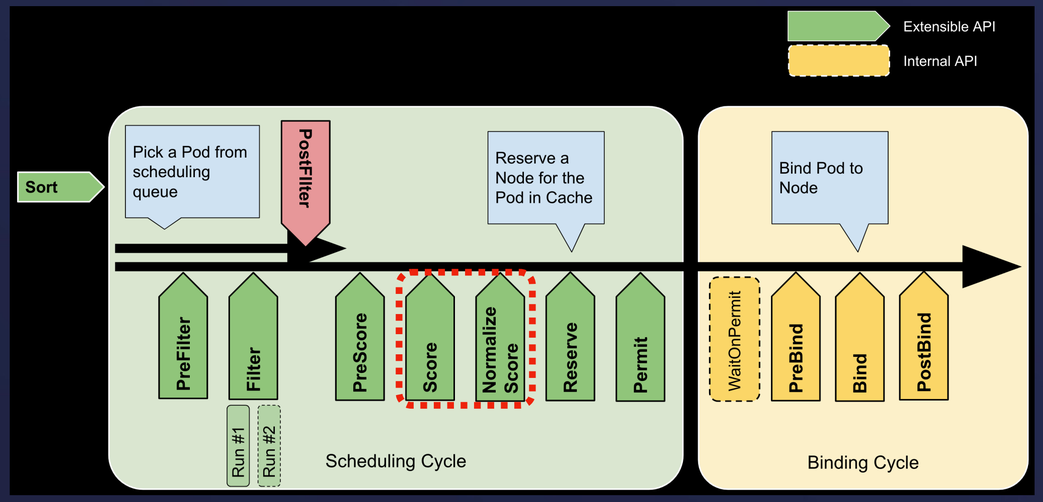


- Pre-calculation of `cycleState` (for Score)

- Leverage cluster snapshot if needed
- Parallelize processing if needed

```
PreScore(ctx context.Context, state *CycleState, pod *v1.Pod, nodes []*v1.Node) *Status
```

Score / Normalize Score



✎ Given a Node, calculate the score so as to rank all feasible Nodes

- Leverage pre-calculated `cycleState` if needed
- Implement `ScoreExtensions` to normalize Scores
- Scores calculated in each {plugin, Node} are weighted and summed up

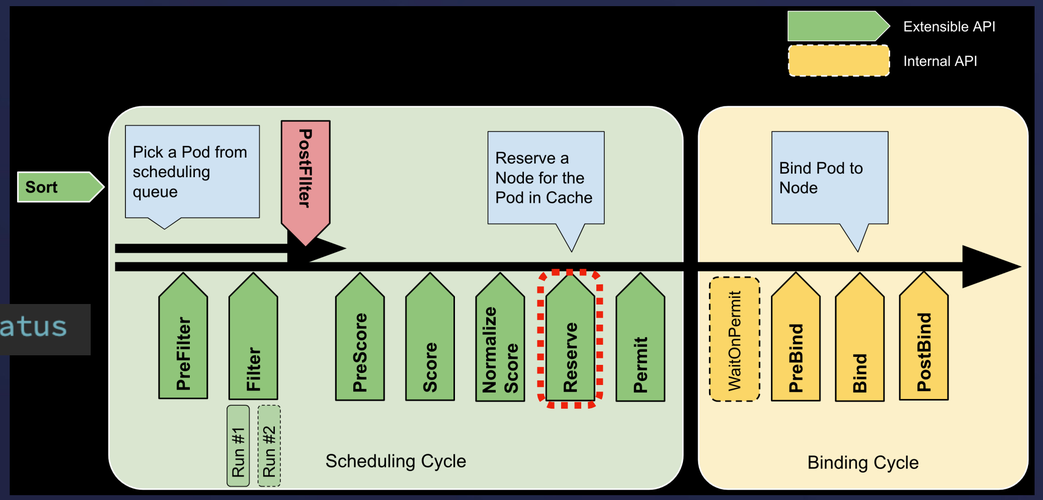
```
Score(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string) (int64, *Status)
```

```
NormalizeScore(ctx context.Context, state *CycleState, p *v1.Pod, scores NodeScoreList) *Status
```

Reserve

```
Reserve(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string) *Status
```

```
Unreserve(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string)
```



✏ Transiently reserve a chunk of Pod resources in the core scheduler cache

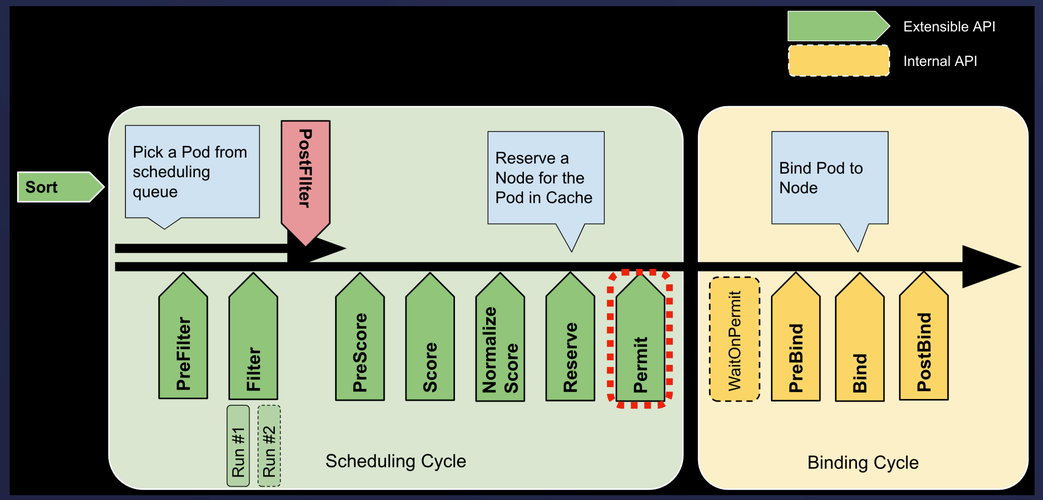
✨ Designed for optimistic concurrency

✨ False positive better than false negative

⚠ Faulty implementation will cause inconsistency

⚠ Keep in mind to `Unreserve` upon unexpected errors

Permit



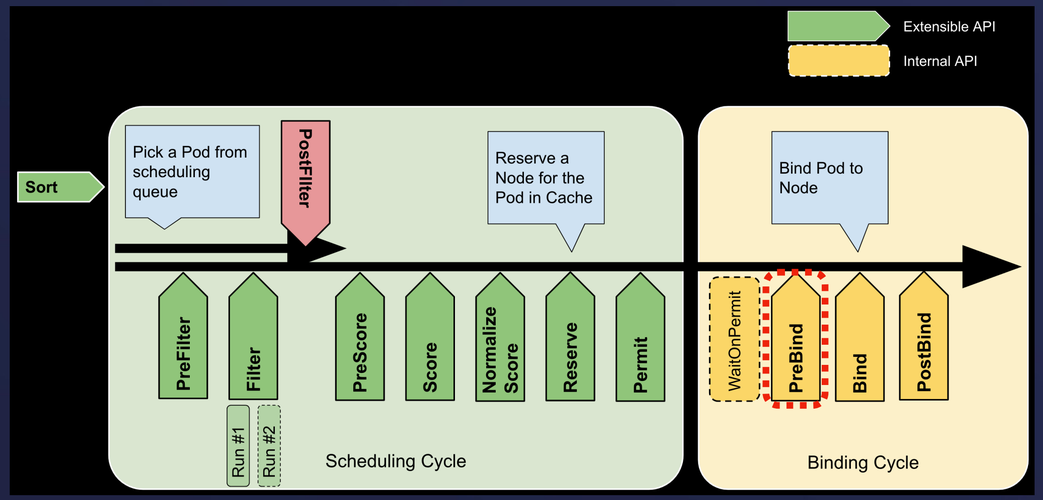
✍ Semi-approve a Pod's scheduling, with a timeout waiting for the final approve

Typically used to schedule a group of Pods that have inter-dependencies

Allow / Reject the relevant Pods in time

```
Permit(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string) (*Status, time.Duration)
```

PreBind



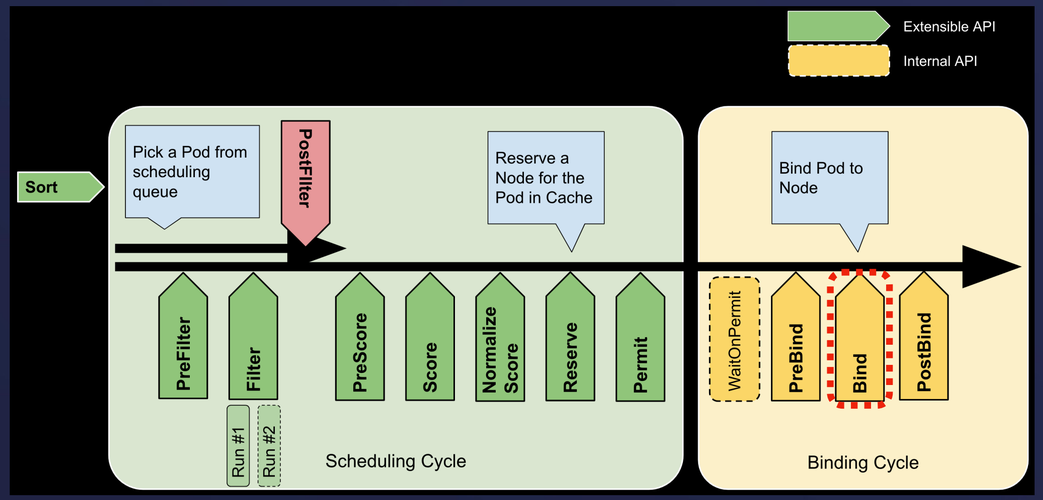
✏️ Final gate to fire Pod's scheduling

✨ Typically used to ensure a Pod's dependent resources are ready (e.g. VolumeBinding)

✨ Call `Unreserve` to cleanup resources if needed

```
PreBind(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string) *Status
```

Bind

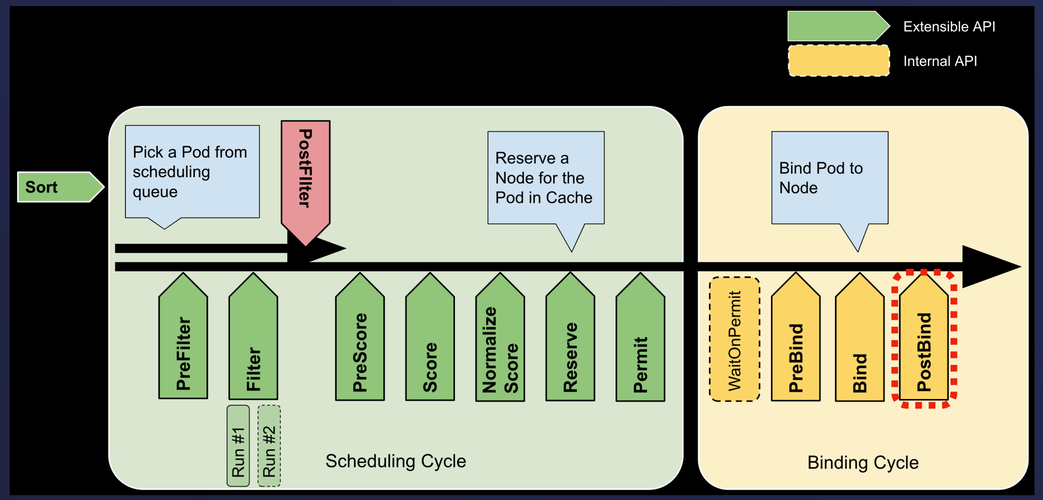


- Bind the Pod to a particular Node

- DefaultBind plugin suffices in most cases
- May be needed in a multi-cluster scheduling scenario
- Call `Unreserve` to cleanup resources if needed

```
Bind(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string) *Status
```

PostBind



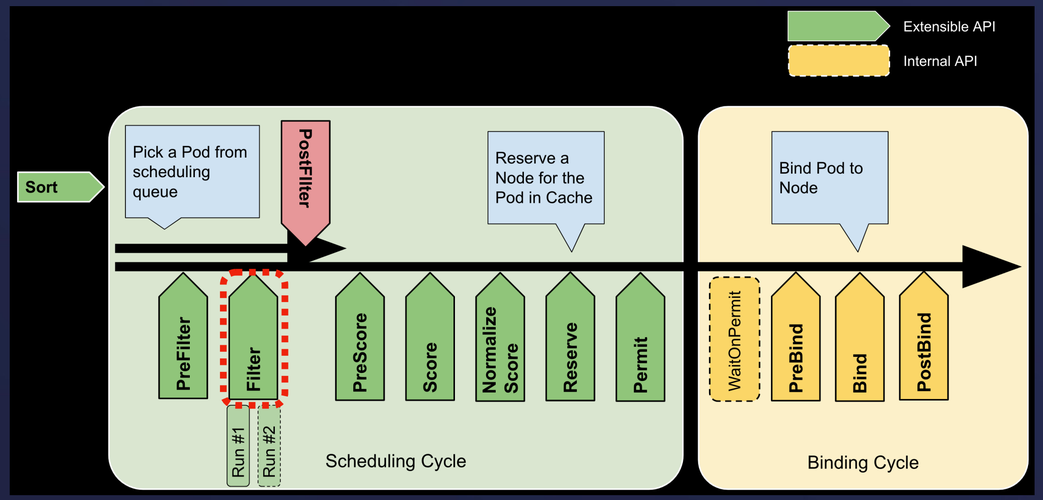
✏ Post processing after a successful Pod binding

Usually for informational purpose, like logging

⚠ Not recommended to mutate the bound Pod

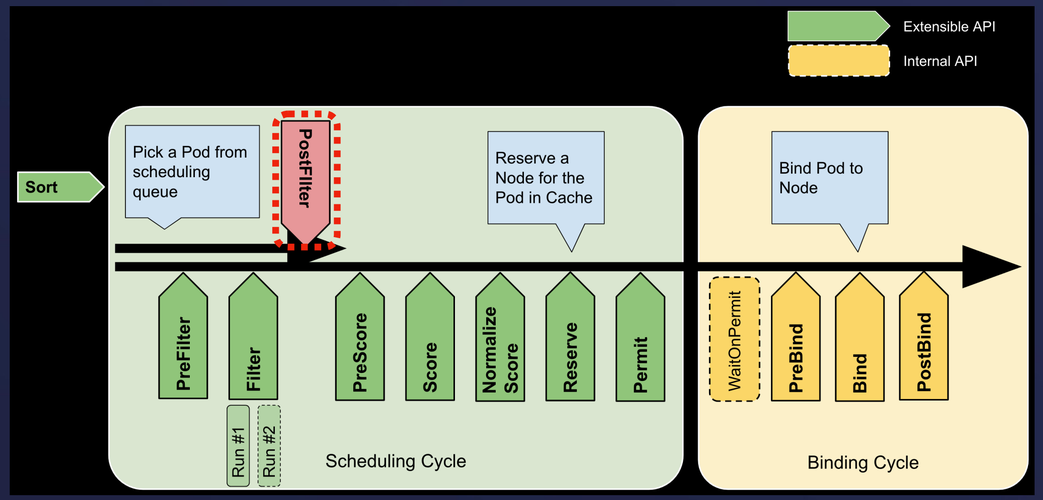
```
PostBind(ctx context.Context, state *CycleState, p *v1.Pod, nodeName string)
```

Filter



No Node to place the Pod

PostFilter



✏ Try to find a Node to host the Pod, by running some extra actions

- Preemption is a default PostFilter plugin
- Cascaded with logic implemented in PreFilter / Filter
- Implement your own preemption logic (like cross-node preemption)

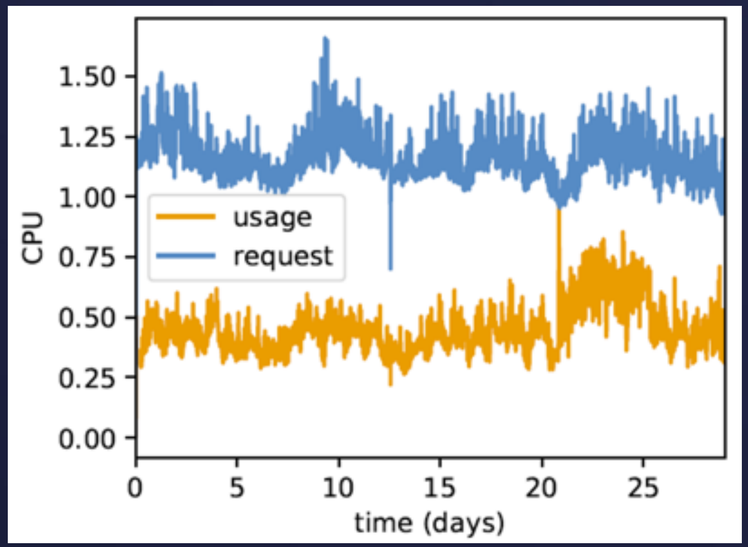
⚠ Think twice whether you really need it

```
PostFilter(ctx context.Context, state *CycleState, pod *v1.Pod, filteredNodeStatusMap NodeToStatusMap)
(*PostFilterResult, *Status)
```

Use case I: Load-aware Scheduler Plugins

- Kubernetes
  - Default scheduler schedules pods based on their *requests* and what is available on nodes.
- Developers
  - They are known to grossly *over-provision resources* for their containers to avoid under-provisioning risks (pod evictions due to OOM and performance issues, etc.)
- Benchmarking to get accurate requests?
  - Cumbersome
  - May not be feasible: *estimating real load is hard*.

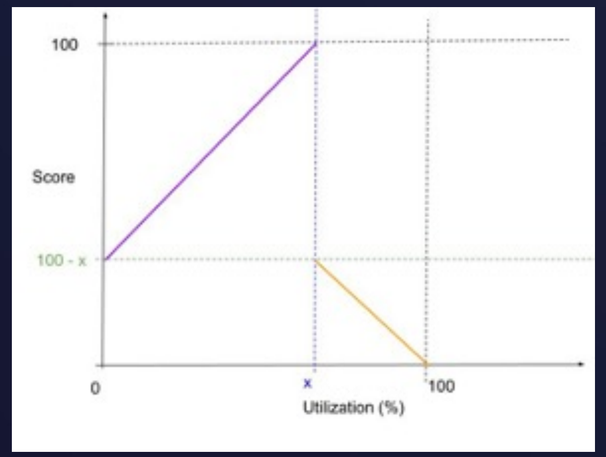
Allocation based scheduling can lead to *underutilized nodes* (CPU, Memory…) and *fragmentation* of cores



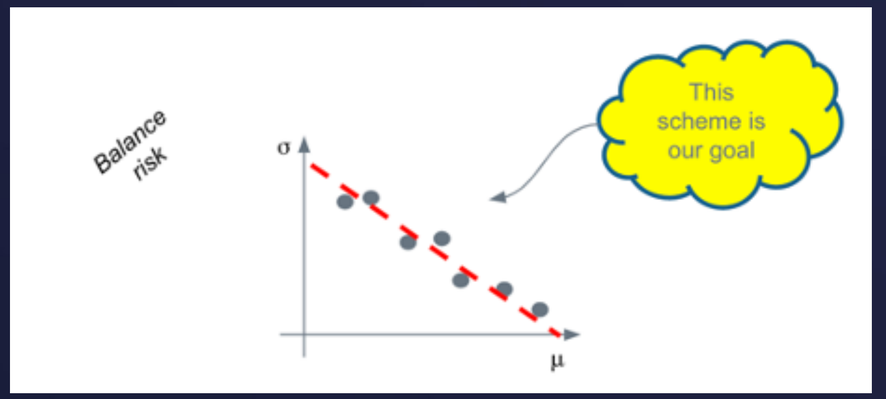
29 day Google trace
Source: [TN Le, Z Liu] Flex: Closing the Gaps between Usage and Allocation

Use case I: Load-aware Scheduler Plugins

- Target Load Packing Plugin
  - Scoring Plugin
- Objective
  - Target utilization on all running nodes
    - Maintain x% CPU utilization for all nodes in the cluster
    - Other nodes with 0% utilization can be powered off.
  - Maintain a safe margin when there are enough resources
    - No nodes exceed x% utilization when there are nodes with <x%

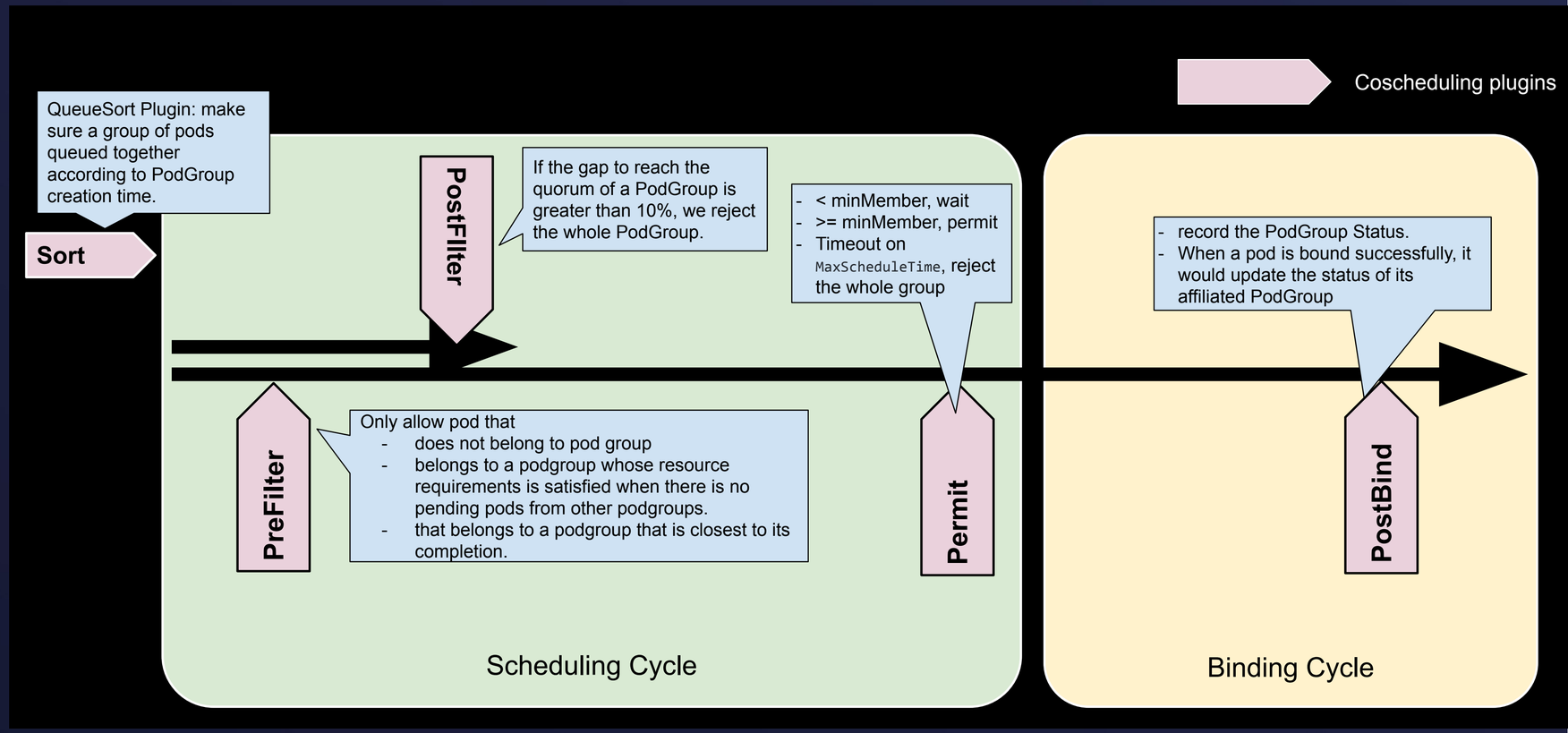


- Load Variation Risk Balancing Plugin
  - Scoring Plugin
- Objective
  - The usage variations on both CPU and memory on nodes are considered when scheduling pods.
  - The scheduler is able to balance the risk of having pod evictions or performance issues due to large load variations on nodes.



Use case II: Co-scheduling

- Motivation
  - Ensure a group of pods can be scheduled altogether.
- Use cases
  - Spark jobs, TensorFlow Job
- Coscheduling Plugin
  - QueueSort
  - PreFilter
  - PostFilter
  - Permit
  - PostBind



Steps to Write and Run a Scheduler Plugin

- Step 1: Define your Score Plugin Struct
  - Create a file under pkg/scorebylabel/score_by_label.go
  - Create a plugin struct with the Plugin name - ScoreByLabel

```
type ScoreByLabel struct {
    handle       framework.Handle
}

const (
    Name = "ScoreByLabel" // Name is the name of the plugin used in Registry and configurations.
)

var _ = framework.ScorePlugin(&ScoreByLabel{})

func (s *ScoreByLabel) Name() string {
    return Name
}
```

Full Tutorial

Steps to Write and Run a Scheduler Plugin

- Step 2: Write the Score Function for your plugin

```go
func (s *ScoreByLabel) Score(ctx context.Context, state *framework.CycleState, p *v1.Pod, nodeName string) (int64, *framework.Status) {
  nodeInfo, err := s.handle.SnapshotSharedLister().NodeInfos().Get(nodeName)
  if err != nil {
    return 0, framework.NewStatus(framework.Error, fmt.Sprintf("error getting node information: %s", err))
  }

  nodeLabels := nodeInfo.Node().Labels

  if val, ok := nodeLabels[LabelKey]; ok {
    scoreVal, err := strconv.ParseInt(val, 10, 64)
    if err != nil {
      klog.V(4).InfoS("unable to parse score value from node labels", LabelKey, "=", val)
      klog.V(4).InfoS("use the default score", DefaultMissingLabelScore, " for node with labels not convertable to int64!")
      return DefaultMissingLabelScore, nil
    }

    klog.Infof("[ScoreByLabel] Label score for node %s is %s = %v", nodeName, LabelKey, scoreVal)

    return scoreVal, nil
  }

  return DefaultMissingLabelScore, nil
}
```

Parse node label with the key defined by LabelKey!

Score

Full Tutorial

NormalizeScore

```go
func (s *ScoreByLabel) ScoreExtensions() framework.ScoreExtensions {
  return s
}

func (s *ScoreByLabel) NormalizeScore(ctx context.Context, state *framework.CycleState, scores framework.NodeScoreList) *framework.Status {
  var higherScore int64
  higherScore = framework.MinNodeScore
  for _, node := range scores {
    if higherScore < node.Score {
      higherScore = node.Score
    }
  }

  for i, node := range scores {
    scores[i].Score = node.Score * framework.MaxNodeScore / higherScore
  }

  klog.Infof("[ScoreByLabel] Nodes final score: %v", scores)
  return nil
}
```

Steps to Write and Run a Scheduler Plugin

- Step 3: Prepare the Plugin Configuration Struct
  - Add <PluginName>Args in apis/config/types.go and apis/config/v1beta2/types.go

```go
// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

// ScoreByLabelArgs holds arguments used to configure the ScoreByLabel plugin.
type ScoreByLabelArgs struct {
        metav1.TypeMeta `json:",inline"`

        // LabelKey is the name of the label to be used for scoring.
        LabelKey *string `json:"labelKey,omitempty"`
}
```

  - Add function SetDefaults_<PluginName>Args in the config/v1beta2/defaults.go to set the default value if input label key is empty.

```go
const (
        // Defaults for ScoreByLabel plugin
        DefaultLabelKey = "score-by-label"
)

// SetDefaults_ScoreByLabelArgs sets the default parameters for the ScoreByLabel plugin
func SetDefaults_ScoreByLabelArgs(obj *ScoreByLabelArgs) {
        if obj.LabelKey == nil {
                obj.LabelKey = &DefaultLabelKey
        }
}
```

  - Generate code for new APIs: ./hack/update-codegen.sh

Full Tutorial

Steps to Write and Run a Scheduler Plugin

- Step 4: Run the scheduler as a secondary scheduler
  - Configure the scheduler to enable only the ScoreByLabel Plugin

Full Tutorial

```yaml
apiVersion: kubescheduler.config.k8s.io/v1beta2
kind: KubeSchedulerConfiguration
leaderElection:
  leaderElect: false
profiles:
- schedulerName: scorebylabel
  plugins:
    score:
      enabled:
      - name: ScoreByLabel
      disabled:
        - name: "*" # disable all default plugins
  pluginConfig:
    - name: ScoreByLabel
      args:
        labelKey: "score-by-label"
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: testpod
spec:
  schedulerName: scorebylabel
  containers:
    - name: testpod
      image: k8s.gcr.io/ubuntu-slim:0.1
      command: ["/bin/sh"]
      args:
        - "-c"
        - "sleep 3600s"
      resources:
        requests:
          cpu: "200m"
          memory: 50Mi
```

Demo

https://youtu.be/ZDkvMtuc2JE

```
> kubectl get nodes
NAME             STATUS   ROLES    AGE   VERSION
10.177.222.232   Ready    <none>   40d   v1.24.4+IKS
10.177.222.243   Ready    <none>   40d   v1.24.4+IKS
10.177.222.253   Ready    <none>   40d   v1.24.4+IKS

~/co/G/s/s/scheduler-plugins/m/scorebylabel | on kubecon22                    ok | at 12:50:35
```

References

- Kubernetes Scheduler Configuration Reference
- kube-scheduler Configuration v1
- Out-of-tree Scheduler Plugins
- Kubernetes Scheduling Framework
- Resource Bin-Packing

KubeCon | CloudNativeCon North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Friday, October 28 • 11:55am - 12:30pm

SIG-Scheduling Deep Dive - Wei Huang, Apple; Qingcan Wang, Alibaba; Kante Yin, DaoCloud; Kensei Nakada, Mercari

Q/A

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Thank you!

Please scan the QR Code above to
leave feedback on this session