Deprecate Min Share in Fair Scheduler

Yufei Gu

Problem

We should deprecate the min share (or Min Resource) since it is almost useless; not only does it make Fair Scheduler (FS) too complex, but it also confuses users. Using FairScheduler queue properties presents the majority queue use cases in FS. None of them is, however, related to min share.

A Brief Introduction to Min share, Fair Share and Max Share

Queue is the key entity to enforce fairness in FS. Min share, fair share and max share are 3 major queue properties working together to achieve fairness. People normally assume it works as the following way. Their relation is min share < fair share < max share. Max share is the hard limit quota for a queue, which guarantees that no more than max resources can be used in the queue. Fair share is a weight-based share, which is a percentage of all available resources for the queue. Min share is a lower bound, which guarantees that the queue at least get min resources. Those are all true except min share doesn't work in that way. Min share is just another fair share. It is literally the fair share if min share is larger than fair share, described by the U1 in the following section. It acts like another fair share when it is smaller than fair share in terms of determining which application/queue gets resources first and which application is in starvation. There is no guarantee that a queue can get the min share. Why we need the min share? Can we deprecate it or consolidate it into fair share?

For example, assume there are 30G memory and 30 vcores in the clusters. The following queue configuration shows that queue A has:

- Min share: <2G, 2 vcores>
- Fair share: <20G, 20 vcores> two third of whole resources
- Max share: <25G, 25 vcores>

```
<queue name="root">
   <queue name="queueA">
     <weight>2</weight>
     <minResources>2048 mb, 2 vcores</minResources>
     <maxResources>25600 mb, 25 vcores</maxResources>
   </queue>
   <queue name="queueB">
     <weight>1</weight>
   </queue>
 </queue>
```

If we change the min share to <21 G, 21 vcores> then its fair share is <21 G, 21 vcores>.

How Min Share is used in FS

Min share is involved in three places of the fair scheduler: **fair share calculation**, **scheduling**, and **min share preemptions**.

U1. Fair share calculation

Min share in fair share calculation is only used to set queues' fair share if it's larger than weight-based fair share. However, this behavior would lead to a 0-fairshare queue if the sum of min share of its sibling queues is larger than parents' fair share, which makes sense but confuse users. The calculation would be simpler and less confusing if we remove min share.

U2. Schedulable priority comparison in scheduling

A schedulable could be a queue or an application. Min share is used to compare two schedulables and decide which one has higher scheduling priority. It only makes sense if we assume min share is less than weight-based fair share for a queue, which isn't always true. It duplicates the fair share comparison if min share is larger than fair share. It doesn't work for applications. Overall min share is quite useless in schedulable priority comparison.

U3. Min share preemption

Min share preemption was there before fair share preemption. We keep it for the sake of compatibility. However, min share preemption is counter-intuitive, and somehow messes up overall preemption functionality.

Firstly, it mainly duplicates the functionality of fair share preemption. Secondly, it conflicts with fair share preemption in some ways. For example, it is used in starvation detection phase but not in identifying which container to kill. Because of that, an application involved in min share preemption can preempt itself (YARN-8061). Thirdly, it makes preemption complicated to understand. All things considered, we should deprecated min share preemption.

Pros:

- provide a similar solution to fair share with lower bound

Cons:

- It is confusing, most users probably don't know how to use it. A major use case is to set customized fair share since you can't set fair share with different weights for different resources.
- It makes scheduler too complex, see details in U3.

In conclusion, min share, as well as min share preemption should be deprecated to make the scheduler more consistent, less buggy and less confusing.

Proposal

Use min share to set customized fair share

We can still keep min share setting by using it as a customized fair share, described by U1. Furthermore, we need to update fair share even min share is smaller than fair share, which doesn't exist in current fair share computation logic, but quite useful.

Deprecate the internal implementation of min share

Deprecate the usage of min share in U2 and U3, and remove them in the next major release.