SIG-Scheduling Deep Dive

Da Ma (Huawei, @k82cn)
Wei Huang (IBM, @Huang-Wei)
Scheduler Overview
Scope of Scheduler

- API Server
- Controller Mgr
- Scheduler
- Kubelet
- CRI (Docker)

- create deploy
  - watch deploys
  - add/delete pods

- watch unbound pod
  - bind pod

- watch bound pod

- update pod status

- docker run
Detailed Scheduling Flow

1. **Scheduler**
   - **Internal Scheduling PQ**
     - Active Q
     - Backoff Q
     - Unscheduled Q

2. **API Server**

3. **Scheduler Informers**

4. **Main Scheduling Goroutine**

5. **Binding Goroutine**

- **watch**
- **add unbound pod**
- **Update internal cache**
- **cluster snapshot()**
- **Self Reconcile**
- **Pop pod**
- **(a) Put back pod**
- **(b) Post a binding req**
- **bind pod to a node**
Predicates - filter out nodes
For incoming pod, gives a **YES** or **NO** answer whether it fits on a node or not.

For each node, check if the node fits the incoming pod (running in Parallel):

For each sorted predicate: (running in batch)

Run the predicate function:

- If succeeded, continue to next predicate
- If failed, record fail reason, break

If failure can be resolved by Preemption, proceed with Preemption

Aggregate fail reason(s), and return if it’s a fit

Filter out failed nodes, and get a node list that fits for incoming node

---

1 Prior to 1.10, the behavior is continue to next predicate. Starting 1.10, the default behavior was optimized to “break” the predicates loop (see PR 56926). And it’s configurable by parameter “AlwaysCheckAllPredicates” - which is useful for debugging.
When a pod can’t be scheduled, starts to preempt: blue == dry-run operation
For each node, delete pods which has lower priority. Based on this state, check:
  If incoming pod cannot be scheduled, return;
  If incoming pod can be scheduled:
    For each pod with lower priority than incoming pod, start with higher one:
      Add the pod. Check:
      If incoming pod can still be placed, keep current pod. Continue
      If not, remove current pod, and add current pod to a “candidates” list
    Return the “candidates” list
If len(candidates) !=0, pick a node which a series of candidate victims belong to, and talk to APIServer to “reserve” that node (set NominatedNode) for incoming pod, then delete the candidate victims.
Priorities - rank the remaining nodes
For incoming pod, give a score (0~10) on each filtered node after Predicates phase. Each priority is defaulted weight 1.

Implemented using map/reduce pattern.

For each filtered node:
  For each priority:
    Calculate a score, then multiple its weight
    Final score for current node = sum(score * weight)
Get a final score list on all nodes
Pick up the node which has highest final score.
Design Rationale of Scheduler

1. The scheduler is **NOT** responsible for managing life cycle of Pods.
2. The minimum scheduling unit is **POD** (tried EquivalenceCache, but not good as supposed to be)
3. Schedule **one pod** at a time (scheduler can have multiple replicas, but only one leader is running)
4. **Best Fit** vs. First Fit
5. **Predicates** and **Priorities**
6. **Configurable** (schedule config file)
7. **Plugable** (new scheduler framework, scheduler extender, multiple schedulers)
Recent Developments
Performance improvements
GA: Priority and Preemption

Planned features
● Even Pod Spreading
● Scheduler Framework

Overview of kube-batch

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, Chaos Cloud
Overview of kube-batch (cont.)

- **kube-apiserver**
  - QueueJob
    - QueueJobSpec
      - PodGroup
        - Pod
  - CRDJob
    - CRDJobSpec
      - PodGroup
        - Pod

- **Cache**
  - JobInfo
    - PodGroup
      - Pod
    - NodeInfos

- **OpenSession**
  - predicate
    - allocate
      - preempt
        - Backfill
          - CloseSession

- **DRF plugin**
  - JobOrderFn
  - PreemptableFn

- **Priority plugin**
  - JobOrderFn
    - TaskOrderFn
    - PreemptableFn

- **Gang plugin**
  - JobOrderFn
    - PreemptableFn
    - JobReadyFn

- **Reconstruct JobInfo in Cache by PodGroup**

- **Predicate, allocate, preempt are Actions, and they’re pluggable**

- **Plugins on demand**
Features of kube-batch

- Co-scheduling
- “Fair-sharing” (job/queue)
- Preemption/Reclaim
- Task Priority within Job
- Predicates
- Queue
- Backfill (partially)
- Dynamic configuration

Batch Capability into Kubernetes (#68357)
Poseidon/Firmament scheduler augments the current Kubernetes scheduling capabilities by incorporating a new novel flow network graph based scheduling capabilities alongside the default Kubernetes Scheduler.

Firmament models workloads on a cluster as flow networks and runs min-cost flow optimizations over these networks to make scheduling decisions.
Features of Poseidon

1. Node level Affinity and Anti-Affinity
2. Pod level Affinity and Anti-Affinity
3. Taints & Tolerations
4. Gang Scheduling
How those schedulers work together ???
Sorry, I don't know :(
Multi-Schedulers

Option 1

Option 2

accept

reject
Multi-Schedulers

Option 3

Option 4
Trigger Of Pod Movement/Migration

Eviction -> Creation -> Re-schedule
User Cases of Descheduler

- Some nodes are under or over utilized.
- The original scheduling decision does not hold true any more, as taints or labels are added to or removed from nodes, pod/node affinity requirements are not satisfied any more.
- Some nodes failed and their pods moved to other nodes.
- New nodes are added to clusters.
Policy & Strategy

- RemoveDuplicates
- LowNodeUtilization
- RemovePodsViolatingInterPodAntiAffinity
- RemovePodsViolatingNodeAffinity
Pod Eviction Restriction

• Critical pods (with annotations scheduler.alpha.kubernetes.io/critical-pod) are never evicted.
• Pods (static or mirrored pods or stand alone pods) not part of an RC, RS, Deployment or Jobs are never evicted because these pods won't be recreated.
• Pods associated with DaemonSets are never evicted.
• Pods with local storage are never evicted.
• Best efforts pods are evicted before Burstable and Guaranteed pods.
• Pod are never evicted If violates its PDB
Contact Us
Contact Us

Chairs

- @bsalamat
- @k82cn

Home page: https://github.com/kubernetes/community/tree/master/sig-scheduling

Slack channel: https://kubernetes.slack.com/messages/sig-scheduling

Mail list: https://kubernetes.slack.com/messages/sig-scheduling

Google doc:
https://docs.google.com/document/d/13mwye7nvrmV11q9_Eg77z-1w3X7Q1GTbslpml4J7F3A/view