Kubernetes scheduler development

An overview

Cristiano Colangelo
30/06/2022

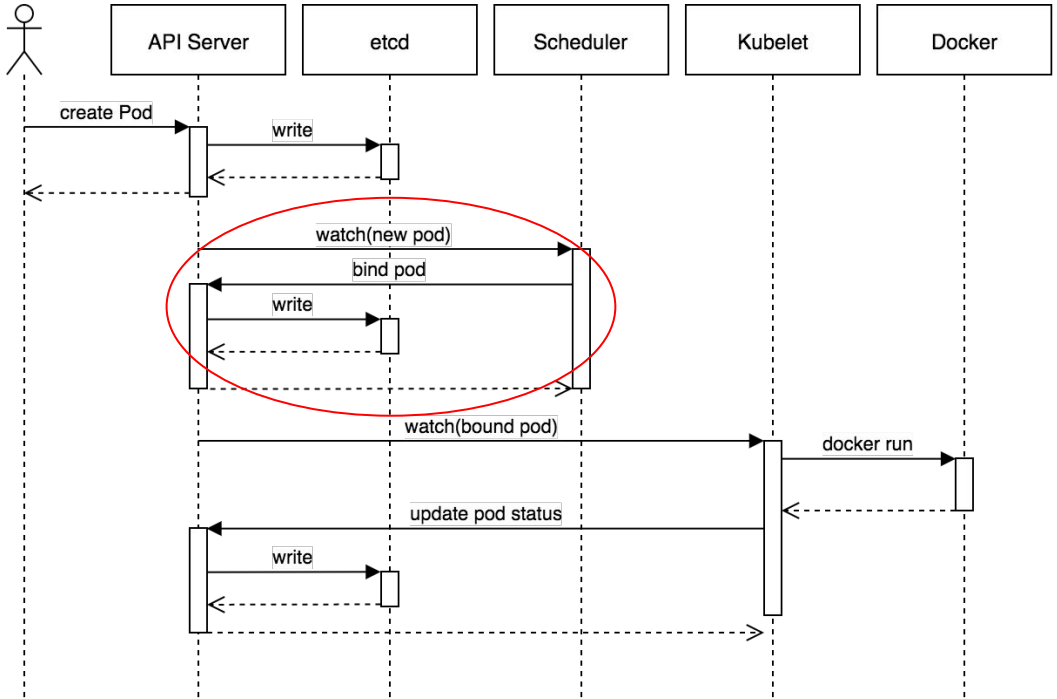
helio

Index

Introduction

- Special type of controller watching Pod resources
- Can be replaced or extended with multiple schedulers
- Assigns Pods to Nodes based on filters and scores
- Node with highest score will be selected to run the relative Pod
- Pods can be preempted (descheduled) if necessary

Introduction – The birth of a Pod



Introduction – Hard vs soft constraints

- Hard constraints
  - nodeSelector
  - Taints and tolerations
- Soft constraints
  - PreferNoSchedule
  - nodeAffinity
  - podAffinity

Deployment of new scheduling rules

3 options:

1. “Scheduler Framework” – plugins installed in the default scheduler
2. Implement scheduler process to replace or run alongside default one
3. Scheduling extensions – webhooks called by scheduler during final scheduling phase

Config registered using a **KubeSchedulerConfiguration** resource.

Config is passed as CLI argument to the scheduler.

Deployment of new scheduling rules

- Option 1 is a compromise between 2 and 3
  - Keep default scheduler implementation; extend it with plugins
  - Work on some established extension points
  - Must compile the plugin into the scheduler binary
  - Does not require a fork of the default scheduler

Deployment of new scheduling rules

- Option 2 is the most versatile and powerful way, but
  - Requires more maintenance to keep it up to date with K8s API versions
  - The new scheduler should significantly differ from the default one

Deployment of new scheduling rules

- Option 3 has some drawbacks
  - Worse performance (HTTP overhead, more computations)
  - Cache inconsistency
  - Limited extension points
- … but also some advantages
  - Can extend existing scheduler without recompilation
  - Can work with different versions of kube-scheduler

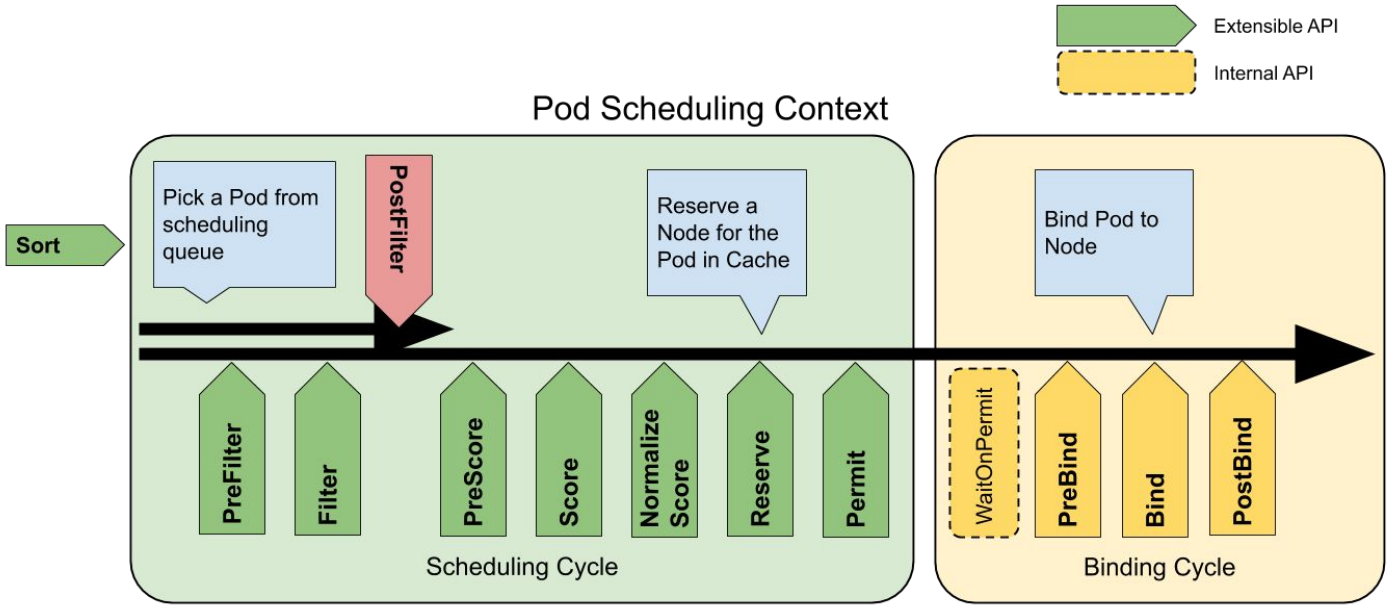
Deployment of new scheduling rules

- Option 1 or 2 is better when
  - Care about performance
  - Want maximum ability to customize the scheduler's behavior
- Option 3 is better when
  - Don't care about latency and cache consistency
  - Don’t care about having access to all extension points
  - Don't want to maintain own build of the default scheduler

Scheduling framework

- Plugins add scheduling behaviors to the scheduler
- Plugins are included at compile time of the scheduler
- Plugins can be registered at one or more extension points
- The scheduler's ComponentConfig allows plugins to be enabled and ordered

Extension points



Not exposed

E.g. Score plugin priority function (score = priority)

For example, suppose a plugin `BlinkingLightScorer` ranks Nodes based on how many blinking lights they have.

```
func (*BlinkingLightScorer) Score(state *CycleState, _ *v1.Pod, nodeName string) (int, *Status) {
   return getBlinkingLightCount(nodeName)
}
```

Scheduling extensions

- Used by the Intel Scheduler
- Extender allows **external processes** to filter and prioritize nodes
- One HTTP(s) call for “filter” and one for “prioritize”
- In addition “**bind**” call to bind the Pod to apiserver

Scheduling extensions

- Need **Scheduler policy** to configure the **communication** with an extender
  - How to reach extender (URL prefix) and its verbs (filter, prioritize, bind)
  - Weight, a multiplier on the score generated by prioritize
  - Use HTTP or HTTPS, TLS config, timeout
  - Optionally which resources Pods should be managed by scheduler (selector)

Conclusions

- Significant amount of work to create plugin/extender/from scratch
  - Great extensibility
- Just use the Intel Telemetry Aware scheduler? Fork it?
  - Less work but less flexibility
- Scheduler framework plugin + operator pattern
  - More work but more flexibility

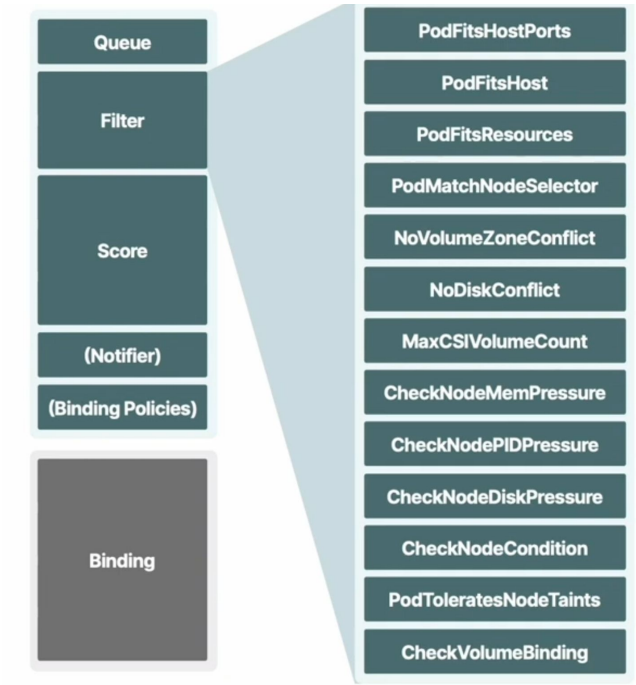
Outlook

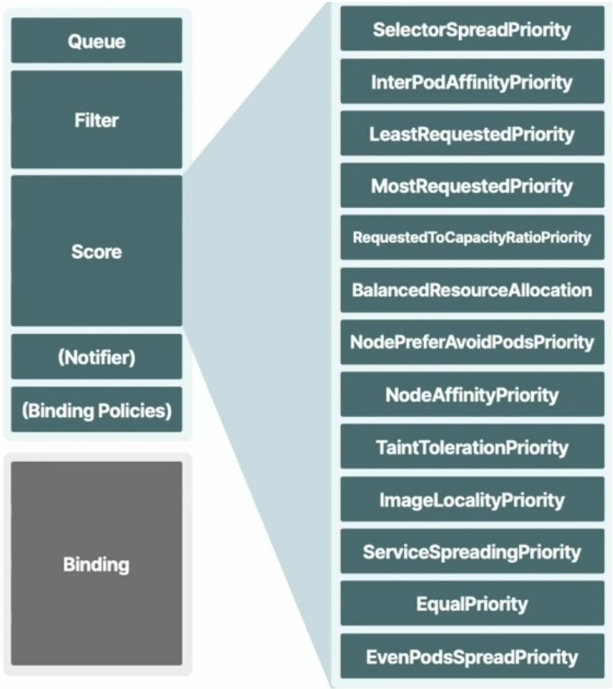
- Need to agree on formal contract between system boundaries asap
- I suggest we start from here – installing a dummy scheduler plugin
- Create Git repo
- In the meantime also have a look at Trimaran Real Load Aware plugin

Outlook

- Preemption (descheduling) maybe more work than scheduling
  - Control eviction of Pods
  - Merge workloads, defragmentation
  - Alternative to descheduling: In-place update of Pod resources (planned for v1.25, 23rd August)

Appendix - Filtering & Scoring





Appendix - Extenders calls

Filter

- **Input**: set of nodes filtered through the k8s predicates + Pod being scheduled (`schedulerapi.ExtenderArgs`)
- **Output**: the filtered set of nodes (`schedulerapi.ExtenderFilterResult`)

Prioritize (Score)

- **Input**: set of nodes filtered through the filter call + Pod being scheduled (`schedulerapi.ExtenderArgs`)
- **Output**: list of all priorities (`schedulerapi.HostPriorityList`)
- Scores returned by prioritize are added to the overall scores
- Scores are computed through priority functions
- Resulting score decides on which Node to schedule the Pod

Appendix - Extenders calls

Bind

- **Input**: Pod name, Pod Namespace, Pod UID, Node on which to schedule (`scheduler.ExtenderBindingArgs`)
- **Effect**: binds (schedules) the given Pod to a Node
- Optional
- Used to delegate binding of the Pod to a Node through the Extender
- The Extender becomes responsible for issuing bind call to the apiserver
- When multiple bind calls are issued, the first successful binding goes through, others are ignored

Sources & References

- Scheduling Framework KEP
  https://github.com/kubernetes/enhancements/tree/master/keps/sig-scheduling/624-scheduling-framework
- Older scheduler extender KEP (note that it's not implemented. Look at the Intel Scheduler for an example of a current extender, and official docs and references).
  https://github.com/kubernetes/design-proposals-archive/blob/main/scheduling/scheduler_extender.md
- Up to date scheduler extender KEP (same comment as for the older scheduler extender)
  https://github.com/kubernetes/enhancements/tree/master/keps/sig-scheduling/1819-scheduler-extender
- Scheduler plugins based on the scheduler framework
  https://github.com/kubernetes-sigs/scheduler-plugins
- Official docs for scheduling configuration
  https://kubernetes.io/docs/reference/scheduling/config/

Sources & References

- How Does The Kubernetes Scheduler Work?
  https://www.youtube.com/watch?v=8WdYk4oTnrw
- Building a Kubernetes Scheduler using Custom Metrics
  https://youtu.be/4TaHQgG9wEg