Foundation of Internet Platform Development & Operation

Kubernetes in Details: Customer Scheduler & Coscheduling

2019-11-12

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Customer Scheduler

- Object
  - Customize the scheduling strategies to fit for on-demand business or operation requirements

Customer Scheduler

- Use the upstream source code
  - modify the code in place, and then re-compile to run the "hacked" scheduler
    - Fork the code
    - Build your customized kubernetes

Customer Scheduler

- Run a separate scheduler along with the default scheduler
  - The default scheduler and your custom scheduler covers respective pods exclusively
    - spec.schedulerName
  - Issues
    - Distributed lock
    - Cache synchronization
      - When pods get scheduled onto the same node by multiple schedulers
    - Maintaining a high-quality custom scheduler isn’t trivial

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-scheduler-
  namespace: kube-sys
data:
  config.yaml: |
    apiVersion: compo
    kind: KubeSchedul
    schedulerName: my
    algorithmSource:
      policy:
        configMap:
          namespace:
          name: my-sc
    leaderElection:
      leaderElect: tr
      lockObjectName:
      lockObjectNames
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-scheduler-policy
  namespace: kube-system
data:
 policy.cfg : |
  {
    "kind" : "Policy",
    "apiVersion" : "v1",
    "predicates" : [
      {"name" : "PodFitsHostPo
      {"name" : "PodFitsResour
      {"name" : "NoDiskConflic
      {"name" : "MatchNodeSele
      {"name" : "HostName"}
    ],
    "priorities" : [
      {"name" : "LeastRequeste
      {"name" : "BalancedResou
      {"name" : "ServiceSpread
      {"name" : "EqualPriority"
    ],
    "extenders" : [{
      "urlPrefix": "http://loca
      "filterVerb": "predicates
      "prioritizeVerb": "priorities/zero_score",
      "preemptVerb": "preemption",
      "bindVerb": "",
      "weight": 1,
      "enableHttps": false,
      "nodeCacheCapable": false
    }],
```


```yaml
containers:
- name: my-scheduler-ctr
  image: gcr.io/google_containers/hyperkube:v1.11.1
  imagePullPolicy: IfNotPresent
  args:
  - kube-scheduler
  - --config=/my-scheduler/config.yaml
  - -v=4
  volumeMounts:
  - name: my-scheduler-config
    mountPath: /my-scheduler
- name: my-scheduler-extender-ctr
  image: a/b:c
  imagePullPolicy: IfNotPresent
  livenessProbe:
```

Customer Scheduler

- Scheduler extender
  - The solution to extend scheduler with minimal efforts
  - Compatible with the upstream scheduler
  - Configurable webhooks
    - Filter
    - Prioritize

Kubernetes Scheduler Workflow

- 1. The default scheduler starts up according to the parameters given.
- 2. It watches on apiserver, and puts pods where its spec.nodeName is empty into its internal scheduling queue.
- 3. It pops out a pod from the scheduling queue and starts a standard scheduling cycle.
- 4. It retrieves “hard requirements” from the pod’s API spec
  - Predicates
- 5. It retrieves “soft requirements” from the pod’s API spec and also applies some default soft “policies”
  - Priorities
- 6. It talks to the apiserver and sets spec.nodeName to indicate the node that this pod should be scheduled to

Scheduler Extender

- Startup parameters
  - kube-scheduler
    - --config

```
{
    "kind" : "Policy",
    "apiVersion" : "v1",
    "extenders" : [{
        "urlPrefix": "http://localhost:8888/",
        "filterVerb": "filter",
        "prioritizeVerb": "prioritize",
        "weight": 1,
        "enableHttps": false
    }]
}
```

```
# content of the file passed to "--config"
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: "/var/run/kubernetes/scheduler.kubeconfig"
algorithmSource:
  policy:
    file:
      path: "/root/config/scheduler-extender-policy.json"
```

Scheduler Extender

- Extender
  - http server
  - Endpoints
    - /filter
    - /prioritize
  - Implementation

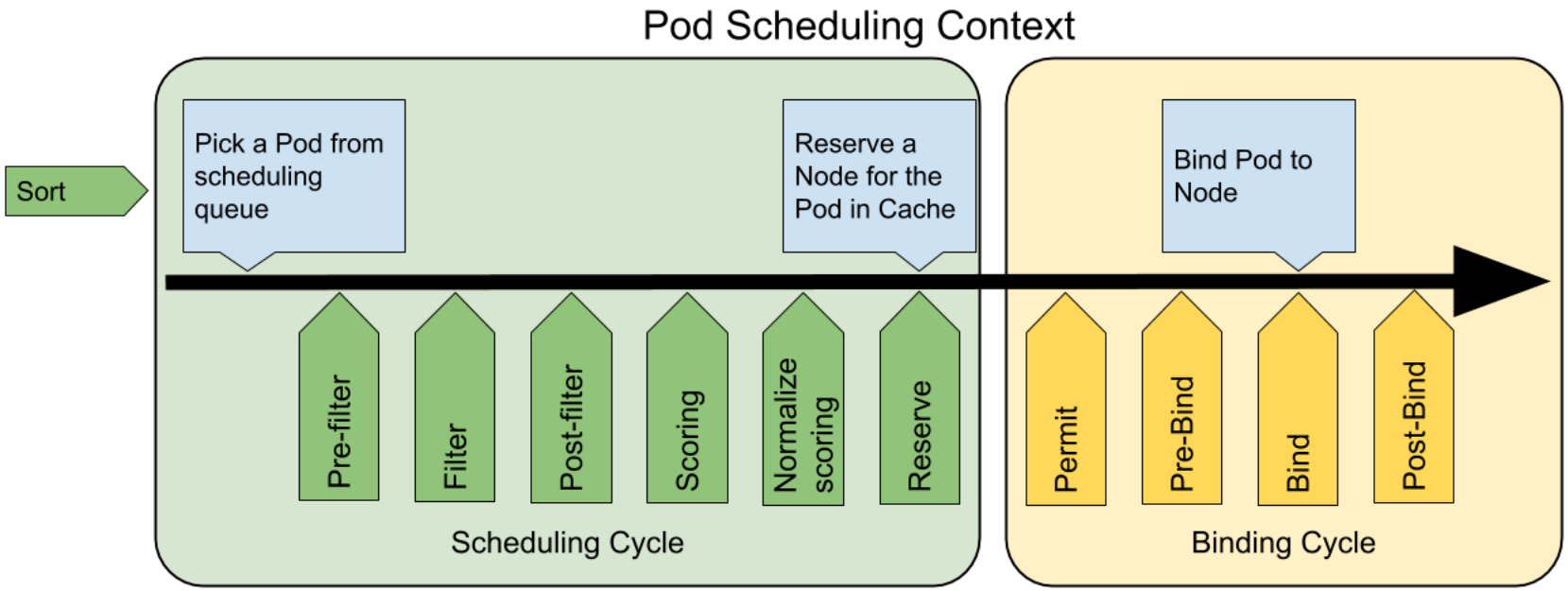
```
func main() {
    router := httprouter.New()
    router.GET("/", Index)
    router.POST("/filter", Filter)
    router.POST("/prioritize", Prioritize)

    log.Fatal(http.ListenAndServe(":8888", router))
}
```

Scheduler Extender

- Communication cost
- Limited extension points
- Subtraction over addition
- Cache sharing

Customer Scheduler

- Scheduler framework



Coscheduling

- Scenario
  - When running a Tensorflow/MPI job, all tasks of a job must be start together; otherwise, did not start anyone of tasks
- kube-batch proposed
  - Focuses on "batch" workload in Kubernetes
  - Share the same scheduling frameworks

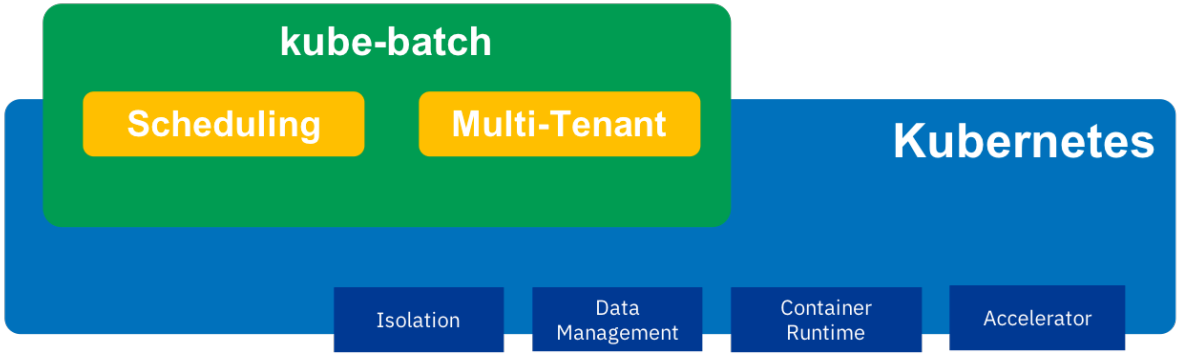
```
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration

...

plugins:
  reserve:
    enabled:
    - name: bar
    - name: foo
    disabled:
    - name: foo
```

Kube-batch

Infra

kube-batch focus on:

- "Batch" scheduling
- Resource sharing between multi-tenant

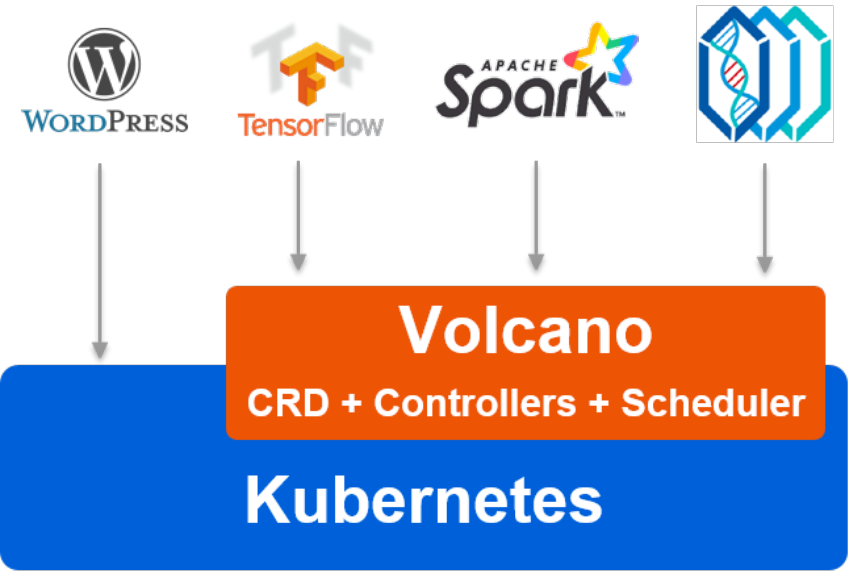


kube-batch **NOT** support:

- Data Management
- Accelerator (Kubelet), e.g. GPU
- Isolation for multi-tenant
- Job Management
- New container runtime, e.g. Singularity, CharlieCloud

Volcano

- Huawei



Domain frameworks:

- Deployment/Installation of framework in k8s
- Map framework's terms/concepts into common concept, e.g. Job, Queue
- Enable related features for frameworks, e.g. gang-scheduling for TensorFlow training

Common Service for high performance workload:

- Batch scheduling, e.g. fair-share, gang-scheduling
- Enhanced job management, e.g. multiple pod template, error handling
- kubectl plugins, e.g. show Job/Queue information