[RESERVATION] Add support for reservation-based routing-V1

1.Introduction

The purpose of this design document is to explain how to make YARN Router support the Reservation System.

2.Design

2.1 Router Policy

moudle: hadoop-yarn-server-common

package: org.apache.hadoop.yarn.server.federation.policies.router

Contains Various policies of the Router, which specifically control the router's strategy for assigning apps to subclusters.

2.1.1 RouterPolicyFacade

class: org.apache.hadoop.yarn.server.federation.policies.RouterPolicyFacade

This class provides a facade to the policy subsystem, and handles the lifecycle of policies (e.g., refresh from remote, default behaviors etc.).

There are 2 important usage classes

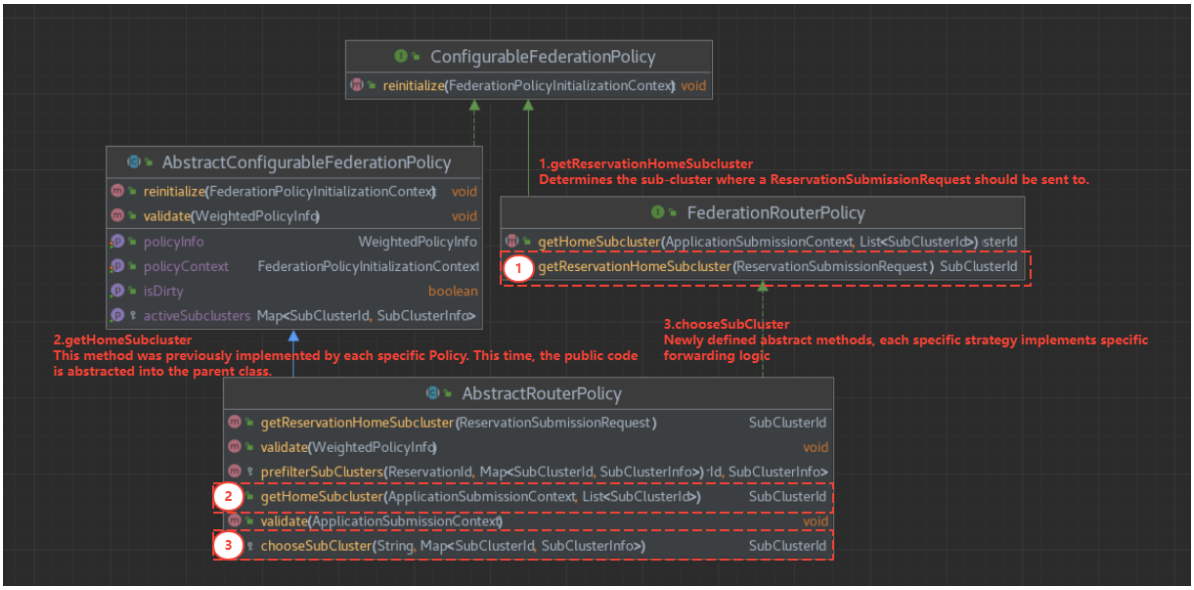
- FederationClientInterceptor / FederationInterceptorREST
    - provides an implementation for federation of YARN RM and scaling an application across multiple YARN SubClusters.

The changes are as follows:

- A new method **getReservationHomeSubCluster** is defined. This method is similar to getHomeSubcluster, Subcluster is selected according to different policies.
- getReservationHomeSubCluster / getHomeSubcluster will share part of the code, extract this part of the logic to avoid code duplication, the new abstract method is **getFederationRouterPolicy**.

2.1.2 RouterPolicy

An overview of the overall modification is as follows:



2.1.2.1 FederationRouterPolicy

class: org.apache.hadoop.yarn.server.federation.policies.router.FederationRouterPolicy

Implements the logic for determining the routing of an application submission based on a policy.

Define new method `getReservationHomeSubcluster`, Determines the sub-cluster where a `ReservationSubmissionRequest` should be sent to.

2.1.2.2 AbstractRouterPolicy

class:org.apache.hadoop.yarn.server.federation.policies.router.AbstractRouterPolicy

This class has 4 changes:

1.Define the abstract method `chooseSubCluster`, this method is used for each specific strategy to select SubCluster to use.

```
 protected abstract SubClusterId chooseSubCluster(String queue,
      Map<SubClusterId, SubClusterInfo> preSelectSubClusters) throws
YarnException;
```

2.Analyzed the code of specific policies (HashBasedRouterPolicy, LoadBasedRouterPolicy ... etc), refactor the `getHomeSubcluster` method, extract it into an abstract class(AbstractRouterPolicy), and eliminate redundant code for each strategy.

The code logic requires the following 4 steps

```
// Step1.null checks and default-queue behavior
validate(appContext);

// Step2.apply filtering based on reservation location and active sub-clusters
Map<SubClusterId, SubClusterInfo> filteredSubClusters = prefilterSubClusters(
appContext.getReservationID(), getActiveSubclusters());

FederationPolicyUtils.validateSubClusterAvailability(
new ArrayList<>(filteredSubClusters.keySet()), blackLists);

// Step3.remove black SubCluster
if (blackLists != null) {
blackLists.forEach(filteredSubClusters::remove);
}

// Step4.pick the chosen subCluster from the active ones
return chooseSubCluster(appContext.getQueue(), filteredSubClusters);
```

3.Define the method `getReservationHomeSubcluster` and implement, This is a generic method that will be used by all policys

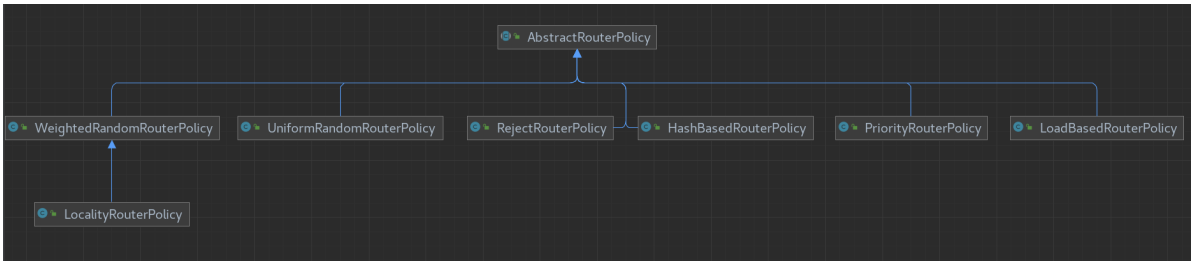
(HashBasedRouterPolicy, LoadBasedRouterPolicy ... etc);

4.Define the method `prefilterSubClusters`, This method will try to find the corresponding SubCluster according to reservationId, if it cannot be found, it will directly return activeSubClusters.

2.1.2.3 Specific Policy Implementation

The policy inheritance relationship is as follows:

The six currently implemented policies all implement the abstract class AbstractRouterPolicy



2.1.2.3.1 HashBasedRouterPolicy

description	This Policy pick a subcluster based on the hash of the job's queue name.
className	org.apache.hadoop.yarn.server.federation.policies.router.AbstractRouterPolicy

Schematic diagram of code logic modification

- Abstract the public code of the original `getHomeSubcluster` to the parent class implementation
- Implement the logic of selecting clusters in `chooseSubCluster`

2.1.2.3.2 LoadBasedRouterPolicy

- Similar to HashBasedRouterPolicy

2.1.2.3.3 PriorityRouterPolicy

- Similar to HashBasedRouterPolicy

2.1.2.3.4 RejectRouterPolicy

- Similar to HashBasedRouterPolicy

2.1.2.3.5 UniformRandomRouterPolicy

- Similar to HashBasedRouterPolicy

2.1.2.3.6 WeightedRandomRouterPolicy

- Similar to HashBasedRouterPolicy

2.2 FederationStateStore

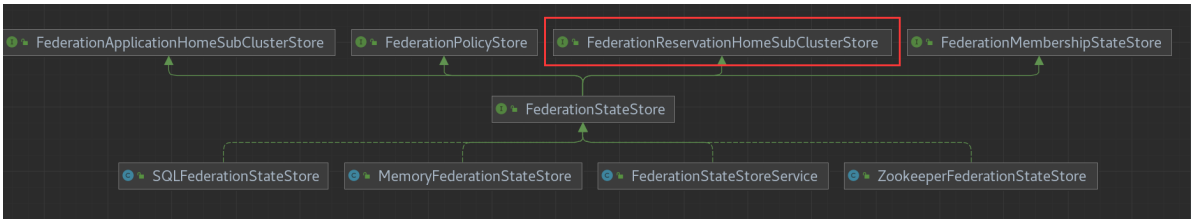
moudle: hadoop-yarn-server-common

package: org.apache.hadoop.yarn.server.federation.store

class: org.apache.hadoop.yarn.server.federation.store.FederationStateStore

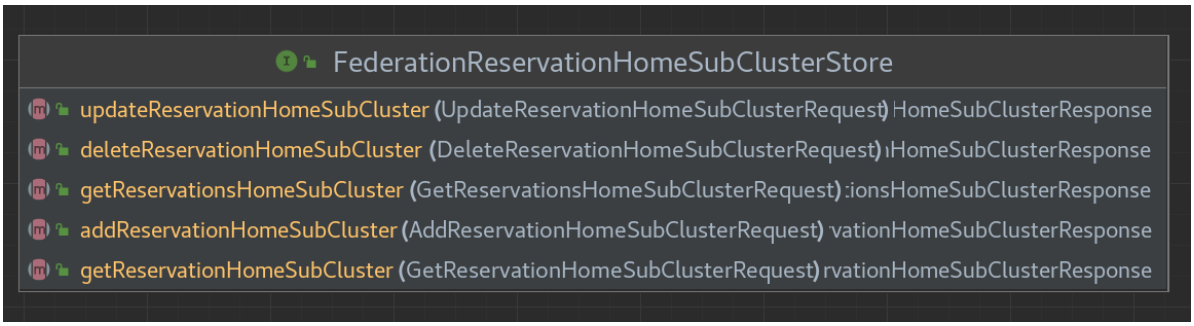
FederationStore extends the four interfaces used to coordinate the state of a federated cluster.

The inheritance relationship of the overall class can refer to the following figure:



2.2.1 FederationReservationHomeSubClusterStore

- maintains the state of all Reservations that have been submitted to the federated cluster.
- Mainly include these 5 methods



2.2.1.1 addReservationHomeSubCluster

Register the home SubClusterId of the newly submitted ReservationId. Currently response is not empty if the operation was successful, if not an exception reporting reason for a failure. If a mapping for the Reservation already existed, the SubClusterId in this response will return the existing mapping which might be different from that in the AddReservationHomeSubClusterRequest.

SerialNumber	Class
1	AddReservationHomeSubClusterResponse.java
2	AddReservationHomeSubClusterRequest.java
3	ReservationHomeSubCluster.java

2.2.1.2 updateReservationHomeSubCluster

2.2.1.3 getReservationHomeSubCluster

2.2.1.4 getReservationsHomeSubCluster

2.2.1.5 deleteReservationHomeSubCluster

2.3 FederationClientInterceptor