How Multi-Cloud Multi-Cluster HPA Helps Trip.com Group Deal with Business Downturn and Rapid Recovery

Honghui Yue&Jingxue Li

Agenda

- Background
- Design
  - Reliability First
  - Capacity Scheduling
- Implementation
- Lessons learned

About Us

Trip.com Group

Trip.com Group is a leading global travel service provider comprising of Trip.com, Ctrip, Skyscanner, and Qunar with the mission" to pursue the perfect trip for a better world".

Across its platforms, Trip.com Group helps travellers around the world make informed and cost-effective bookings for travel products and services. Trip.com Group technology enables partners to connect their offerings with users through the aggregation of comprehensive travel-related content and resources, and an advanced transaction platform consisting of apps, websites and 24/7 customer service centers.

Ctrip QUNAR Trip.com Skyscanner

Travix travelfusion make my trip Dil toh roaming hai Tripadvisor

Project Core Contributors

Honghui Yue

Container&Hybrid Cloud

Cloud Native R&D Director & Architect

Jingxue Li

Container&Hybrid Cloud

Cloud Native R&D Expert

Zongqing Li

Container&Hybrid Cloud

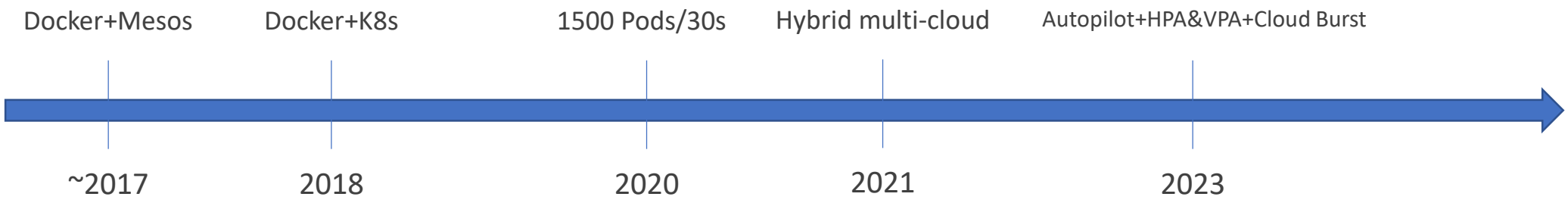
Senior Cloud Native R&D Engineer

Yao Wang

Container&Hybrid Cloud

Cloud Native R&D Engineer

The Cloud Native Trip



Kubernetes Native

~30 K8s Clusters

~500K Pods

~20 Workloads(App、Redis、Kafka...)

Hybrid Multicloud

~5 Cloud Providers

~3 Regions(Asia, EU)

~60 Cloud Native Services

Elastic Scheduling

1500 Pods/30s

90%+ Apps enable Autopilot+HPA&VPA

App Node avg CPU 25%~35%

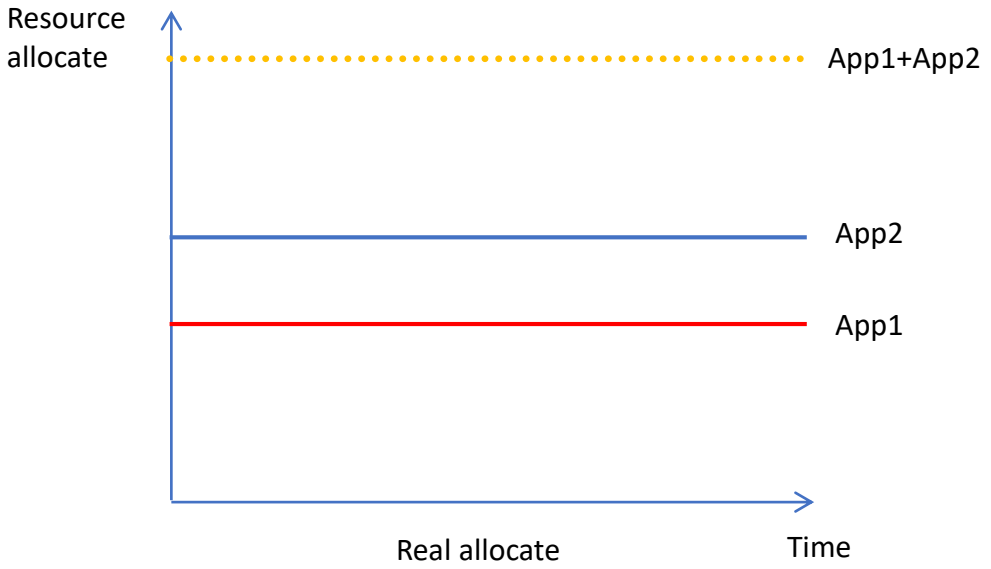
The Cloud Native Trip

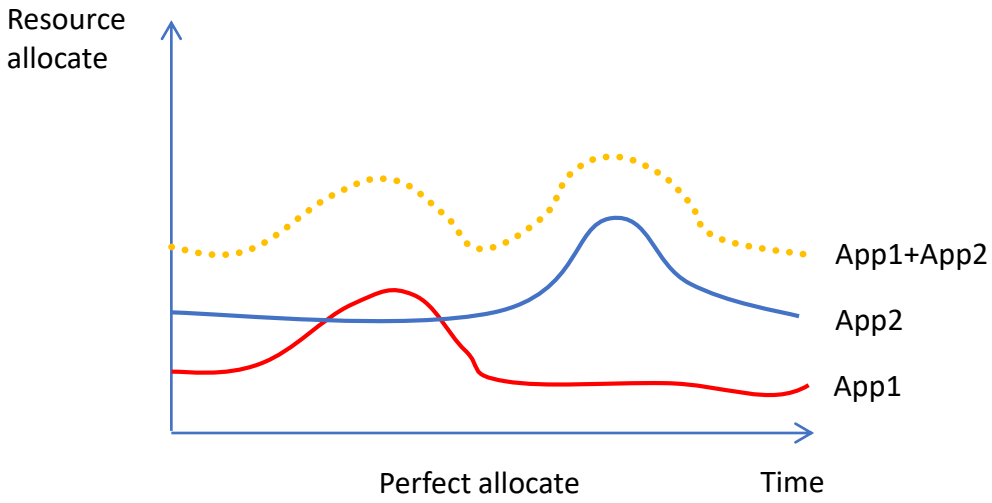
- Decreased the capacity peak of AZ by more than **28%** during the last Labor Day
- We successfully dealt with multiple holiday traffic peaks after the pandemic, with the highest expansion peak of **~2000 Pod/min**
- More than **30%** of load was offloaded to the public cloud at peak times

What problems can HPA solve?

- It’s hard to maintain the capacity of Apps manually
  - Why
    - Too many deployments: **10k+** apps and **20k+** deployments
    - Hard to predict traffic for individual Apps: Covid-19 and policy changing
    - Hard to balance between availability and cost: high pressure on cost saving during pandemic, which means less resource redundancy.
  - Bad result
    - **Low efficiency**: Too much time is wasted on maintaing capacity of Apps.
    - **Capacity risk**: Disordered and paniced scales of Apps during the site peak put an overwhelming strain on resources.
    - **Bad Experience**: Developers&SREs are unhappy.

What problems can HPA solve?





- Too much resources are wasted
  - Why
    - Scale out beyond the actual demand.
    - Scale out in advance, without resource time-sharing effect between Apps.
  - Bad result
    - Low resource utilization(avg CPU < 10%)
    - Need to prepare much more resources than the actual demand for the site capacity plan.

Agenda

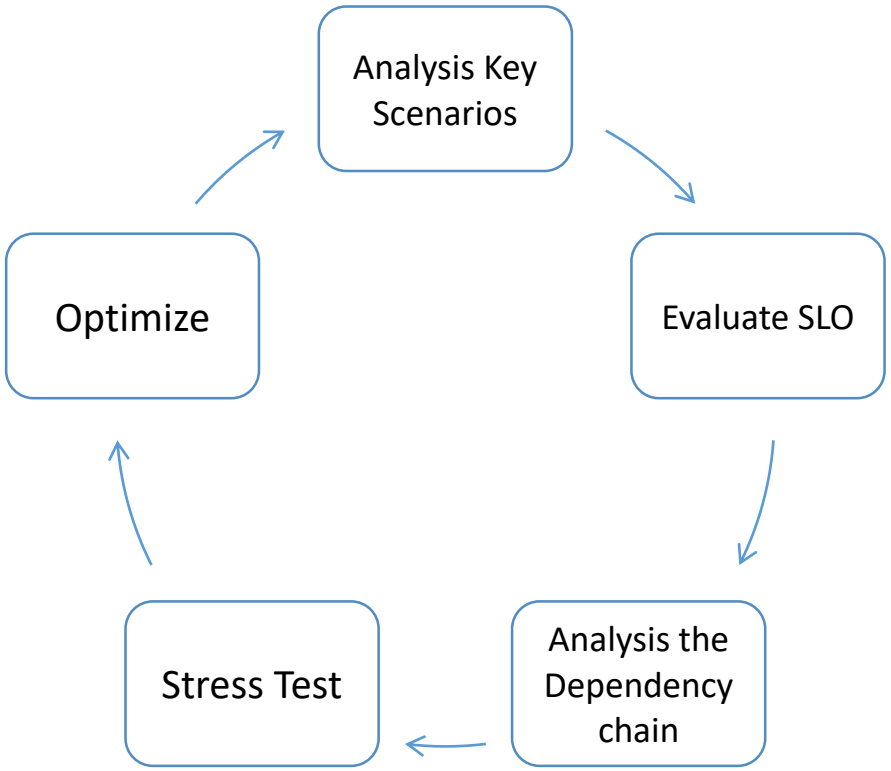
- Background
- Design
  - Reliability First
  - Capacity Scheduling
- Implementation
- Lessons learned

Confidence&Trust

HPA looks good, but what if it fails to scale at critical moments?

High performance + high reliability + enough capacity!

E2E Scale Performance SLO



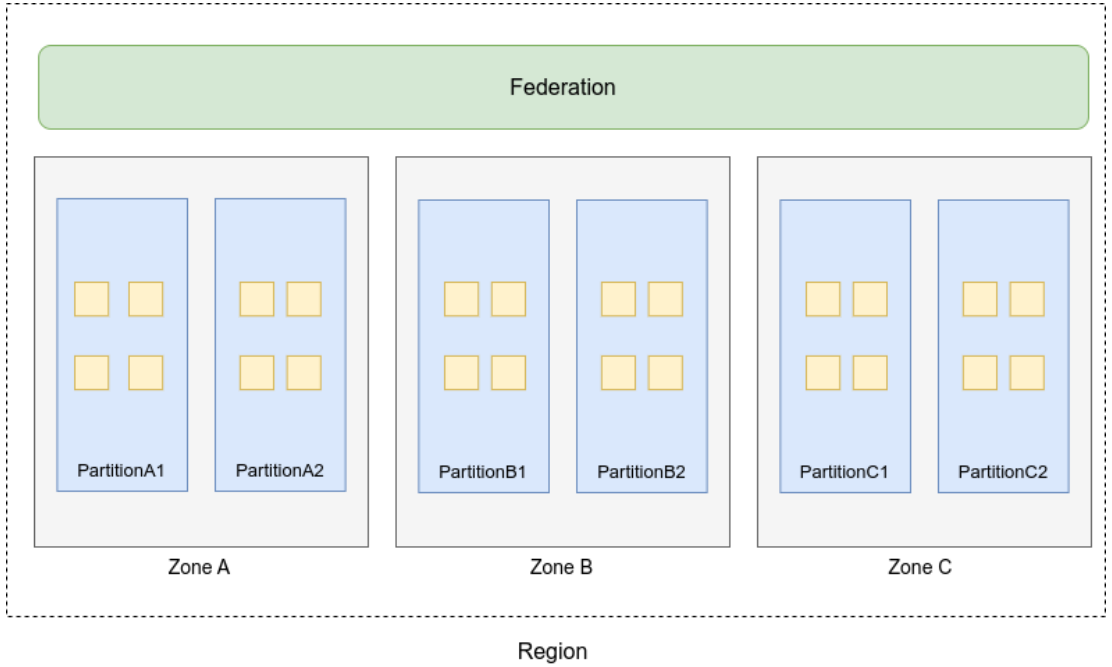
- Evaluate SLO based on scenarios
  - Daily Expansion
  - Holidays
  - Disaster Recovery
  - Future growth
- Stress testing
  - Construct testcases based on product data
  - Test e2e in real environment
  - Collect key metrics(throughput, latency, success rate)
- Optimization
  - Refactor the architecture and minimize cross-cluster dependency in the east-west direction.
  - Optimize the code

Fault isolation architecture

How can we ensure the reliability of the overall system, as partial failure is unavoidable?

Fault isolation+Redundancy+Failover！

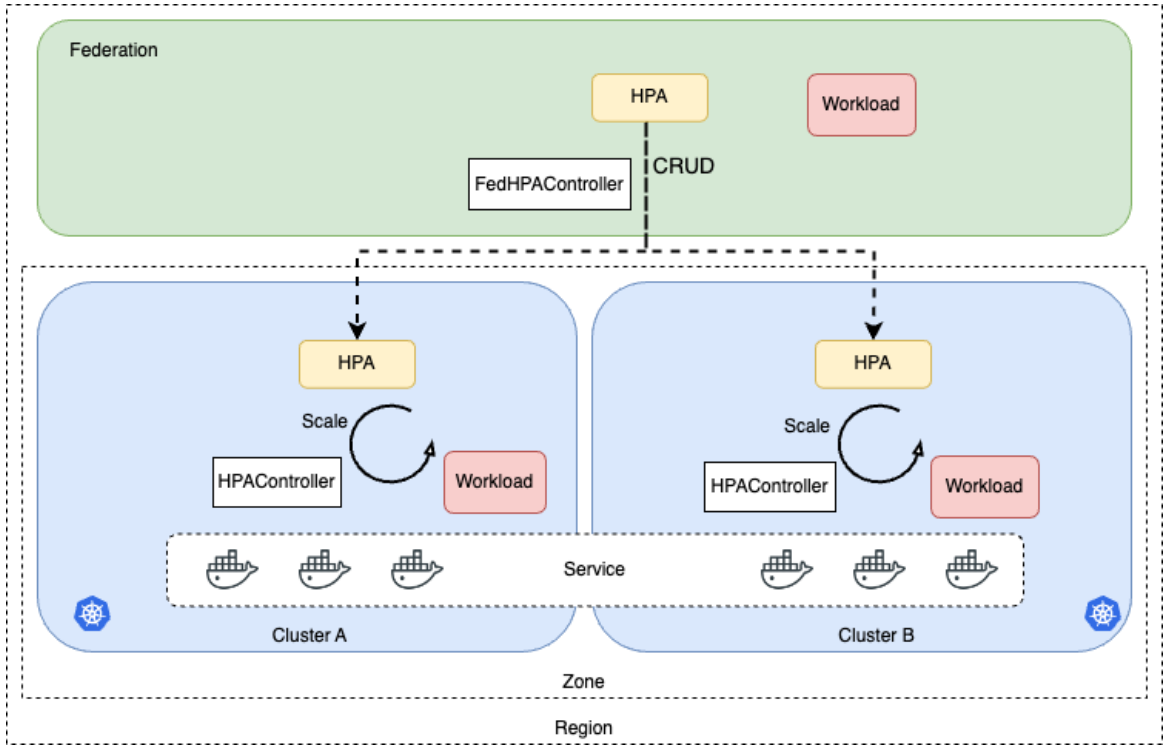
Fault isolation architecture



- Architecture
  - Isolation domain: Region/AZ/Partition
  - At least 1 K8s cluster in each partition, and 2 partitions in each AZ, at least 3 AZs in each region
  - The regional Federation is responsible for
    - Hiding the K8s member clusters to the end users
    - Managing AZ/Partition/Cluster within the region
    - Scheduling resource between Clusters within the AZ

- Key constraints
  - The critical processing chain should be done within the fault domain
  - Minimize dependency between isolation domains in the east-west direction
  - App pods should be distributed across multiple partitions within multiple AZs
  - Changes should not be released on different AZs or even different partitions at the same time

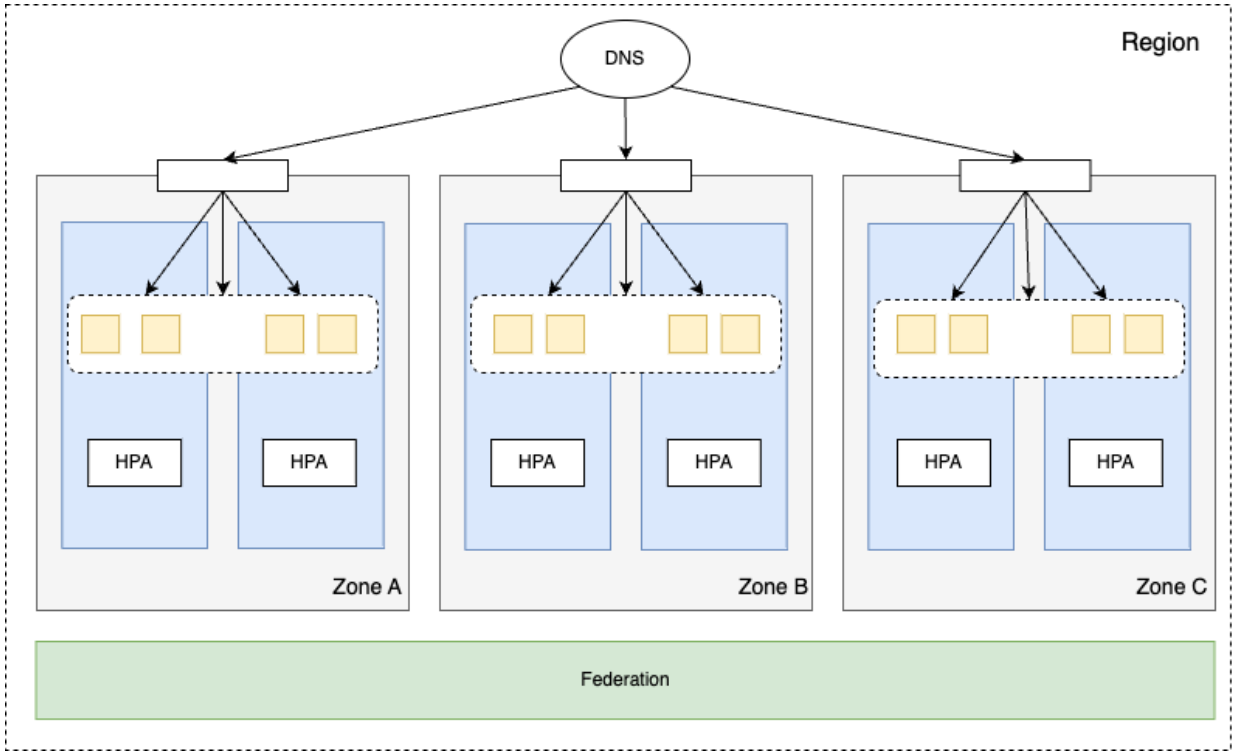
Fault isolation architecture



Isolation domains for multi-cluster HPA

function	critical	frequency	isolation domain
workload scale	yes	high	Cluster
HPA CR CRUD	no	low	Region
cluster rebalance	no	low	Region

Capacity scheduling on Hybrid-cloud



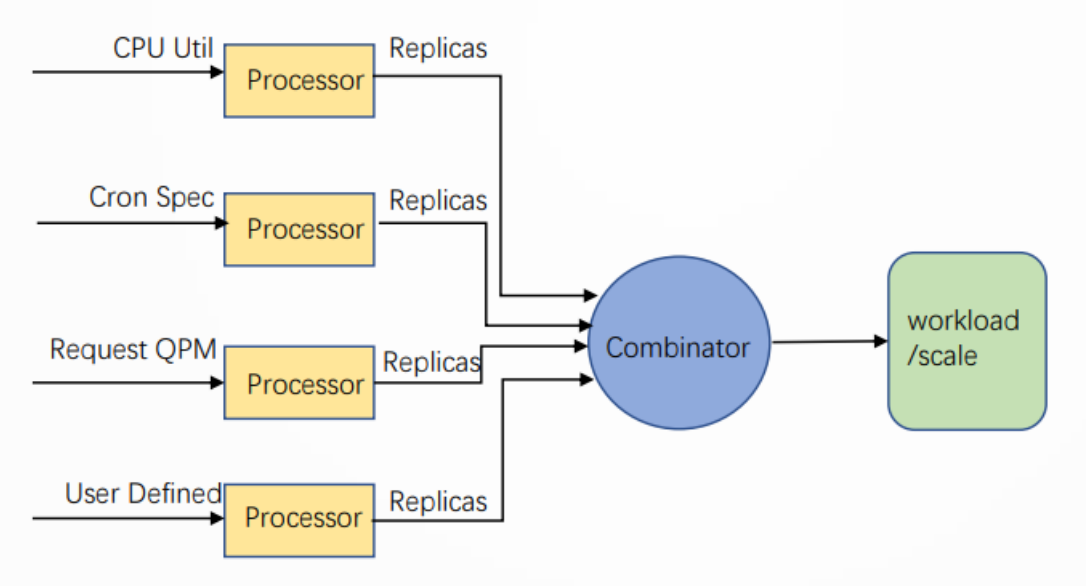
- Network&Traffic
  - Routable pod IP across clusters
  - App pods within the same AZ share the same traffic scheduling unit(e.g.: svc, ingress)
  - Locality load balancing
- Capacity scheduling
  - Schedule load between AZs by traffic load balancing
  - Rebalance between clusters within each AZ by Federation scheduling

Zone A: Public Cloud
Zone B&C: Private Cloud

Agenda

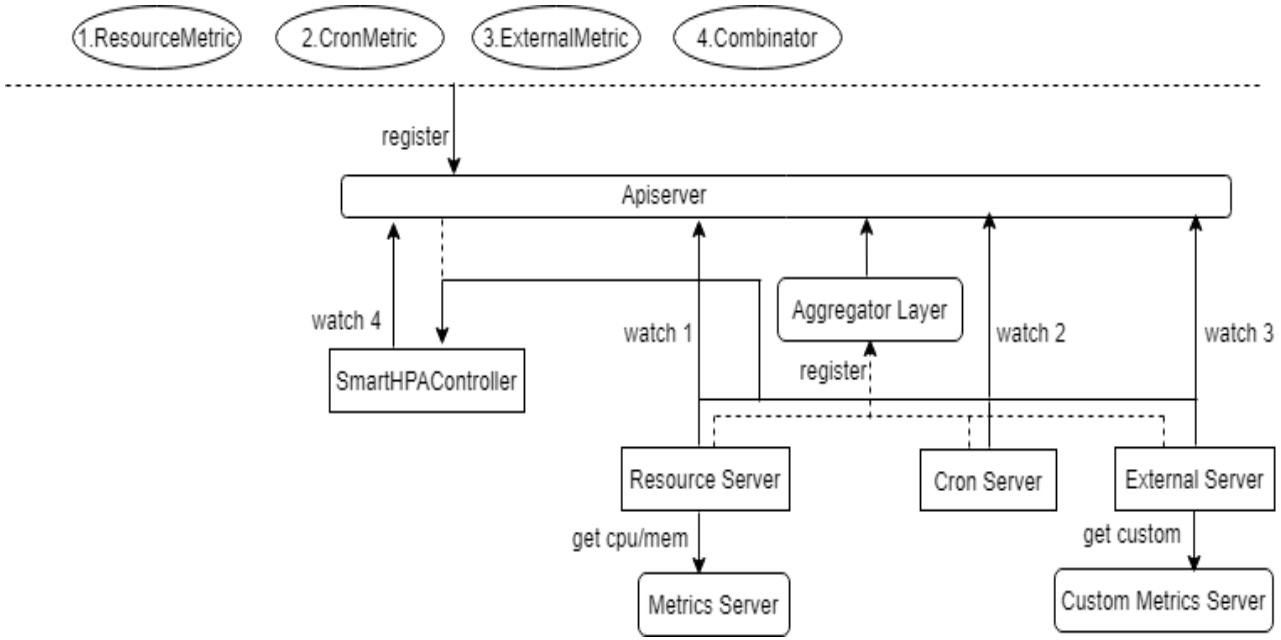
- Background
- Design
  - Reliability First
  - Capacity Scheduling
- Implementation
- Lessons learned

Design of HPA



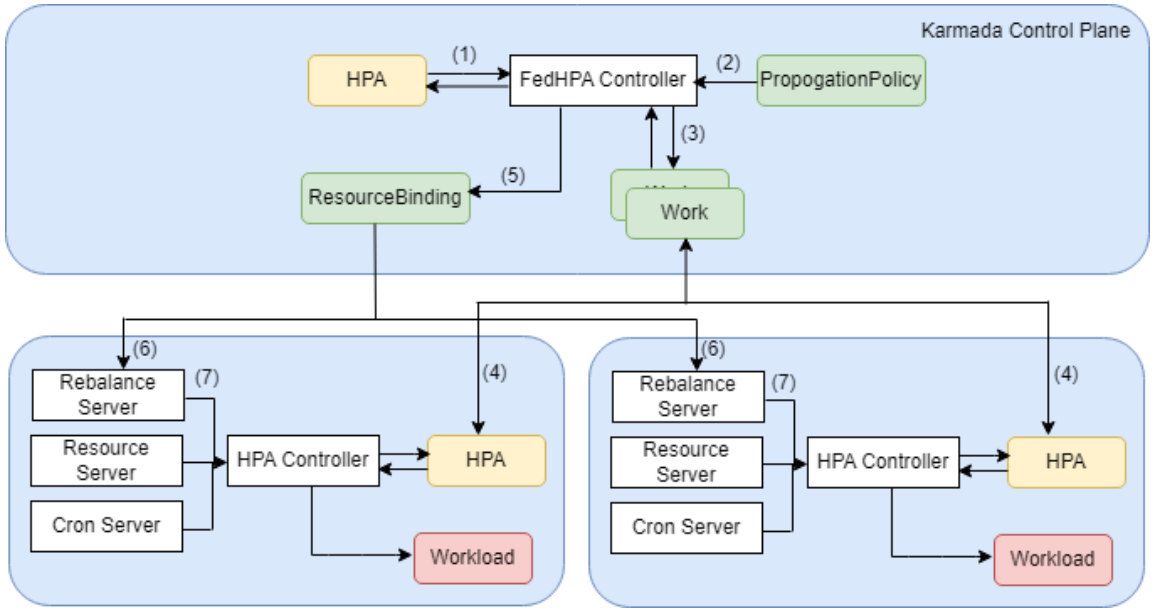
- Components
  - Processor
    - output replicas recommendation
  - Combinator
    - calculate the final replicas leverage the input of replicas recommendations from Processor
    - Scale the target workload

Architecture of HPA



- Components
  - Processor
    - Resource Server
    - Cron Server
    - External Server
  - Combinator
    - SmartHPAController(which is the name of HPA product in Trip.com, can be simply called HPAController)

Multi-cluster HPA based on Karmada

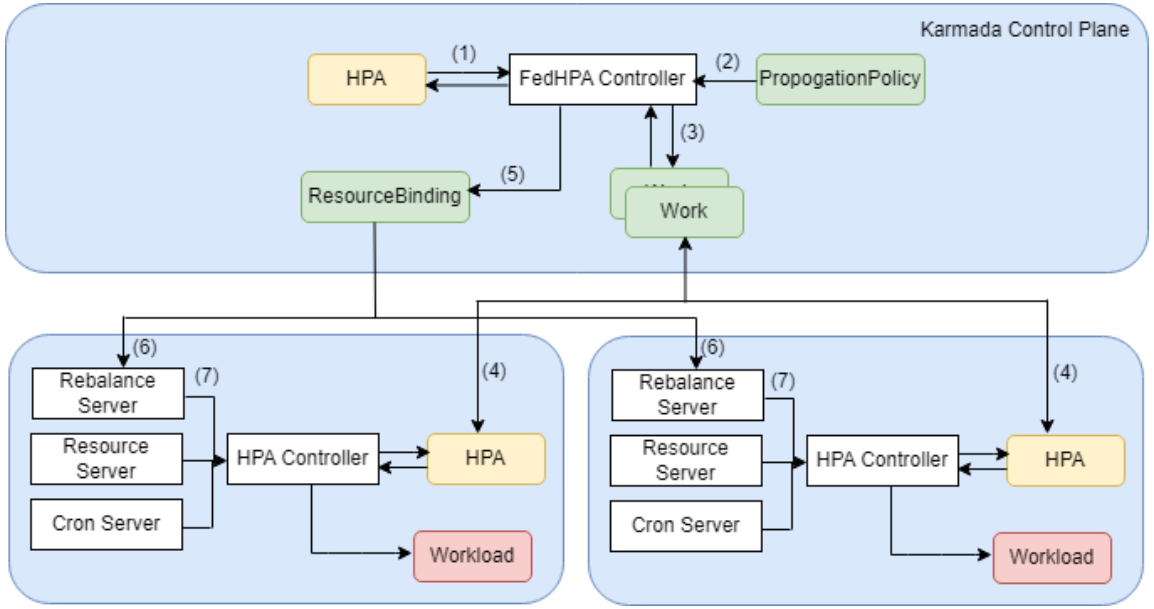


- Solution
  - The same HPA definition is shared in the federation and member cluster
  - HPA in member clusters can work independently and will not be impacted due to the failure of other clusters or federation
  - FedHPAController distributes HPA objects and rebalances between clusters according to the PropagationPolicy

```
  replicaScheduling:
    replicaDivisionPreference: Weighted
    replicaSchedulingType: Divided
    weightPreference:
      staticWeightList:
      - targetCluster:
          clusterNames:
          - shaxy-a
        weight: 3
      - targetCluster:
          clusterNames:
          - shaxy-b
        weight: 7
priority: 0
resourceSelectors:
- apiVersion: captain.cloud.ctrip.com/v1
  kind: Rollout
  name: r100012822-91007693
  namespace: pro-captain
schedulerName: default-scheduler
```

design proposal： https://github.com/karmada-io/karmada/blob/master/docs/proposals/hpa/federated-hpa.md

Distribute HPA Resource According to Propagation



(1) FedHPAController watches HPA object

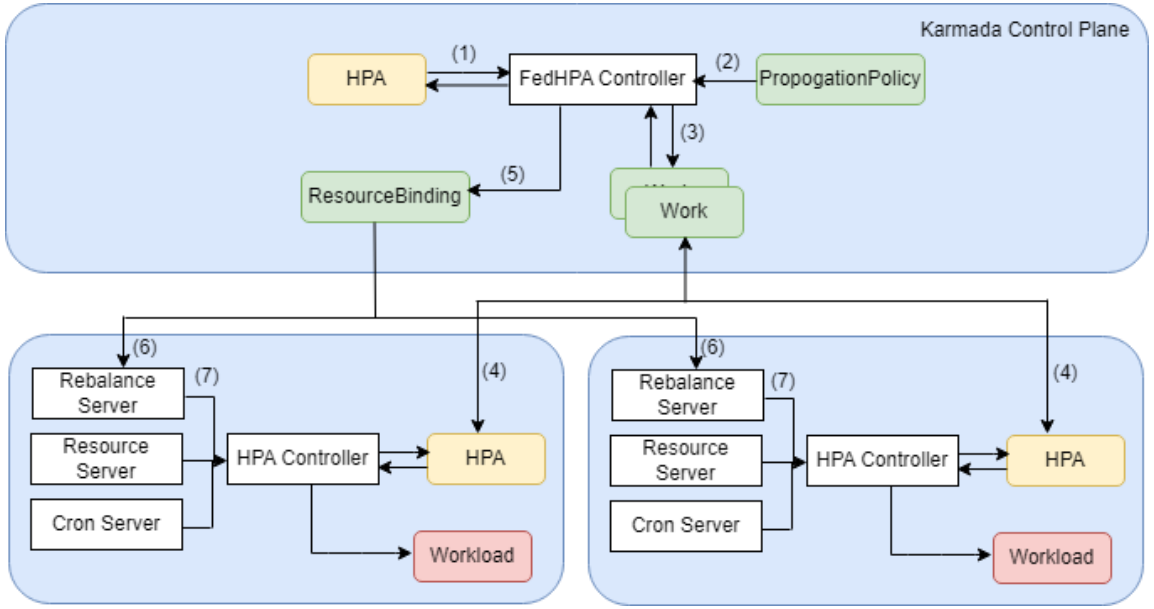
(2) FedHPAController watches PropagationPolicy (weight)

(3) FedHPAController creates Works according to the weight and HPA

- work created for each HPA in each cluster
- execute algorithm DistributeReplicasByWeight(replicas, clusterWeight)

(4) Karmada distributes Works to member clusters

Rebalance between clusters gracefully



- How do the weights in Propagation affect the redistribution of minReplicas/maxReplicas among clusters?
  - FedHPAController calculates desired distribution according to the weight → (2)
- Reconcile multiple rounds to approach the desired weight
  - FedHPAController calculates the current round of applied distribution based on the actual ready pod and desired and updates the Works →(3)
  - FedHPAController distributes HPA to member clusters according to Works → (4)
  - FedHPAController calculates the current round of applied distribution based on the actual ready pod and desired and updates the Works →(3)
  - ……

Rebalance between clusters gracefully

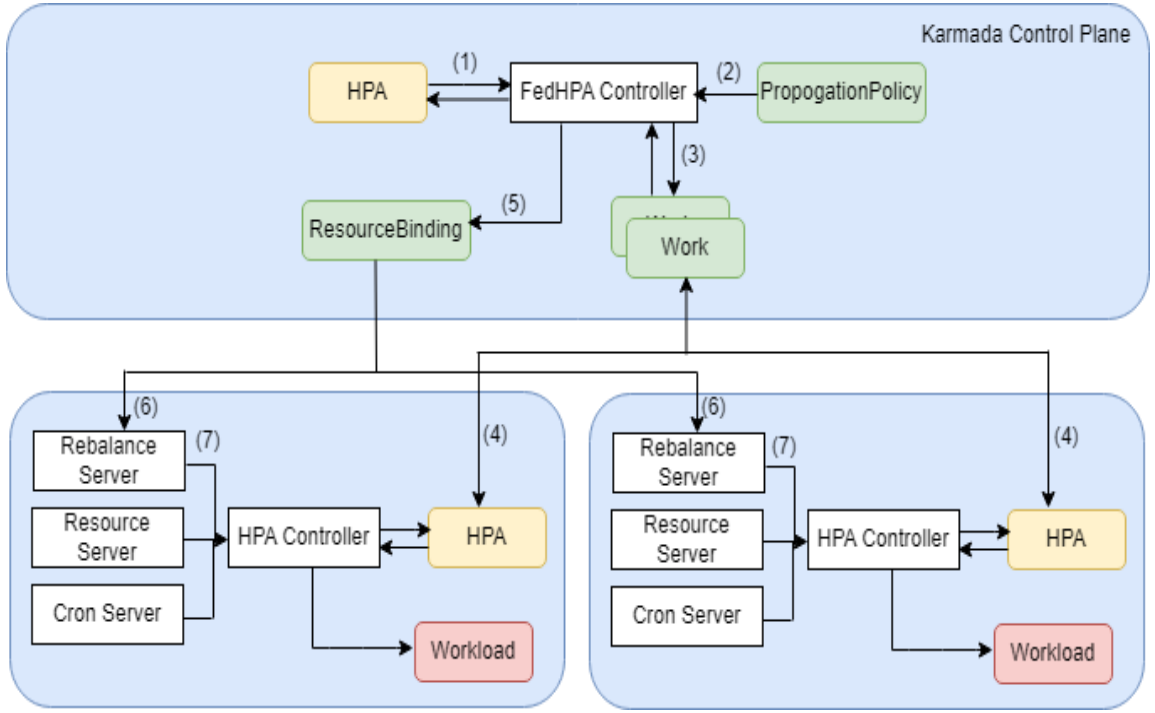
	before		=>	after	
	A	B		A	B
weight	1	0		0	1
desired max replicas	20	0		0	20
desired min replicas	10	0		0	10
actual replicas	10	0		0	10
ready replicas	10	0		0	0
note	-	-		10 in terminating	10 in creating

How to rebalance minReplicas/maxReplicas among member clusters?

- Calculate desired distribution for min/max replicas
- Reconcile multiple rounds to approach the desired distribution

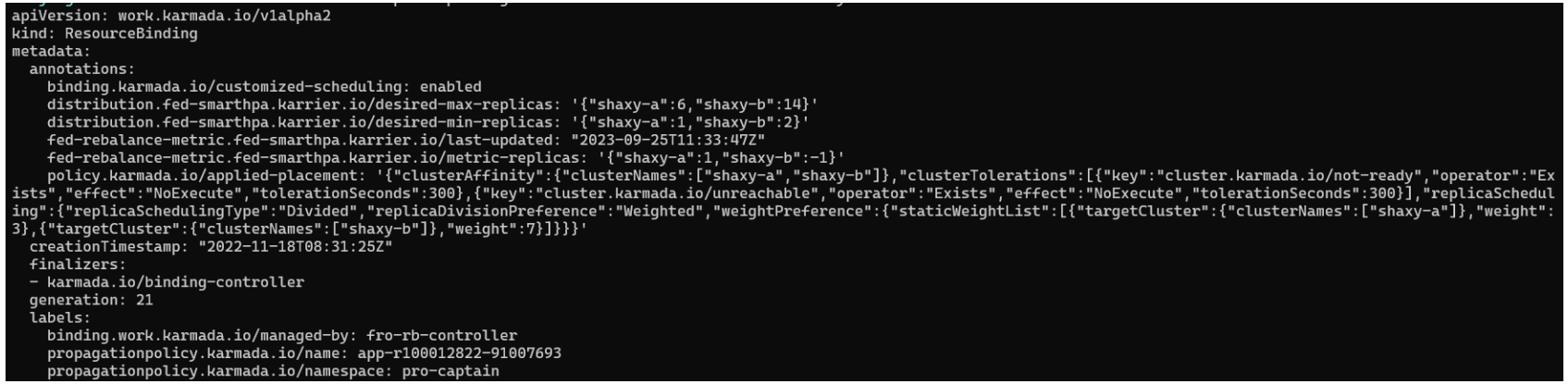
	Group X		
cluster	A	B	C
total minReplicas (0th remaining)	12		
weight	1	2	3
ready replicas (0th offset)	3	6	1
(1st) desired minReplicas	2	4	6
1st distribution	2	4	1
1st remaining	5		
1st offset (buffer / weight)	1	2	-
2nd desired	2	3	-
2nd distribution	1	2	-
2nd remaining	2		
2nd offset (buffer / weight)	-	-	5
3rd distribution	-	-	2
applied minReplicas	3	6	3

Rebalance between clusters gracefully



- When the weight in the Propagation changes, how are the actual replicas redistributed among the clusters?
  - FedHPAController calculates the desired distribution based on actual replicas and weight → (2)
- Reconcile multiple rounds to approach the desired distribution
  - FedHPAController calculates the applied distribution based on actual and desired, recorded in the ResourceBinding annotation → (5)
  - RebanlaceServer within member cluster watches Resourcebinding, converts it into a rebalance signal → (6)
  - HPAController scales workload based on rebalance and other signals → (7)
  - FedHPAController calculates applied distribution and then saves it using annotation of ResourceBinding according to actual and desired values → (5)
  - ……

Rebalance between clusters gracefully



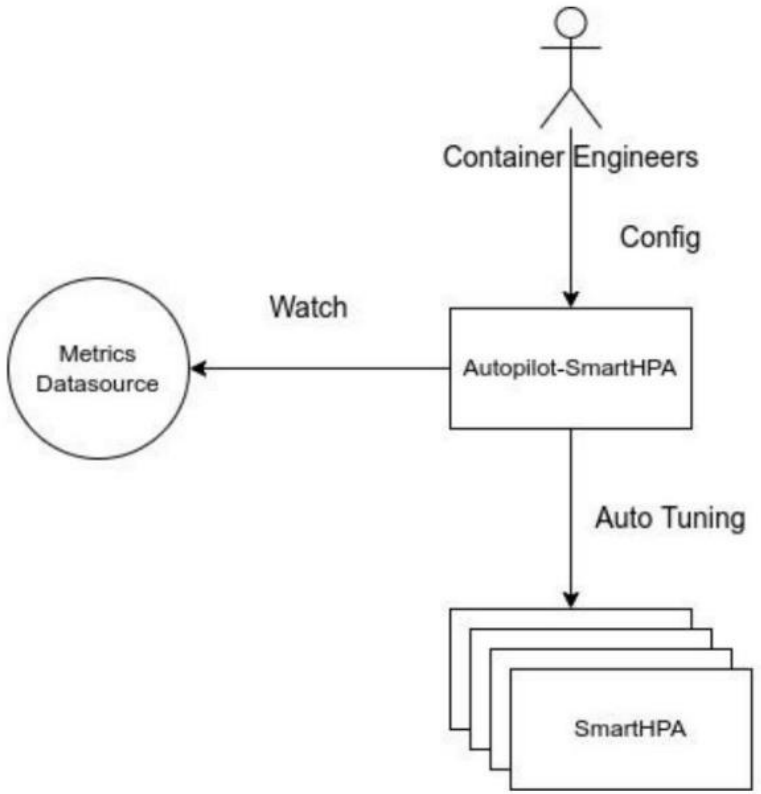
```
apiVersion: work.karmada.io/v1alpha2
kind: ResourceBinding
metadata:
  annotations:
    binding.karmada.io/customized-scheduling: enabled
    distribution.fed-smarthpa.karrier.io/desired-max-replicas: '{"shaxy-a":6,"shaxy-b":14}'
    distribution.fed-smarthpa.karrier.io/desired-min-replicas: '{"shaxy-a":1,"shaxy-b":2}'
    fed-rebalance-metric.fed-smarthpa.karrier.io/last-updated: "2023-09-25T11:33:47Z"
    fed-rebalance-metric.fed-smarthpa.karrier.io/metric-replicas: '{"shaxy-a":1,"shaxy-b":-1}'
    policy.karmada.io/applied-placement: '{"clusterAffinity":{"clusterNames":["shaxy-a","shaxy-b"]},"clusterTolerations":[{"key":"cluster.karmada.io/not-ready","operator":"Ex
ists","effect":"NoExecute","tolerationSeconds":300},{"key":"cluster.karmada.io/unreachable","operator":"Exists","effect":"NoExecute","tolerationSeconds":300}],"replicaScheduli
ing":{"replicaSchedulingType":"Divided","replicaDivisionPreference":"Weighted","weightPreference":{"staticWeightList":[{"targetCluster":{"clusterNames":["shaxy-a"]},"weight":
3},{"targetCluster":{"clusterNames":["shaxy-b"]},"weight":7}]}}}'
  creationTimestamp: "2022-11-18T08:31:25Z"
  finalizers:
  - karmada.io/binding-controller
  generation: 21
  labels:
    binding.work.karmada.io/managed-by: fro-rb-controller
    propagationpolicy.karmada.io/name: app-r100012822-91007693
    propagationpolicy.karmada.io/namespace: pro-captain
```

How to balance the load between member clusters?

- calculate desired distribution according to actual replicas
- Reconcile multiple rounds to approach the desired distribution. Scale out the application on the side with fewer replicas than desired distribution first in each step.

cluster	A	B
weight	1	1
current replicas	9	1
desired replicas	5	5
rebalance signal	-1	3
current replicas	9	3
current replicas	7	3
rebalance signal	-1	4
current replicas	7	4
current replicas	6	4

How to manage HPA parameters?



- Fully managed parameters
  - min replicas
  - max replicas
  - target
  - ...
- Safety
  - support different profiles for different Apps
  - reconcile step by step and avoid big changes in a short time duration

Agenda

- Background
- Design
  - Reliability First
  - Capacity Scheduling
- Implementation
- Lessons learned

Lessons learned

- Performance and reliability are the cornerstones of elastic systems and deserve continued investment
- Establish data insight capabilities early
- Find influential seed users who share the same beliefs and solve difficult problems together, and other users will naturally follow
- The current downturn is temporary and we need to prepare in advance for future recovery


KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023

Thanks

Q&A