R7 Test Document

Introduction

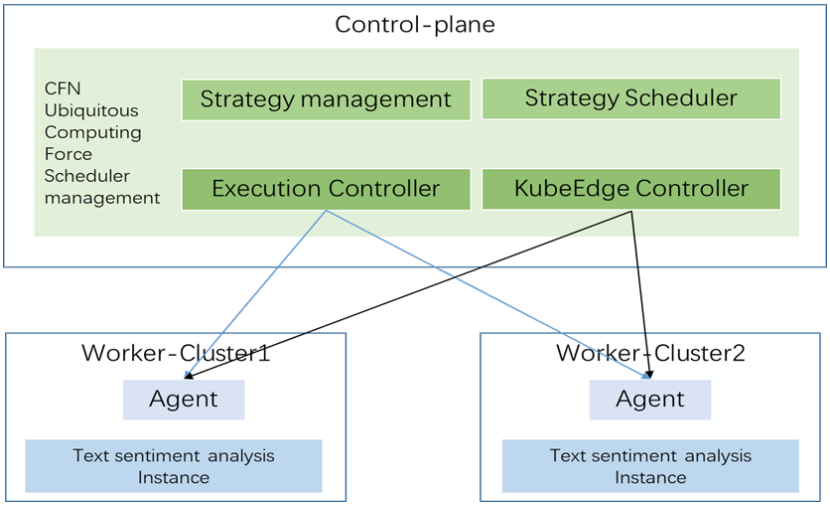
The purpose of this test is to demonstrate two scheduling use cases of text sentiment analysis service:

Case 1. Scheduling computing force by cluster weight;

Case 2. Rescheduling computing force when a cluster resource is abnormal.

Akraino Test Group Information

Test Architecture



Test Framework

Hardware:

Control-panel: 192.168.30.12192.168.30.21

Worker-Cluster1 192.168.30.5 192.168.30.22192.168.30.20

Worker-Cluster2 192.168.30.2192.168.30.16192.168.30.25

Software:

Karmada: V1.4.0, Open, Multi-Cloud, Multi-Cluster Kubernetes Orchestration

Kubernetes: an open-source system for automating deployment, scaling, and management of containerized applications.

sentiment: an text emotion analysis service

Test description

Case 1. Scheduling by weight

1Create a deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sentiment
  labels:
    app: sentiment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sentiment
  template:
    metadata:
      labels:
        app: sentiment
    spec:
      imagePullSecrets:
      - name: harborsecret
      containers:
      - name: sentiment
        image: 192.168.30.20:5000/migu/sentiment:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9600
          protocol: TCP
          name: http
        resources:
          limits:
            cpu: 2
            memory: 4G
          requests:
            cpu: 2
            memory: 4G
```

2Create nginx deployment yaml file.

Create a deployment and name it sentiment. Execute commands as follow:

```
kubectl --kubeconfig /etc/karmada/karmada-apiserver.config create -f deployment.yaml
```

3. Create a distribution yaml file, PropagationPolicy.yaml

```
apiVersion: policy.karmada.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: sentiment-propagation
spec:
  resourceSelectors:
    - apiVersion: apps/v1
      kind: Deployment
      name: sentiment
  placement:
    clusterAffinity:
      clusterNames:
        - member1
        - member2
    replicaScheduling:
      replicaDivisionPreference: Weighted
      replicaSchedulingType: Divided
      weightPreference:
        staticWeightList:
          - targetCluster:
              clusterNames:
                - member1
            weight: 1
          - targetCluster:
              clusterNames:
                - member2
            weight: 1
```

4.Create PropagationPolicy that will distribute sentiment to worker cluster
We need to create a policy to distribute the deployment to our worker cluster. Execute commands as follow:

```
kubectl --kubeconfig /etc/karmada/karmada-apiserver.config create -f propagationpolicy.yaml
```

5.Check the deployment status
We can check deployment status, don't need to access worker cluster. Execute commands as follow:

```
[root@master migu]# ls
deployment.yaml  propagationpolicy.yaml
[root@master migu]# kubectl --kubeconfig /etc/karmada/karmada-apiserver.config create -f deployment.yaml
deployment.apps/sentiment created
[root@master migu]# kubectl --kubeconfig /etc/karmada/karmada-apiserver.config create -f propagationpolicy.yaml
propagationpolicy.policy.karmada.io/sentiment-propagation created
[root@master migu]# kubectl --kubeconfig /etc/karmada/karmada-apiserver.config get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx       8/8     8            8           17d
sentiment   2/2     2            2           17s
```

In worker cluseterwe can see the result as follow:

```
[root@cluster1-master ~]# kubectl get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx       4/4     4            4           11d
sentiment   1/1     1            1           40s
[root@cluster1-master ~]#
[root@cluster1-master ~]#
```

6.Next, we will change deployment.yaml and propagationpolicy.yaml , then retry.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sentiment
  labels:
    app: sentiment
spec:
  replicas: 4
  selector:
    matchLabels:
      app: sentiment
  template:
    metadata:
      labels:
        app: sentiment
    spec:
      imagePullSecrets:
      - name: harborsecret
      containers:
      - name: sentiment
        image: 192.168.30.20:5000/migu/sentiment:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9600
          protocol: TCP
          name: http
        resources:
          limits:
            cpu: 2
            memory: 4G
          requests:
            cpu: 2
            memory: 4G
```

Execute command as follow:

```
kubectl --kubeconfig /etc/karmada/karmada-apiserver.config apply -f deployment.yaml
```

vi propagationpolicy.yaml

```
apiVersion: policy.karmada.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: sentiment-propagation
spec:
  resourceSelectors:
    - apiVersion: apps/v1
      kind: Deployment
      name: sentiment
  placement:
    clusterAffinity:
      clusterNames:
        - member1
        - member2
    replicaScheduling:
      replicaDivisionPreference: Weighted
      replicaSchedulingType: Divided
      weightPreference:
        staticWeightList:
          - targetCluster:
              clusterNames:
                - member1
            weight: 1
          - targetCluster:
              clusterNames:
                - member2
            weight: 3
```

Execute commands as follow:

```
kubectl --kubeconfig /etc/karmada/karmada-apiserver.config apply -f propagationpolicy.yaml
```

7.Retry, Check the deployment status
We can check deployment status, don't need to access member cluster. Execute commands as follow:

```
[root@master migu]# kubectl --kubeconfig /etc/karmada/karmada-apiserver.config get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx       8/8     8            8           17d
sentiment   3/4     4            3           6m5s
```

In worker cluseterwe can see the result as follow:

```
[root@192 ~]# kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
nginx-85b98978db-87lql        1/1     Running   0          3d23h
nginx-85b98978db-9m47b        1/1     Running   0          17d
nginx-85b98978db-dlr2c        1/1     Running   0          17d
nginx-85b98978db-j8tt9        1/1     Running   0          3d23h
sentiment-6fd4c7867c-hvzfd    1/1     Running   0          5m47s
sentiment-6fd4c7867c-j6qs2    0/1     Pending   0          5m24s
sentiment-6fd4c7867c-vrmnm    1/1     Running   0          11m
[root@192 ~]#
```

```
[root@cluster1-master ~]# kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
nginx-85b98978db-7qlv8        1/1     Running   0          11d
nginx-85b98978db-h4tr9        1/1     Running   0          3d23h
nginx-85b98978db-qq9rp        1/1     Running   0          11d
nginx-85b98978db-rnx27        1/1     Running   0          11d
sentiment-6fd4c7867c-jkcqn    1/1     Running   0          11m
[root@cluster1-master ~]#
```

Case 2. Rescheduling

1.First we create a deployment with 2 replicas and divide them into 2 worker clusters.

```
apiVersion: policy.karmada.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: sentiment-propagation
spec:
  resourceSelectors:
    - apiVersion: apps/v1
      kind: Deployment
      name: sentiment
  placement:
    clusterAffinity:
      clusterNames:
        - member1
        - member2
    replicaScheduling:
      replicaDivisionPreference: Weighted
      replicaSchedulingType: Divided
      weightPreference:
        dynamicWeight: AvailableReplicas
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sentiment
```

```
  labels:
    app: sentiment
  namespace: migu
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sentiment
  template:
    metadata:
      labels:
        app: sentiment
    spec:
      imagePullSecrets:
      - name: harborsecret
      containers:
      - name: sentiment
        image: 192.168.30.20:5000/migu/sentiment:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9600
          protocol: TCP
          name: http
        resources:
          limits:
            cpu: 2
            memory: 4G
          requests:
            cpu: 2
            memory: 4G
```

It is possible for these 2 replicas to be evenly divided into 2 worker clusters, that is, one replica in each cluster.

2.Now we taint all nodes in worker1 and evict the replica.

```
$ kubectl --context worker1 cordon control-plane
# delete the pod in cluster worker1
$ kubectl --context worker1 delete pod -l app=sentiment
```

A new pod will be created and cannot be scheduled by kube-scheduler due to lack of resources.

```
# the state of pod in cluster worker1 is pending
$ kubectl --context worker1 get pod
NAME                         READY   STATUS    RESTARTS   AGE
sentiment-6fd4c7867c-jkcqn   1/1     Pending   0          80s
```

3.After about 5 to 7 minutes, the pod in worker1 will be evicted and scheduled to other available clusters.

```
# get the pod in cluster worker1
$ kubectl --context worker1 get pod
No resources found in default namespace.
# get a list of pods in cluster worker2
$ kubectl --context worker2 get pod
NAME                         READY   STATUS    RESTARTS   AGE
sentiment-6fd4c7867c-hvzfd   1/1     Running   0          6m3s
sentiment-6fd4c7867c-vrmnm   1/1     Running   0          4s
```

Test Dashboards

N/A

Additional Testing

N/A

Bottlenecks/Errata

N/A