Understanding Scalability and Performance in the Kubernetes Master

Xingyu Chen, Fansong Zeng

Alibaba Cloud
Agenda

- Background
- Kubernetes in Alibaba
- Experience on scalability
  - etcd/apiserver/controller
- Q&A
Background

It is a long story...

AI
Zeus
Hippo
AliSwarm
Sigma
Kubernetes

2013 2015 2017 2018
Kubernetes in Alibaba

- Production environment
  - 10,000s of applications
  - 1,000,000s of containers
  - 10s of clusters
  - 100,000s of nodes
  - 10,000 nodes / largest cluster
Experience on scalability

A 10k nodes cluster

- # Pods: ~200k
- # Objects: ~1000k
- # Latency: ~10s
Experience on scalability

1 read/write latency spike
2 too many request ddos
3 unable to write when hitting storage limit

Scheduler throughput low
get pods/nodes latency very slow
Controller can’t catch up
Fail restart slowly
• Boost storage limit
• Level up data isolation
Boost performance of single etcd cluster

Segregated hashmap algorithm
Efficient node heartbeats

- Kubelet send heartbeats every 10s
- 15kb with tens of images and volumes

transaction log \(\sim 1\text{GB/min}\)

etcd

CPU overload larger than 80%
Efficient node heartbeats

- Add a new `Lease` build-in API

![Diagram showing the interaction between Kubelet, Node Object, Node Controller, and Lease]

- Update every 10s
- Node Object
- Grace period 40s
- Lease objects ~100 bytes
- Enabled @ 1.14
API Server load balancing

- 3 nodes HA cluster

![Diagram showing master nodes (etcd, apiserver, scheduler, controller) and slave nodes (kublet, autopilot, pouch). System daemon and containers are also shown. Chart on the right indicates CPU usage with upgrades and failures.]
API Server load balancing

- Add a load balancer?

1. Expand three `apiserver` services and link each to a load balancer (lb).
   - `apiserver` to `kubelet`

2. Expand `apiserver` services and link each to a load balancer (lb) using a direct connection.
   - `apiserver` to `kubelet`
API Server load balancing

- Throttling:
  1. Send 'too many requests' when exceeded the low
  2. Send 'Connection: close' when exceeded the high

- Retry:
  1. Try another server if there are too many 'too many requests'
  2. Try another server after several minutes

Upgrade with {maxSurge=3}
• Key communication mechanisms between client and server

updates

5 pod a
6 node b
7 pod c

etcd

API Server

Cache

Informer

Store

Reflector

deltas

watch

Reflector

watch

Controllers/Kubelets

Alibaba Cloud
- What happens if the connection is broken?

```
list = [1, 2, 3, 4, 5]
watch = [2, 4, 5, 7, 8, 9]
deltas = [3, 5]
rv = 5
```

- pods are reflected to the cache.
- When the connection is broken, the cache becomes stale.
- The pods are older than the version in the cache.
- The `watch` function is triggered to update the cache.

Diagram:
- API Server
- Cache
- Store
- Reflector
- Kubelets
- Too old version error
- pods
- API Server Queue
- Watch `bookmark`

**Reduce to 3%** **released at 1.15**

- **API Server**
  - **Queue**
  - **Cache**
  - **Store**
  - **Informer**
  - **Controllers/Kubelets**

**API Server Queue**
- `rv=12` released at 1.15
- `rv=12` reduced to 3%

**API Server**
- `rv=5` deltas
- `rv=2` pods
- `rv=4` pods
- `rv=5` pods
- `rv=7` pods
- `rv=8` pods
- `rv=9` pods
- `rv=11` pods
- `rv=12` pods

**List-Watch & Cacher**
List-Watch & Cacher

- Improve cache to serve read requests

1. Unable to support indexing
2. Page reading
List-Watch & Cacher

- Improve cache to serve read requests

1. request current rv@t0
2. waiting for updates

---

etcd

`watch`

Reflector

Cache

deltas

API Server

API Server

t0

Client

Get/List

Add indexers:
1. nodename
2. namespace
3. labels
......

---

describe node

5s

0.3s
Controller failover

- Restarting controllers takes several minutes

1. Millions of objects
2. Several GBs

MTTR Reduce to several seconds

API Server

Cache
deltas

Informer

Reflector

Store

Reflector

Release Lease on SIGTERM

Controllers

Informer

Controllers
THANK YOU
● Equivalence classes

● Relaxed randomization