Better Queue Management in YARN

Configuration-based Queue Management

Table of Contents

Problems

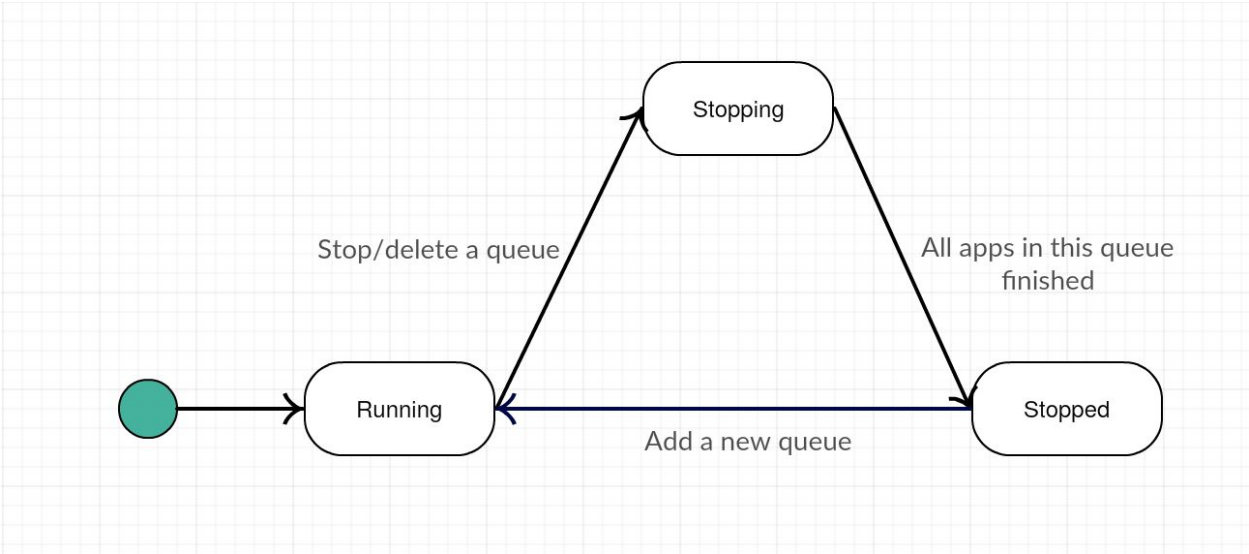
In today's approach (configuration-based) on queue-management in YARN, we still have several places that need to be improved:

- It is possible today to add or modify queues without restarting the ResourceManager, via a refresh-queues feature for the CapacityScheduler/FairScheduler. But for deleting a queue, we have to restart the ResourceManager.
- When a queue is STOPPED, resources allocated to the queue can be handled better. Currently, they'll only be used if the other queues are explicitly setup to go over their capacity.

In this design doc, we will focus on the configuration-based queue management, and how we will improve this approach to solve the existing issues.

Queue State-machine

Each queue will have 3 states: {Running, *Draining*, Stopped}. We will add a simple state-machine to represent the queue. The basic flow will look like the following:



By default (or when adding a new queue), the initial state of the queue will be ***Running***.

1. In the *Running* State:
    a. When we STOP/DELETE a queue, we would transit the state from *Running* state to *Draining* state for the queue as well as all its Children queues.
    b. This means that one can never delete a *Running* queue directly. It will have to first go into a *Draining* state.
2. In the *Draining* state: We have to wait for all the applications in the queue to finish and at the same time we would reject any new applications being submitted to this queue. When all the applications in the queue finish, we would move the state from *Draining* to *Stopped*.
3. A queue can added any time, but the added queue would inherit the same state as its parent.

Also, we must maintain the consistent state for all the queues in a hierarchy:

Parent Queue State	Children Queues State
Stopped	Stopped
Draining	*Draining*/Stopped
Running	Running/*Draining*/Stopped

Configuration-based Queue Management

In order to delete a queue, we should make sure that the queue is in the STOPPED state which means this queue will not accept any new applications, and all the existing application in this queue have already finished. After that, we could delete this queue.

And for re-distribution of stopped/deleted queue. For delete queue it should be obvious, since the queue is gone, sum of its siblings should be 100. For stopped queue, our expectation is, it will be reactivated at some time. So it will be better to keep the capacity as-is, and admin can update max-capacity of its siblings to make sure queue capacity can be utilized.

Overall Work Items

1. Synchronize the state of the parent queue and its child queues
2. Add state machine implementation for Queues
3. Enhancements to STOP queue handling
4. Support for deleting queues without requiring a RM restart