YARN-4108 Lazy Preemption Design - V3

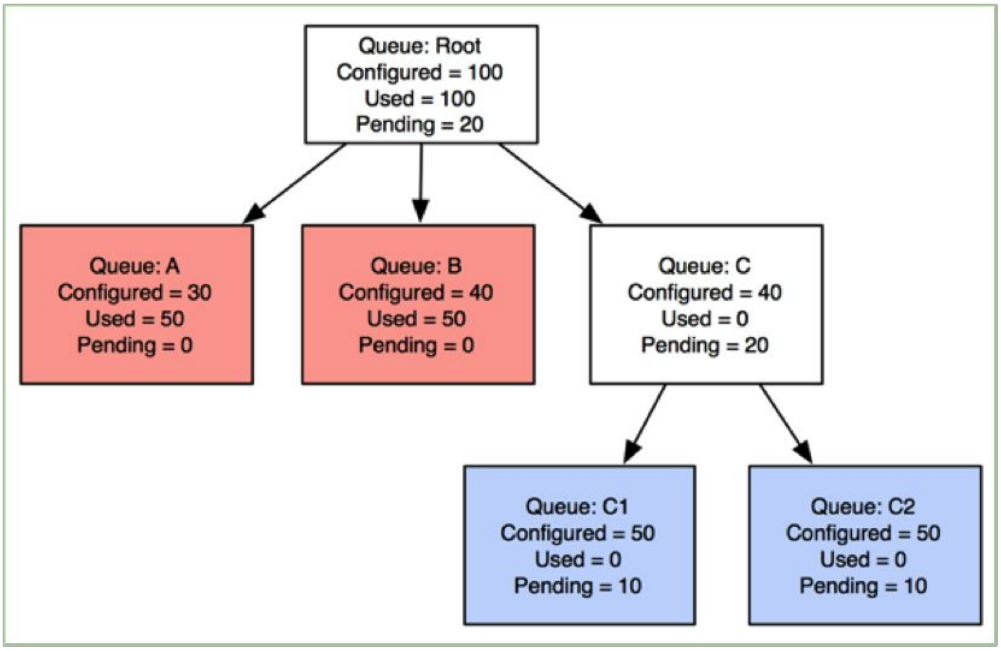
Authors: Wangda Tan **with inputs from** Ram Venkatesh, Carlo Curino, Chris Douglas.
Last Modified Date: Jan 13, 2015

Problems

A little bit of background

Currently, YARN's Capacity Scheduler preemption works in following way:

- Preemption monitor gets queues resource usage information (such as used/reserved/pending resource) from scheduler **periodically**. Like following graph:



- Preemption monitor calculates how much resource to preempt from queues according to queues' resource usages.

- Once preemption monitor finalizes how much resources to preempt, it selects containers from queues to preempt based on many factors:
    - One of the most important considerations is, app should have minimal impact if it has container preempted.
- For more details please refer to blogpost about Capacity Scheduler preemption.

Issues with the current approach

- There's no guarantee that pre-empted resources could be used by pending resource request from under-satisfied queues, there are several causes:
    - Preemption monitor considers **aggregated** pending resource request only: In real world, pending resource request from a queue (queue-A) may look like:
        - App-A needs 3 * 2G containers allocated on /rack1, and it needs 4 * 1G containers allocated if former requests satisfied
        - App-B needs 5 * 3G containers, but it can get resources only if App-A's requests satisfied
        - But in preemption monitor's perspective, queue-A's pending resource = 3 * 2G + 4 * 1G + 5 * 3G = 25G
        - As a result, preemption monitor could choose 25 * 1G containers on 25 different hosts, none of these resources could be used by queue-A.
    - Preemption monitor doesn't consider limits of queues, for example
        - A queue has user-limits to limit how much resources could be used by one user. In the future we could add limit of individual applications as well. (already covered by YARN-3769)
        - Similar to above cause, this could also lead to preempted resources that cannot be used by under-satisfied queues with pending resource request
    - Locality is not considered while doing preemption.
- Above could lead to excessive preemption as well: preemption monitor keep preempting resources from queue-A, but queue-B cannot use preempted resources, so resources will be allocated to queue-A again, then preempting again...

Proposal

Previously we proposed an approach that uses scheduler logic to make sure every selected containers could be used by other applications. See previous design doc.

That sounds like a doable idea, but it has some potential issues:

- Performance: It could add lots of overhead to scheduler.
- Quality of selected to-be-preempted containers: It sounds very good, but we don't have real data to verify if it is true.

- Effort of Implementation: Implementation of the “dryrun” part could mess existing scheduler logic: we need to add “dryrun” flag everywhere (see previous monster POC patch).

So here is a more conservative proposal:
As mentioned above, one of the most critical issue of our existing implementation is excessive preemption.

If we can make sure there’s less excessive preemption happens, we can more confidently ask users to enable this feature. How to select containers could be continuously optimized in the future.

Steps

- Leave existing PCPP as-is, it computes ideal share of each queue, and select to-be-preempt containers like before.
- When a container expires kill-wait (could be killed), “kill container” message will be sent to scheduler, and scheduler marks container to be “killable”
- When scheduler trying to allocate an under-satisfied queue, it will try to deduct resources of “killable” containers. If a new container could be allocated/reserved after killing these “killable” containers. Scheduler will pull the trigger immediately.

In short, this proposal avoids awkwardness that preempted resources cannot be used by who needs the resource, and then such preempted resources come back to over-utilized queue again.

Advantages of this proposal

- Containers will be killed only when demanding application can use that capacity on the specific nodes
- Application requests that cannot be satisfied currently for some reason (queue, user or app limits or other constraints) will not cause any containers to be killed.
- For any reason if the application request gets canceled (including application exit), no containers will be killed.

More details

Performance:

- Performance of this proposal should in the same level of existing scheduler logic. For each container allocation, it takes an additional O(#killable-container-on-the-host) time. And #killable-container-on-the-host is a small constant.

Cancel killable containers:

- When a container is marked “killable”, but it isn’t killed by anyone. We can cancel killable containers
- When a queue has killable container, but resource usage of the queue back to under its guaranteed resource, killable containers will be cancelled.

Container reservation and killable containers

- In existing scheduler we will preempt containers to reserve new container, but it is possible that the reserved container will be allocated at a different node or cancelled. Such container preemption is inefficient.
- To solve this problem, we should only preempt containers when we’re able to allocate the reserved container. However, it could lead to used resource + reserved resource greater than queue’s max capacity.
- We should properly modify container reservation logic to make this could happen in the future. In short term, for simply, we could preempt containers for container reservation.

Not included by this proposal (To be improved in the future)

- How to select containers to be preempted: now to-be-preempted container selection doesn’t consider pending resource request, which needs to be improved.

Related JIRAs

- YARN-2009 Priority support for preemption in ProportionalCapacityPreemptionPolicy
- YARN-2069 CS queue level preemption should respect user-limits
- YARN-2113 Add cross-user preemption within CapacityScheduler's leaf-queue
- YARN-2154 FairScheduler: Improve preemption to preempt only those containers that would satisfy the incoming request
- YARN-3510 Create an extension of ProportionalCapacityPreemptionPolicy which preempts a number of containers from each application in a way which respects fairness
- YARN-4390. Consider container request size during CS preemption