KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

A Story of Managing K8s Watch Events End-to-end Flow in Extremely Large Clusters

Bo Tang, Ant Group

Aug, 2024

Outline

- The Watch Mechanism and The Importance of Watch in Kubernetes
- Definition of Watch SLO and What It Brings to Us
- What We Did to Optimize Watch Flow and the Benefits We Got
- Summary and Future Plans

Watch Mechanism

Watch SLO

Optimization

Summary

K8s Overview

Kubernetes is getting boring?

Let us investigate K8s more

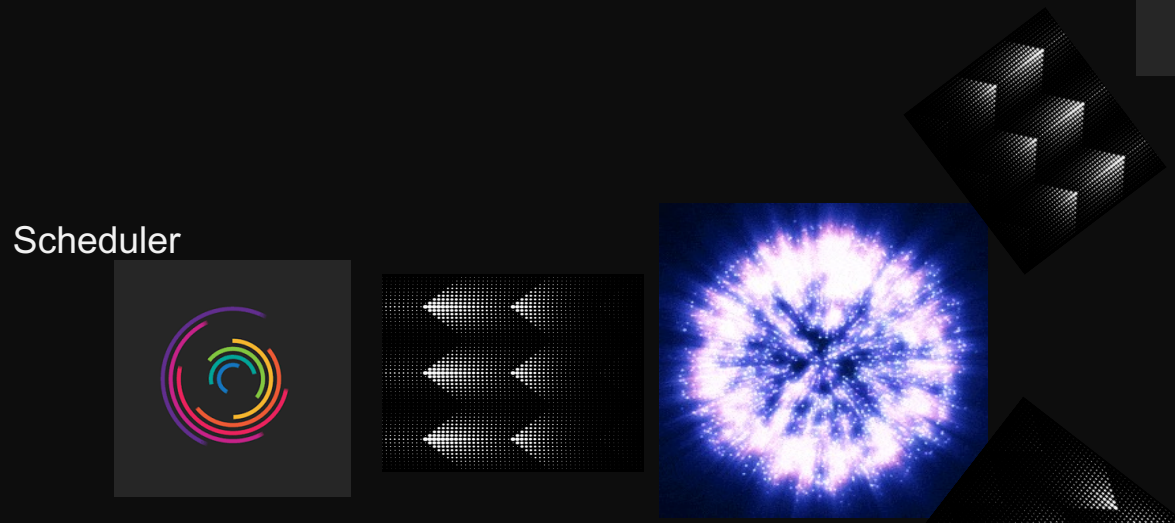
Kubernetes is Boring And the Future is Exciting

- Stable Skeleton
- Comprehensive extensibility
- Design Philosophy

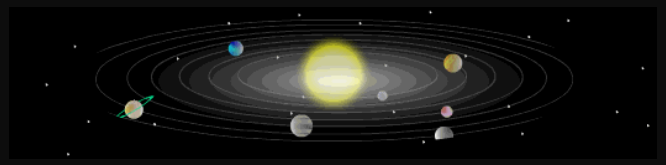
You will get what you want

K8s Birds-eye's View

Controllers

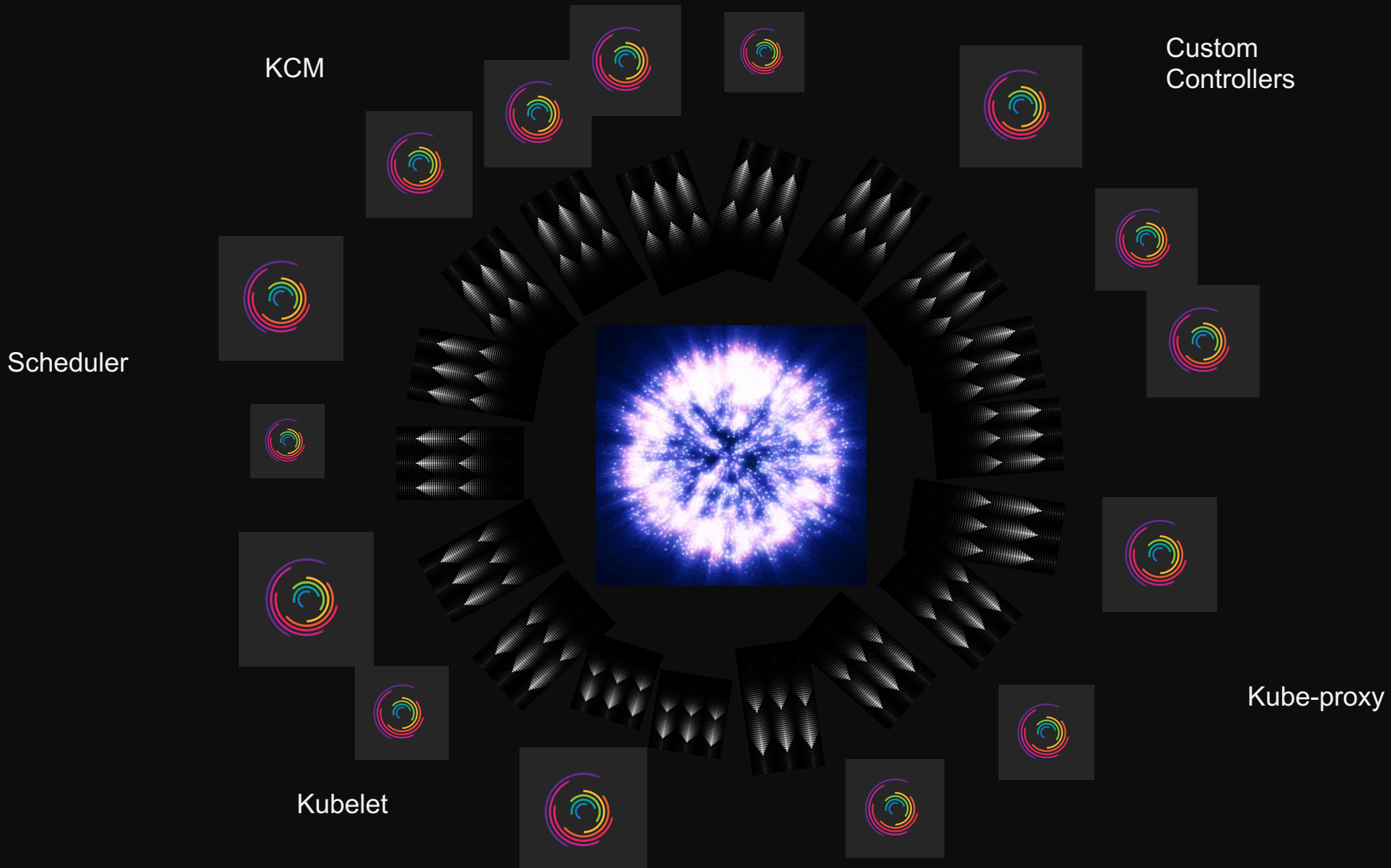


Scheduler



Kubelet

When Cluster Gets Large



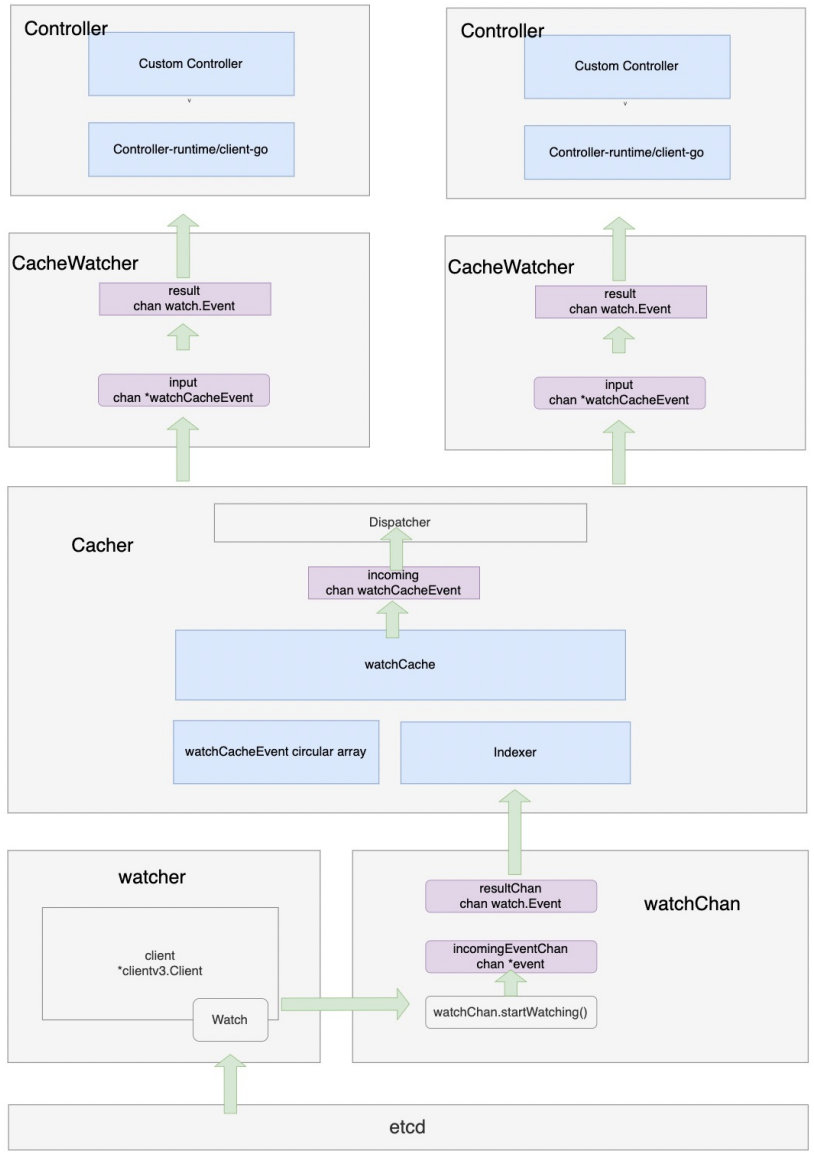
When Cluster Gets Large

Problems arise when cluster gets large:

Large number of nodes and pods.
Large number of CRDs
Heavy traffic
high churn rate of resources.
The provisioning path is long
Different usages of different controllers
Users requirements diverse/vary a lot

Almost every problem is connected to Watch mechanism.
We've done optimization for each component of Kubernetes system, but the link/connection part seems missing.

Watch Procedure



1. **Apiserver** obtains data from etcd, decodes it, formats it, and sends it to apiserver's internal cache.

2. Based on the watched data, apiserver builds its cache internally, which is called the **WatchCache** object. It includes a full set of data and a circular array.

    * The full set of data is a set of data that includes indexing and is used for various client list requests, list-by-label requests, and list-by-field requests.

    * The circular array contains the latest watchCache event and is used for various client watch requests.

3. The apiserver internally **iterates** through all of the cacheWatchers for a resource and sends each cacheWatcher the change events it is interested in one by one.

4. When the client receives data from apiserver, it performs decoding, and puts it into the client's cache. At the same time, generate corresponding **add/update/delete** events and put them into the workqueue.

5. The client's user code reads the workqueue and performs corresponding reconcile **reconciliation** processing.

Watch Mechanism

Watch SLO

Optimization

Summary

K8s SLO Definition

Steady state SLIs/SLOs

Status	SLI	SLO	User stories, test scenarios, ...
Official	Latency of processing mutating API calls for single objects for every (resource, verb) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, verb) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day[1] <= 1s	Details
Official	Latency of processing non-streaming read-only API calls for every (resource, scope) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, scope) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day[1] (a) <= 1s if `scope=resource` (b) <= 30s otherwise (if `scope=namespace` or `scope=cluster`)	Details
Official	Startup latency of schedulable stateless pods, excluding time to pull images and run init containers, measured from pod creation timestamp to when all its containers are reported as started and observed via watch, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, 99th percentile per cluster-day[1] <= 5s	Details

https://github.com/kubernetes/community/blob/master/sig-scalability/slos/slos.md

Guarantee

User stories

- As a user of vanilla Kubernetes, I want some guarantee how quickly my pods will be started.

User stories

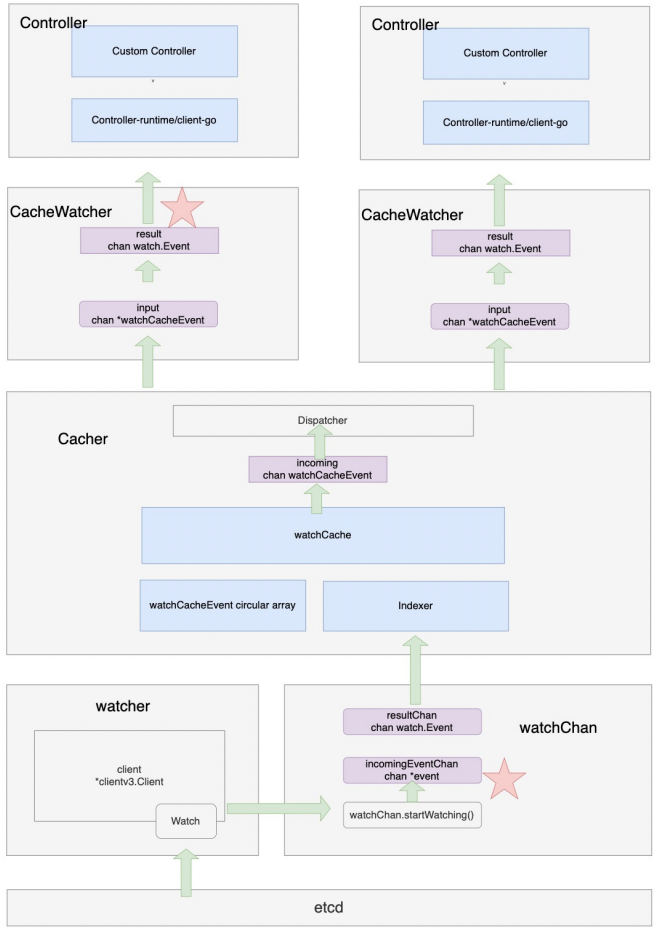
- As a user of vanilla Kubernetes, I want some guarantee how quickly I get the response from an API call.
- As an administrator of Kubernetes cluster, if I know characteristics of my external dependencies of apiserver (e.g custom admission plugins, priority & fairness configuration, and webhooks). I want to be able to provide guarantees for API calls latency to users of my cluster.

Watch SLO

SLI：The SLI is defined as the duration from **the time an event enters the APIserver until it leaves the APIserver**, effectively measuring as the ApiserverTimeSpan. Measured as P99 over last 1 min.

SLO：The SLO is defined as the latency of all events from entering the API server until they exit the API server. Within a rolling window of 1min, the 99th percentile of this latency time should be less than 1 second, which is mathematically represented as P99(ApiserverTimeSpan) < 1s.

ApiserverTimeSpan
Calculation：ApiserverTimeSpan(event) = T(event leaves apiserver) - T(event enters apiserver)



Fill in the last missing piece of K8s SLO puzzle

Watch SLO

Based on the watch SLO, we discovered many problems while operating on of the largest K8s cluster. Nearly any type of failure of large cluster would manifest in this SLO.

These faults include, but are not limited to the following:

- Performance issue of apiserver
- Client-side CPU/mem bottleneck
- Client-side code problem(heavy logic/lock)
- Slow encoding/decoding for built-in resource/CRD
- Client Watch RV = 0
- Massive re-list problem
- Apiserver or load balancer full bandwidth problem
- The issue of multiple clients sharing a single TCP connection leading to a full buffer

A perfect indicator of cluster heath

When users see an increase in Watch RT, they will say “Watch Delay”.

In fact, this is not “Watch Delay”. It is indicating part of the cluster is not in a healthy state.

Watch Mechanism

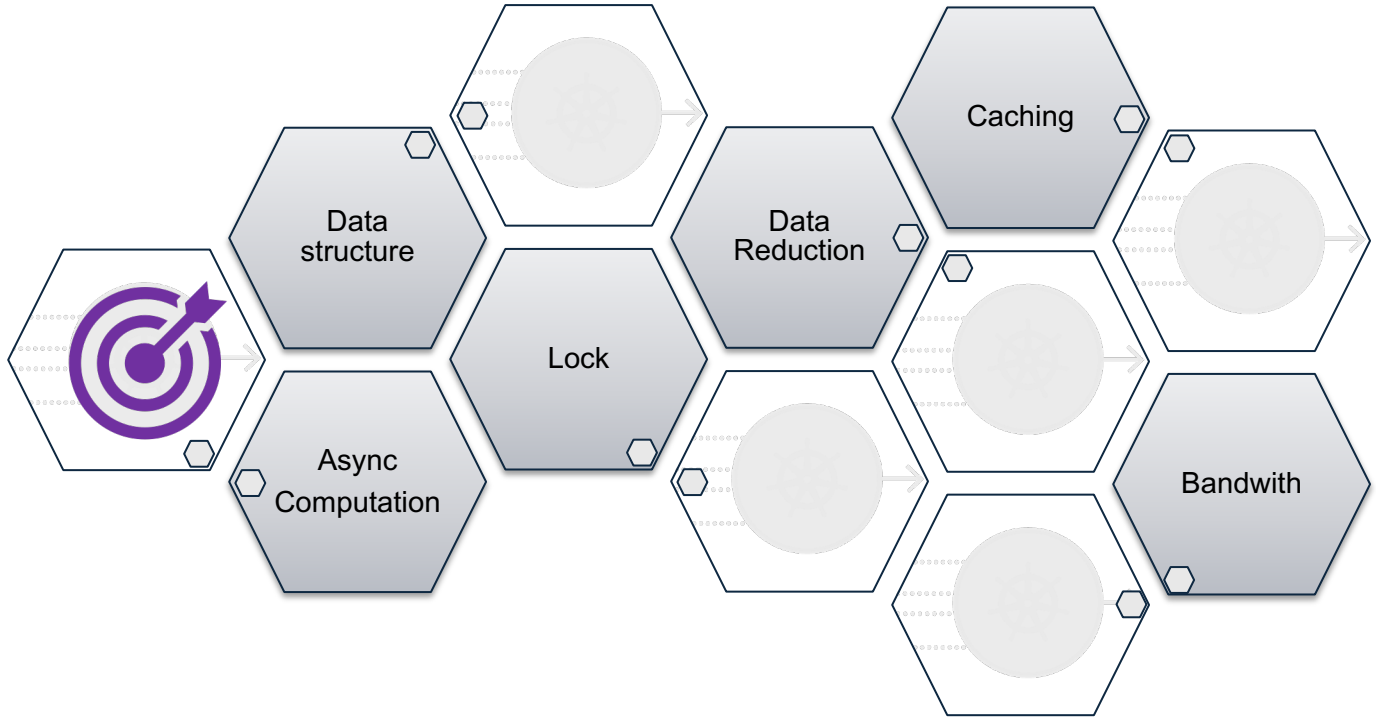
Watch SLO

Optimization

Summary

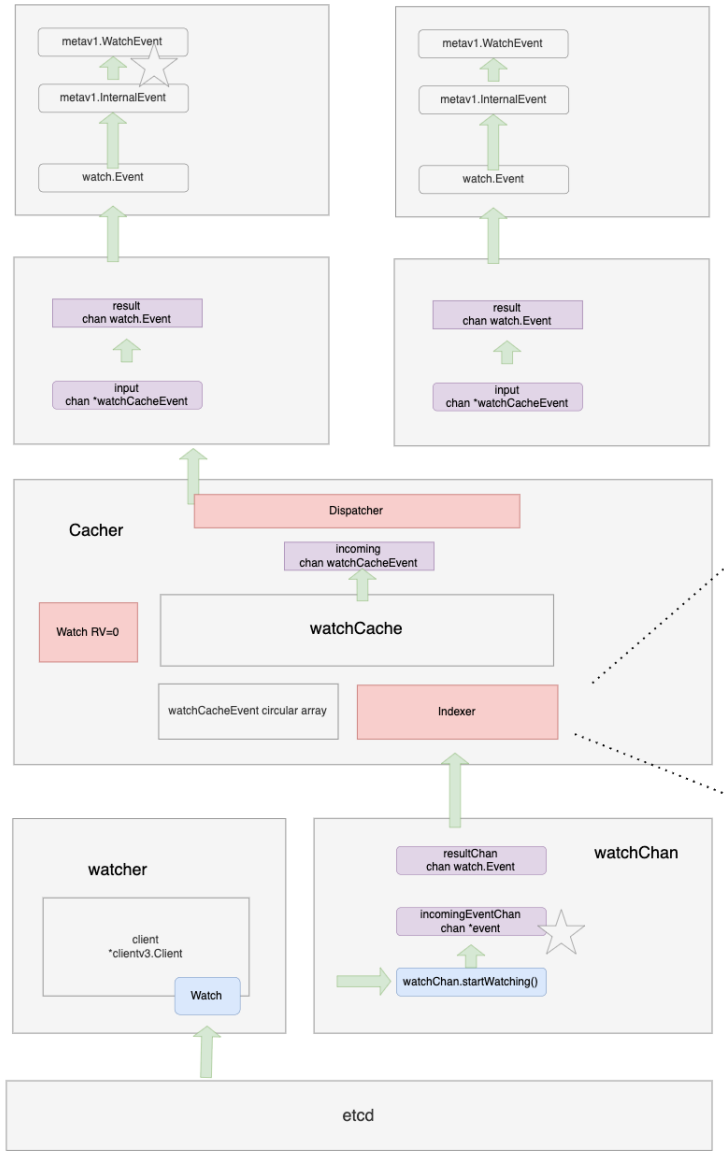
Watch Optimization

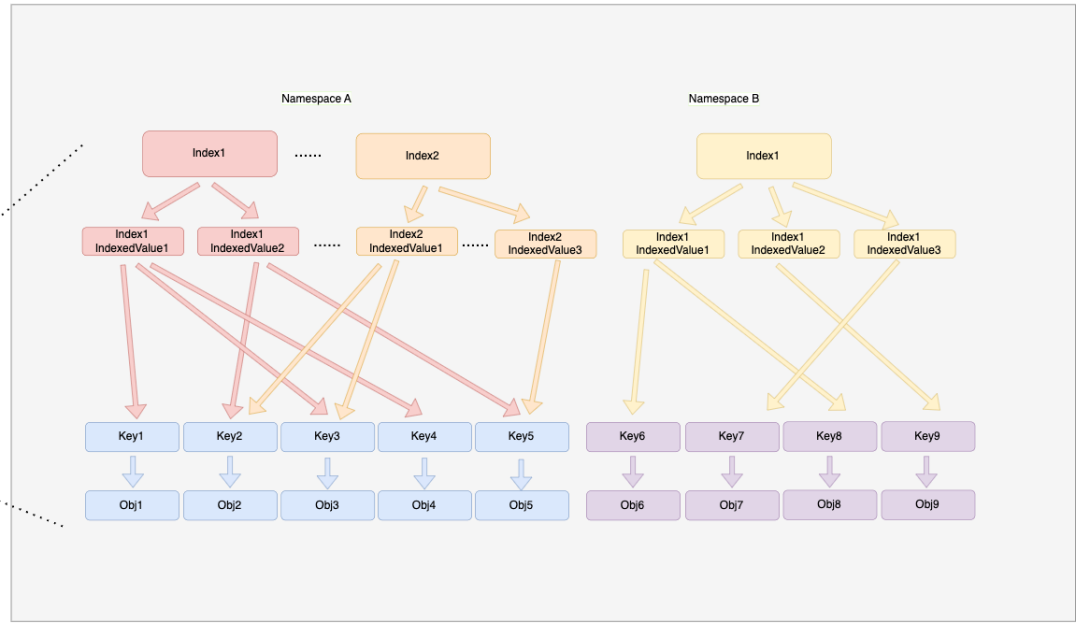
- Data structure improvement
- Lock contention reduction
- Computation reduction/Data reduction
- Asynchronous computation
- Bandwidth reduction
- Proper caching
- Other support functions.



What we've done

- Improve data structured for watchcache
- Asynchronous handle time-consuming operation GetAttrs
- Reduce watch with resource version = 0 data computation.
- Improve the data pushing mode of cacheWatchers



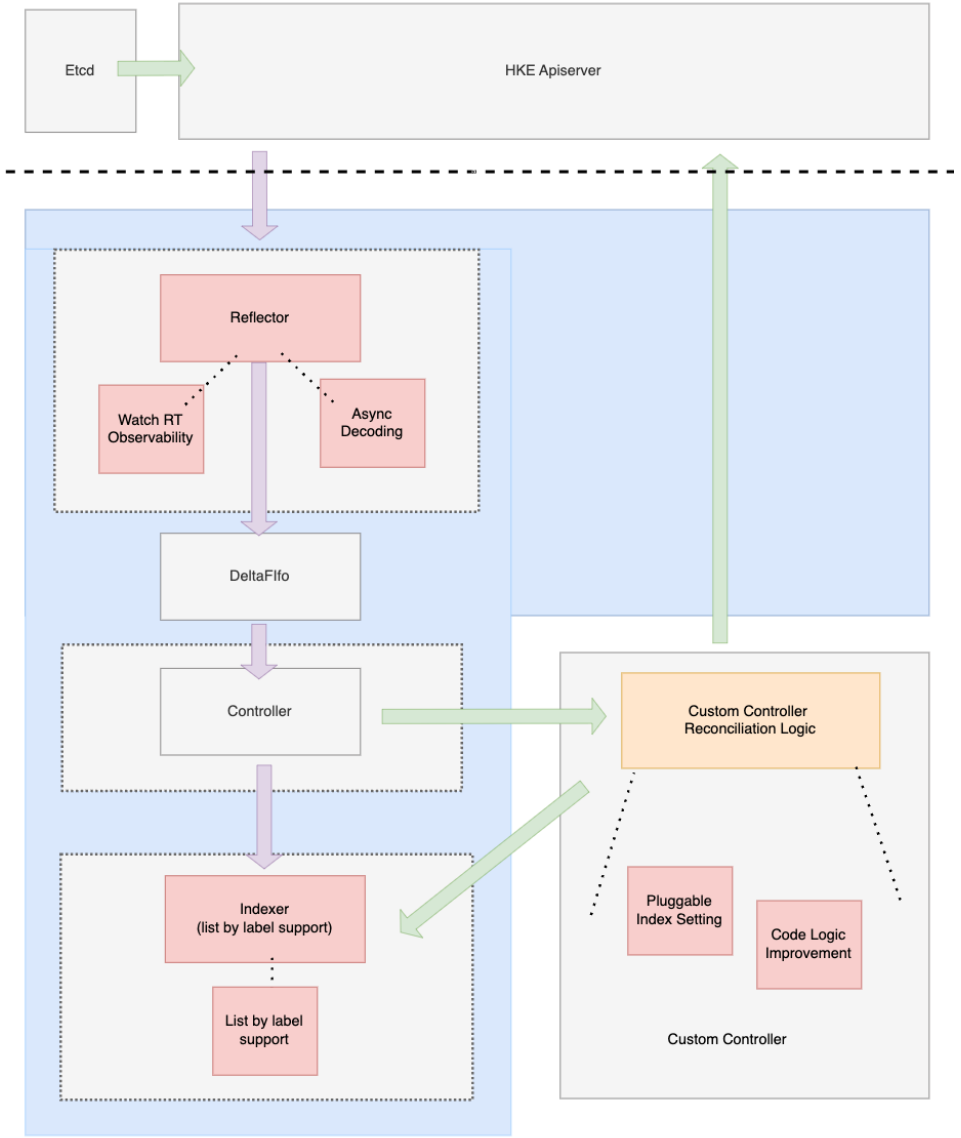


What we've done

Customized controller-runtime for watch

- Main purpose for watch is to support list by LabelSelector
- Also, introduced an asynchronous parallel decoding scheme

Based on this, many controllers are optimized.
kube-controller-manager is also optimized in similar way.

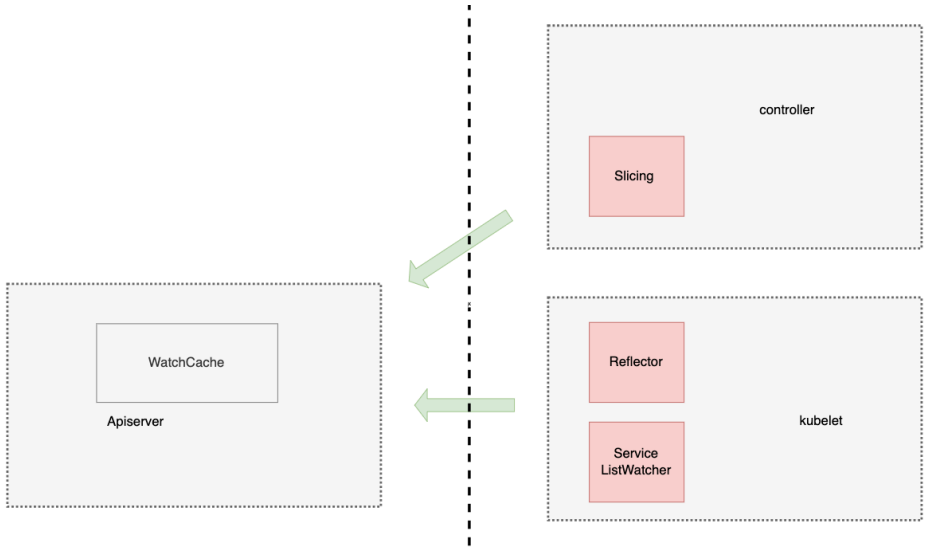


What we've done

Reduction of Apiserver traffic

- Slicing
  - Endpoint Slice
  - CR Slice
- Kubelet Service: only a few services are needed by kubelet for service discovery purposes.
- Watch after watch

  Exxxx 21:07:09.861118 80927 framer.go:101] io short buffer, frameLength is 4285, but read zero bytes, header is , length is 61444, error is net/http: request canceled (Client.Timeout or context cancellation while reading body）
  an error on the server ("unable to decode an event from the watch stream: got short buffer with n=0, base=4092, cap=163840")



What we've done

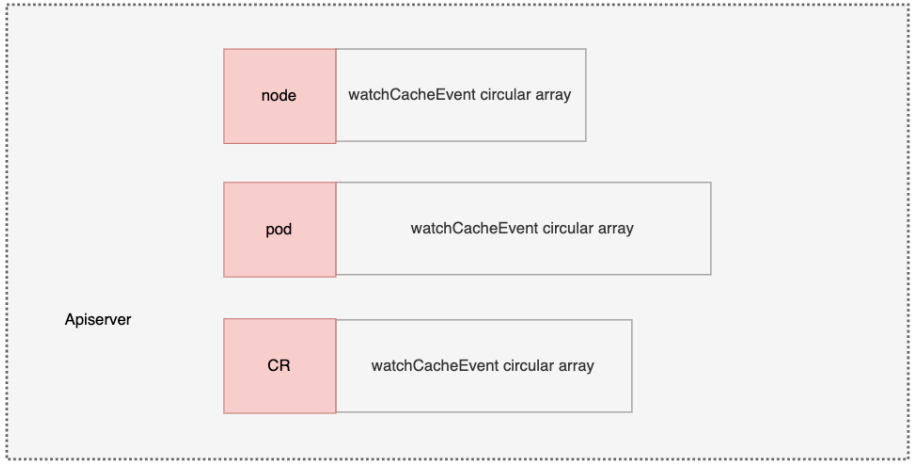
Watch cache size adjustment
Static adjustment
Dynamic adjustment
For pods, nodes, configmaps, important Custom Resources for clients.

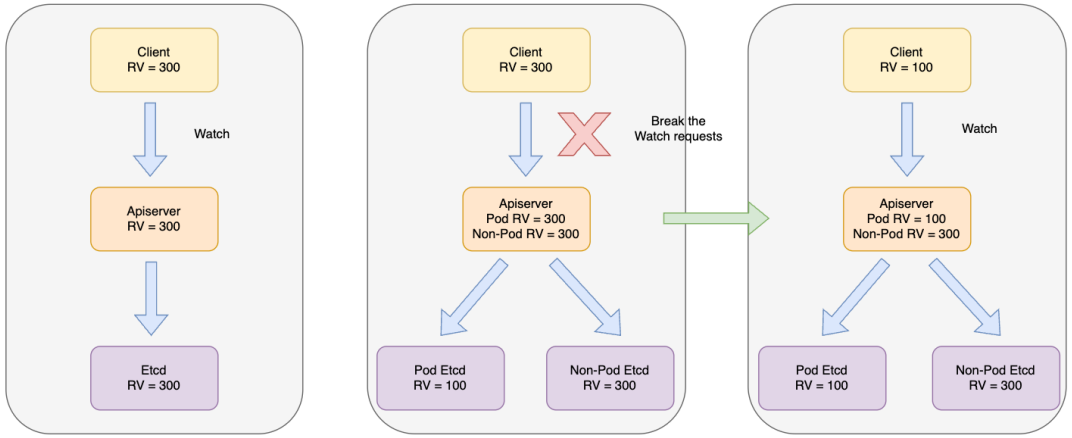
Two watch metrics:
Forcing-close
Init-events-error

Encoding scheme

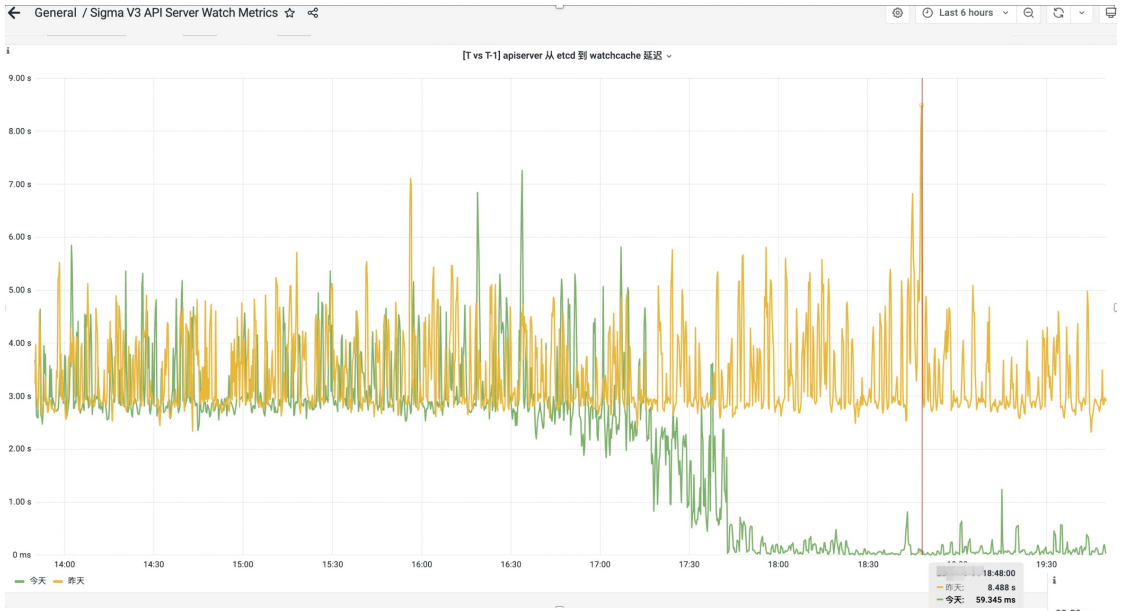
Etcd progress notify to reduce 504

Waiting Watch for etcd split - By returning too large rv for watch requests and let clients to automatically issue an List request after the etcd split operation.



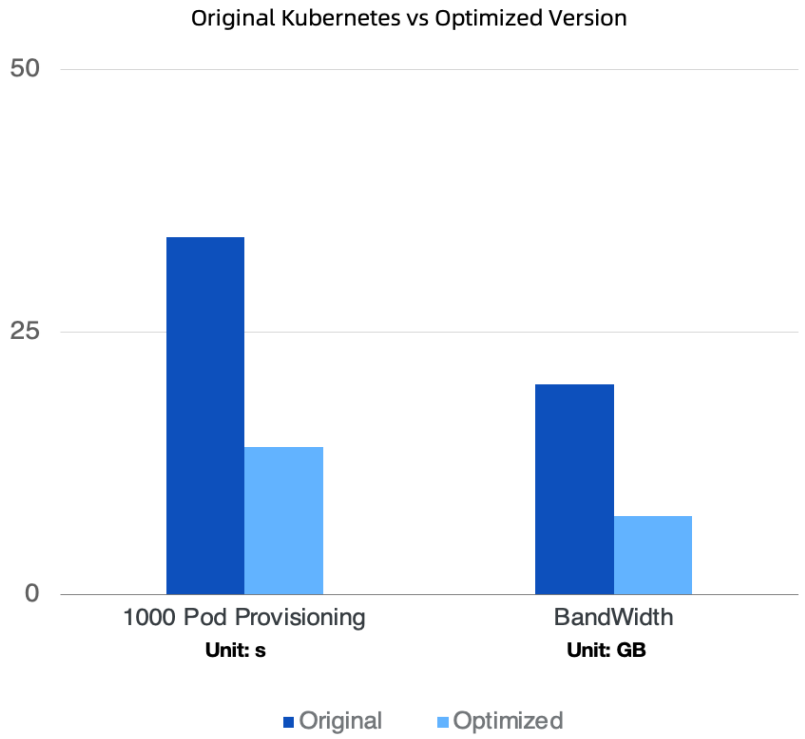


Benefits



- Time from etcd to apiserver watchcache improves from 3s to 100ms level
- Total time from watch event enters apiserver to it leaves apiserver improves from 5s to 500ms level

Benefits



1000 Pod Provision time P95 34.43s -> 14.05s，a 59.2% enchancement

And the apiserver network bandwidth decreases from 20GB to an average of 7.5 GB

Pod
x 100000

WatchEvent
x 1000000000

Watch Mechanism

Watch SLO

Optimization

Summary

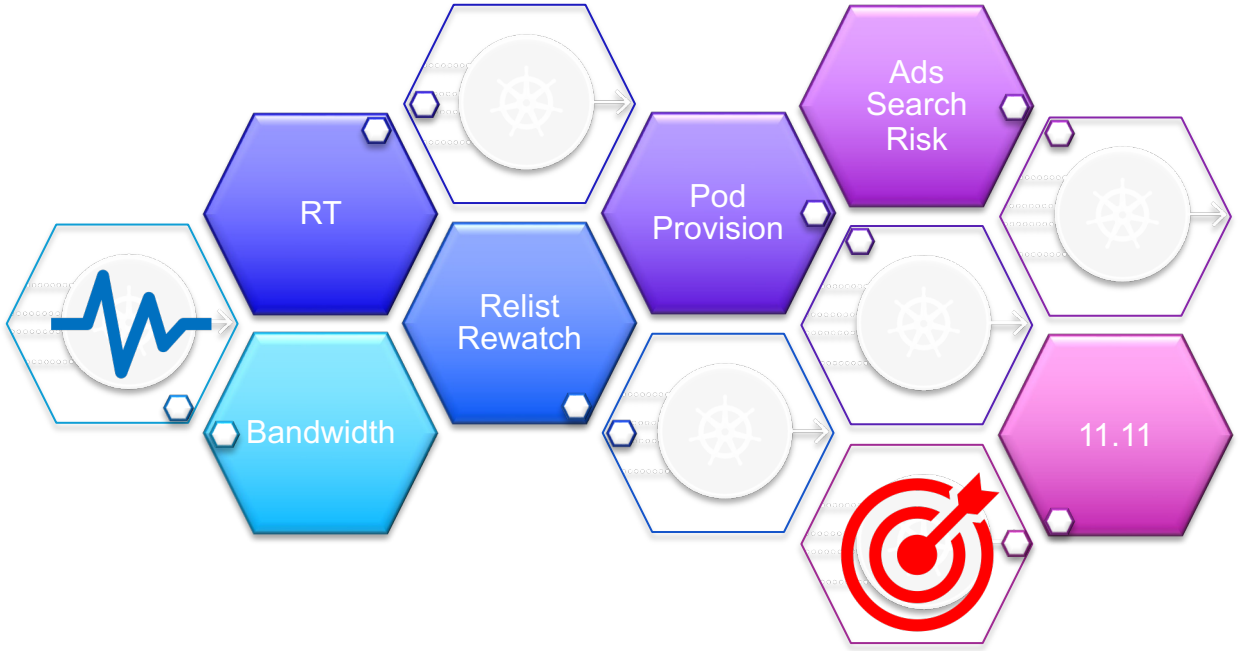
Summary

2x to 10-90x times reduction in terms of the following metrics:

- P99/P999 RT from etcd to apiserver watchcache
- P99/P999 RT from apiserver to client side
- Client side relist/rewatch number reduction and the delay caused by the full relist
- The end-to-end pod provisioning time, from pod creation to running.
- Apiserver load balancer bandwidth reduction
- The scalability/stability of the whole system

Positively Affects **billions** of watch events per day.

Positive feedback from internal users (**Alipay ads/search, risk management**, hpa, etc) and supports **Alibaba/Alipay 11.11** for recent years



You will get what you want very soon

Summary and Road Ahead

Are we done yet?

No

There are still problems going on

- The reduction of client side CPU
- The reduction of lock contention for arbitrary client code
- The proper adjustment of apiserver watch cache
- The proper usage of client – for removing 409
- The etcd watch guarantee
- Traffic measurement and analysis
- The inclusion of higher version K8s feature


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Thank you
You will get what you want
very soon

Reference

https://github.com/kubernetes/community/blob/master/sig-scalability/slos/slos.md

Page 5
ImageSrc: https://packetpushers.net/podcasts/kubernetes-unpacked/ku031-is-kubernetes-getting-boring/
ImageSrc: https://www.sohu.com/a/794174392_120534958
ImageSrc: https://baijiahao.baidu.com/s?id=1748205193664351344&wfr=spider&for=pc

Page6-Page7
ImageSrc: https://giphy.com/gifs/dots-circles-arrows-Q99A6dXgEr3shEYNi3
ImageSrc: https://m.thepaper.cn/newsDetail_forward_27922170
ImageSrc: https://www.cnblogs.com/hongmaju/p/4408739.html
ImageSrc: https://www.godic.net/webting/Play?id=07487f16-c48f-4870-8013-d36a2ed780d2

Page12-Page13
ImageSrc: https://www.vecteezy.com/photo/3670909-missing-jigsaw-puzzle-piece-with-lighting-business-concept-for-completing-the-finishing-puzzle-piece
ImageSrc: https://www.vectorvest.com/blog/featured-news/the-near-perfect-indicator/