Foundation of Internet Platform Development & Operation

Kubernetes in Details: Scheduling

2019-11-09

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Scheduling Framework

- Current
  - 3-Phase
    - Predicates
      - Filter out inappropriate nodes
    - Prioritize
      - Apply a set of "priority functions" that rank the nodes that weren't filtered out by the predicate check
    - Bind
      - The node with the highest priority is chosen (or, if there are multiple such nodes, then one of them is chosen at random)

Predicates

- Built-in Predicates
  - 24

```
const (
        // MatchInterPodAffinityPred defines the name of predicate MatchInterPodAffinity.
        MatchInterPodAffinityPred = "MatchInterPodAffinity"
        // CheckVolumeBindingPred defines the name of predicate CheckVolumeBinding.
        CheckVolumeBindingPred = "CheckVolumeBinding"
        // GeneralPred defines the name of predicate GeneralPredicates.
        GeneralPred = "GeneralPredicates"
        // HostNamePred defines the name of predicate HostName.
        HostNamePred = "HostName"
        // PodFitsHostPortsPred defines the name of predicate PodFitsHostPorts.
        PodFitsHostPortsPred = "PodFitsHostPorts"
        // MatchNodeSelectorPred defines the name of predicate MatchNodeSelector.
        MatchNodeSelectorPred = "MatchNodeSelector"
        // PodFitsResourcesPred defines the name of predicate PodFitsResources.
        PodFitsResourcesPred = "PodFitsResources"
        // NoDiskConflictPred defines the name of predicate NoDiskConflict.
        NoDiskConflictPred = "NoDiskConflict"
        // PodToleratesNodeTaintsPred defines the name of predicate PodToleratesNodeTaints.
        PodToleratesNodeTaintsPred = "PodToleratesNodeTaints"
        // CheckNodeUnschedulablePred defines the name of predicate CheckNodeUnschedulablePredicate.
        CheckNodeUnschedulablePred = "CheckNodeUnschedulable"
        // PodToleratesNodeNoExecuteTaintsPred defines the name of predicate PodToleratesNodeNoExecuteTaints.
        PodToleratesNodeNoExecuteTaintsPred = "PodToleratesNodeNoExecuteTaints"
        // CheckNodeLabelPresencePred defines the name of predicate CheckNodeLabelPresence.
        CheckNodeLabelPresencePred = "CheckNodeLabelPresence"
        // CheckServiceAffinityPred defines the name of predicate checkServiceAffinity.
        CheckServiceAffinityPred = "CheckServiceAffinity"
        MaxCinderVolumeCountPred = "MaxCinderVolumeCount"
        // MaxCSIVolumeCountPred defines the predicate that decides how many CSI volumes should be attached.
        MaxCSIVolumeCountPred = "MaxCSIVolumeCountPred"
        // NoVolumeZoneConflictPred defines the name of predicate NoVolumeZoneConflict.
        NoVolumeZoneConflictPred = "NoVolumeZoneConflict"
        // EvenPodsSpreadPred defines the name of predicate EvenPodsSpread.
        EvenPodsSpreadPred = "EvenPodsSpread"

        // DefaultMaxGCEPDVolumes defines the maximum number of PD Volumes for GCE.
        // GCE instances can have up to 16 PD volumes attached.
        DefaultMaxGCEPDVolumes = 16
        // DefaultMaxAzureDiskVolumes defines the maximum number of PD Volumes for Azure.
        // Larger Azure VMs can actually have much more disks attached.
        // TODO We should determine the max based on VM size
        DefaultMaxAzureDiskVolumes = 16

        // KubeMaxPDVols defines the maximum number of PD Volumes per kubelet.
        KubeMaxPDVols = "KUBE_MAX_PD_VOLS"

        // EBSVolumeFilterType defines the filter name for EBSVolumeFilter.
        EBSVolumeFilterType = "EBS"
        // GCEPDVolumeFilterType defines the filter name for GCEPDVolumeFilter.
        GCEPDVolumeFilterType = "GCE"
        // AzureDiskVolumeFilterType defines the filter name for AzureDiskVolumeFilter.
        AzureDiskVolumeFilterType = "AzureDisk"
        // CinderVolumeFilterType defines the filter name for CinderVolumeFilter.
        CinderVolumeFilterType = "Cinder"
)
```

Predicates

- Default Predicates
  - 7
  - Predicates ordering

```
func defaultPredicates() sets.String {
        return sets.NewString(
                predicates.NoVolumeZoneConflictPred,
                predicates.MaxEBSVolumeCountPred,
                predicates.MaxGCEPDVolumeCountPred,
                predicates.MaxAzureDiskVolumeCountPred,
                predicates.MaxCSIVolumeCountPred,
                predicates.MatchInterPodAffinityPred,
                predicates.NoDiskConflictPred,
                predicates.GeneralPred,
                predicates.PodToleratesNodeTaintsPred,
                predicates.CheckVolumeBindingPred,
                predicates.CheckNodeUnschedulablePred,
        )
```

```
var (
        predicatesOrdering = []string{CheckNodeUnschedulablePred,
                GeneralPred, HostNamePred, PodFitsHostPortsPred,
                MatchNodeSelectorPred, PodFitsResourcesPred, NoDiskConflictPred,
                PodToleratesNodeTaintsPred, PodToleratesNodeNoExecuteTaintsPred, CheckNodeLabelPresencePred,
                CheckServiceAffinityPred, MaxEBSVolumeCountPred, MaxGCEPDVolumeCountPred, MaxCSIVolumeCountPred,
                MaxAzureDiskVolumeCountPred, MaxCinderVolumeCountPred, CheckVolumeBindingPred, NoVolumeZoneConflictPred,
                EvenPodsSpreadPred, MatchInterPodAffinityPred}
)
```

Predicates Ordering

- Static order

Position	Predicate	comments (note, justification...)
1	CheckNodeConditionPredicate	we really don’ t want to check predicates against unschedulable nodes.
2	PodFitsHost	we check the pod.spec.nodeName.
3	PodFitsHostPorts	we check ports asked on the spec.
4	PodMatchNodeSelector	check node label after narrowing search.
5	PodFitsResources	this one comes here since it’ s not restrictive enough as we do not try to match values but ranges.
6	NoDiskConflict	Following the resource predicate, we check disk
7	PodToleratesNodeTaints '	check toleration here, as node might have toleration
8	PodToleratesNodeNoExecuteTaints	check toleration here, as node might have toleration
9	CheckNodeLabelPresence	labels are easy to check, so this one goes before

Predicates in Details

- PodFitsHostPorts:
  - Checks if a Node has free ports (the network protocol kind) for the Pod ports the Pod is requesting.
- PodFitsHost:
  - Checks if a Pod specifies a specific Node by it hostname.
- PodFitsResources:
  - Checks if the Node has free resources (eg, CPU and Memory) to meet the requirement of the Pod.
- PodMatchNodeSelector:
  - Checks if a Pod’ s Node Selector matches the Node’ s label(s).
- NoVolumeZoneConflict:
  - Evaluate if the Volumes that a Pod requests are available on the Node, given the failure zone restrictions for that storage.
- NoDiskConflict:
  - Evaluates if a Pod can fit on a Node due to the volumes it requests, and those that are already mounted.

Predicates in Details

- MaxCSIVolumeCount:
  - Decides how many CSI volumes should be attached, and whether that’ s over a configured limit.
- CheckNodeMemoryPressure:
  - If a Node is reporting memory pressure, and there’ s no configured exception, the Pod won’ t be scheduled there.
- CheckNodePIDPressure:
  - If a Node is reporting that process IDs are scarce, and there’ s no configured exception, the Pod won’ t be scheduled there.
- CheckNodeDiskPressure:
  - If a Node is reporting storage pressure (a filesystem that is full or nearly full), and there’ s no configured exception, the Pod won’ t be scheduled there.
- CheckNodeCondition:
  - Nodes can report that they have a completely full filesystem, that networking isn’ t available or that kubelet is otherwise not ready to run Pods. If such a condition is set for a Node, and there’ s no configured exception, the Pod won’ t be scheduled there.
- PodToleratesNodeTaints:
  - checks if a Pod’ s tolerations can tolerate the Node’ s taints.
- CheckVolumeBinding:
  - Evaluates if a Pod can fit due to the volumes it requests. This applies for both bound and unbound PVCs

Priorities

- Built-in
  - 15
  - Weighted

```
// NodePreferAvoidPodsPriority defines the name of prioritizer function that priorities nodes according to
// the node annotation "scheduler.alpha.kubernetes.io/preferAvoidPods".
NodePreferAvoidPodsPriority = "NodePreferAvoidPodsPriority"
// NodeAffinityPriority defines the name of prioritizer function that prioritizes nodes which have labels
// matching NodeAffinity.
NodeAffinityPriority = "NodeAffinityPriority"
// TaintTolerationPriority defines the name of prioritizer function that prioritizes nodes that marked
// with taint which pod can tolerate.
TaintTolerationPriority = "TaintTolerationPriority"
// ImageLocalityPriority defines the name of prioritizer function that prioritizes nodes that have images
// requested by the pod present.
ImageLocalityPriority = "ImageLocalityPriority"
// ResourceLimitsPriority defines the nodes of prioritizer function ResourceLimitsPriority.
ResourceLimitsPriority = "ResourceLimitsPriority"
// EvenPodsSpreadPriority defines the name of prioritizer function that prioritizes nodes
// which have pods and labels matching the incoming pod's topologySpreadConstraints.
EvenPodsSpreadPriority = "EvenPodsSpreadPriority"
// EqualPriority defines the name of prioritizer function that gives an equal weight of one to all nodes.
EqualPriority = "EqualPriority"
// MostRequestedPriority defines the name of prioritizer function that gives used nodes higher priority.
MostRequestedPriority = "MostRequestedPriority"
// RequestedToCapacityRatioPriority defines the name of RequestedToCapacityRatioPriority.
RequestedToCapacityRatioPriority = "RequestedToCapacityRatioPriority"
// SelectorSpreadPriority defines the name of prioritizer function that spreads pods by minimizing
// the number of pods (belonging to the same service or replication controller) on the same node.
SelectorSpreadPriority = "SelectorSpreadPriority"
// ServiceSpreadingPriority is largely replaced by "SelectorSpreadPriority".
ServiceSpreadingPriority = "ServiceSpreadingPriority"
// InterPodAffinityPriority defines the name of prioritizer function that decides which pods should or
// should not be placed in the same topological domain as some other pods.
InterPodAffinityPriority = "InterPodAffinityPriority"
// LeastRequestedPriority defines the name of prioritizer function that prioritize nodes by least
// requested utilization.
LeastRequestedPriority = "LeastRequestedPriority"
// BalancedResourceAllocation defines the name of prioritizer function that prioritizes nodes
// to help achieve balanced resource usage.
BalancedResourceAllocation = "BalancedResourceAllocation"
// NodePreferAvoidPodsPriority defines the name of prioritizer function that priorities nodes according to
// the node annotation "scheduler.alpha.kubernetes.io/preferAvoidPods".
NodePreferAvoidPodsPriority = "NodePreferAvoidPodsPriority"
```

Priorities

- Default
  - 8
- Weighted

```
func defaultPriorities() sets.String {
        return sets.NewString(
                priorities.SelectorSpreadPriority,
                priorities.InterPodAffinityPriority,
                priorities.LeastRequestedPriority,
                priorities.BalancedResourceAllocation,
                priorities.NodePreferAvoidPodsPriority,
                priorities.NodeAffinityPriority,
                priorities.TaintTolerationPriority,
                priorities.ImageLocalityPriority,
        )
}
```

Priorities in Details

- SelectorSpreadPriority
  - Spreads Pods across hosts, considering Pods that belonging to the same Service, StatefulSet or ReplicaSet.
- InterPodAffinityPriority
  - Computes a sum by iterating through the elements of weightedPodAffinityTerm and adding “weight” to the sum if the corresponding PodAffinityTerm is satisfied for that node; the node(s) with the highest sum are the most preferred.
- LeastRequestedPriority
  - Favors nodes with fewer requested resources. In other words, the more Pods that are placed on a Node, and the more resources those Pods use, the lower the ranking this policy will give.
- MostRequestedPriority
  - Favors nodes with most requested resources. This policy will fit the scheduled Pods onto the smallest number of Nodes needed to run your overall set of workloads.
- RequestedToCapacityRatioPriority
  - Creates a requestedToCapacity based ResourceAllocationPriority using default resource scoring function shape.
- BalancedResourceAllocation
  - Favors nodes with balanced resource usage.

Priorities in Details

- NodePreferAvoidPodsPriority
  - Priorities nodes according to the node annotation scheduler.alpha.kubernetes.io/preferAvoidPods. You can use this to hint that two different Pods shouldn’ t run on the same Node.
- NodeAffinityPriority
  - Prioritizes nodes according to node affinity scheduling preferences
- TaintTolerationPriority
  - Prepares the priority list for all the nodes, based on the number of intolerable taints on the node. This policy adjusts a node’ s rank taking that list into account.
- ImageLocalityPriority
  - Favors nodes that already have the container images for that Pod cached locally.
- ServiceSpreadingPriority
  - For a given Service, this policy aims to make sure that the Pods for the Service run on different nodes. It favoring scheduling onto nodes that don’ t have Pods for the service already assigned there. The overall outcome is that the Service becomes more resilient to a single Node failure.
- CalculateAntiAffinityPriorityMap
  - This policy helps implement pod anti-affinity.
- EqualPriorityMap
  - Gives an equal weight of one to all nodes.

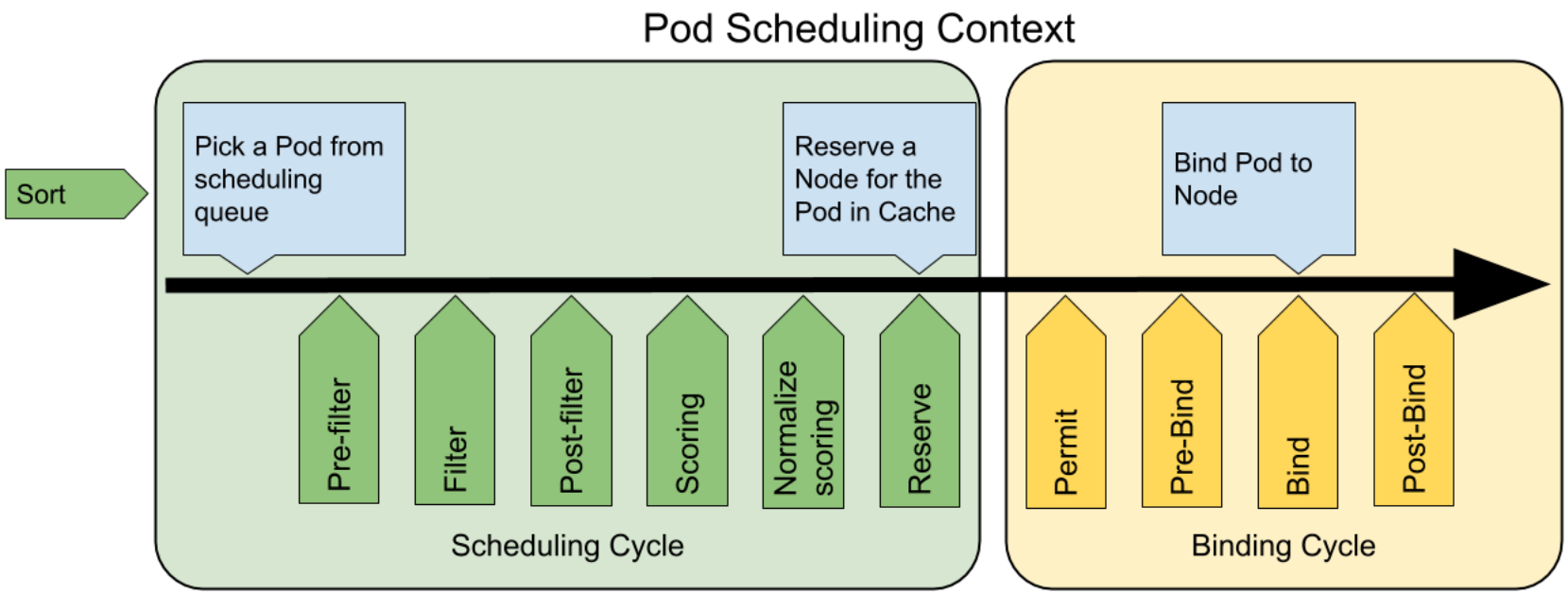
Scheduler Performance

- Percentage of Nodes to Score
  - From 1.12, allow scheduler to stop looking for more feasible nodes once it finds a certain number of them.
  - percentageOfNodesToScore
    - [0, 100]
    - From 1.14, default value provided, >= 5%
  - Round robin iteraition

New Scheduling Framework

- Issues of current framework
  - Limited extension points
  - Based on http(s)
  - Pull communication
  - Isolated process

Extension Points



Extension Points

- Queue sort
  - These plugins are used to sort pods in the scheduling queue. A queue sort plugin essentially will provide a "less(pod1, pod2)" function. Only one queue sort plugin may be enabled at a time.
- Pre-filter
  - These plugins are used to pre-process info about the pod, or to check certain conditions that the cluster or the pod must meet. A pre-filter plugin should implement a PreFilter function, if PreFilter returns an error, the scheduling cycle is aborted. Note that PreFilter is called once in each scheduling cycle..
- Filter
  - These plugins are used to filter out nodes that cannot run the Pod. For each node, the scheduler will call filter plugins in their configured order. If any filter plugin marks the node as infeasible, the remaining plugins will not be called for that node. Nodes may be evaluated concurrently, and Filter may be called more than once in the same scheduling cycle.

Extension Points

- Post-filter
  - This is an informational extension point. Plugins will be called with a list of nodes that passed the filtering phase. A plugin may use this data to update internal state or to generate logs/metrics.
- Scoring
  - These plugins have two phases:
    - Score: which is used to rank nodes that have passed the filtering phase. The scheduler will call Score of each scoring plugin for each node.
    - Normalize scoring: which is used to modify scores before the scheduler computes a final ranking of Nodes, and each score plugin receives scores given by the same plugin to all nodes in "normalize scoring" phase.

Extension Points

- Reserve
  - This is an informational extension point. This happens before the scheduler actually binds the pod to the Node, and it exists to prevent race conditions while the scheduler waits for the bind to succeed.
  - This is the last step in a scheduling cycle. Once a pod is in the reserved state, it will either trigger Un-reserve plugins (on failure) or Post-bind plugins (on success) at the end of the binding cycle.
- Permit
  - These plugins are used to prevent or delay the binding of a Pod. A permit plugin can do one of three things.
    - approve
    - deny
    - wait (with a timeout)
  - Approving a pod binding

Customer Scheduler


```
#!/bin/bash
SERVER='localhost:8001'
while true;
do
    for PODNAME in $(kubectl --server $SERVER get pods -o json | jq '.items[] | select(.spec.schedulerName
;
    do
        NODES=($(kubectl --server $SERVER get nodes -o json | jq '.items[].metadata.name' | tr -d '"'))
        NUMNODES=${#NODES[@]}
        CHOSEN=${NODES[$[ $RANDOM % $NUMNODES ]]}
        curl --header "Content-Type:application/json" --request POST --data '{"apiVersion":"v1", "kind": "B
: "Node", "name": "'$CHOSEN'"}}' http://$SERVER/api/v1/namespaces/default/pods/$PODNAME/binding/
        echo "Assigned $PODNAME to $CHOSEN"
    done
    sleep 1
done
```

Homework III

- Requirements
  - Deploy a Kubernetes cluster with at least 2 nodes, design a experiment to evaluate built-in predicates and priorities
  - Write your own scheduler, design a experiment to evaluate it
    - Predicates
    - Priorities
  - 1p, cluster, 1 built-in predicate
  - 3p, cluster, 1 built-in predicate or priority, 1 your own predicate or priority
  - 4p, cluster, 1 built-in predicate or priority, 1 your own predicate and 1 priority
  - 5p, cluster, 1 built-in predicate and 1 priority, 1 your own predicate and 1 priority

Homework III

- Deliverables
  - Scripts, codes, evaluation report
  - GitHub repo
- Deadline
  - Nov 20, 2019

Reference

- https://kubernetes.io/docs/tutorials/kubernetes-basics/
- https://kubernetes.io/docs/tutorials/online-training/overview/
- https://www.awseducate.com/registration#INFO-Student
- Snap, https://microk8s.io/
  - Microk8s.add-node, microk8s.join, microk8s.remove-node, microk8s.leave