YARN-4108 Lazy Preemption Design - V1

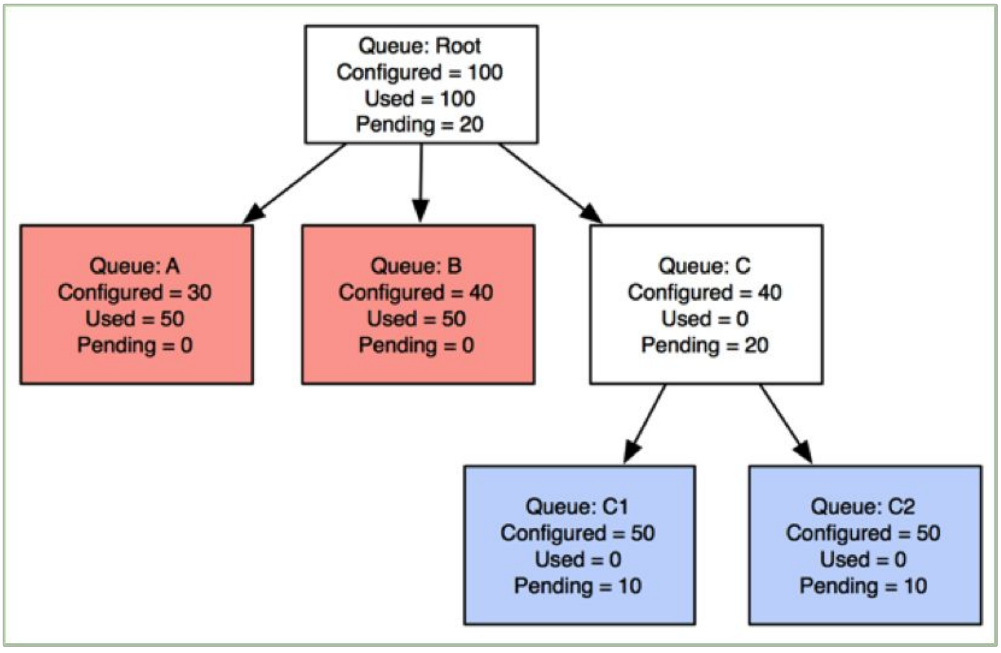
Wangda Tan with inputs from Ram Venkatesh

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.

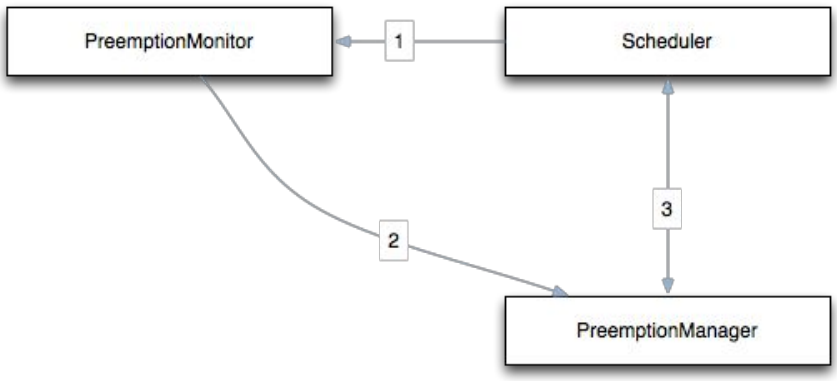
- For more details please refer to blogpost about Capacity Scheduler preemption.

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

Lazy container preemption: Use a 2-phase algorithm to pre-empt containers according to ideal allocation calculated by PreemptionPolicy.



- **Phase 1** PreemptionMonitor periodically pulls scheduler data (1) and calculates ideal share of each queue. (Just like original PreemptionPolicy), ideal share of each queue will be sent to PreemptionManager. (2)
- **Phase 2** In scheduler's logic, we will add a "dryrun" flag to indicate the allocation won't change scheduler's internal state. We will look at if it possible to preempt some containers from over-utilized queues for requests from under-utilized queues while doing "dryrun" (3).

More details:

- PreemptionManager is the central place to store preemption info, it knows
  - how much resources we can preempt from each preemptable-entity and how much resources are already marked to-be-preempted
  - For each to-be-preempted container, PreemptionManager stores which application will consume the preempted resources.
  - Once containers are marked to-be-pre-empted for a specified time, PreemptionManager will send a message to scheduler to kill such containers.
- Scheduler's dryrun logic to check container allocation and preemption
  - Scheduler's dry-run is adding an additional flag to existing allocation code path, it do everything required of container allocation, including but not limited to checking limit / sort requests, etc. But it won't modify scheduler's state.
  - To reduce overhead of preemption checking, we can run such dryrun check once for every X secs of each node. Assume a node has 1 sec heartbeat interval, if we run dry-run once every 10 secs, it only adds 10% additional time to scheduler.
- Scheduler could optimize selected to-be-preempted containers along with allocation, when scheduler find a better combination to preempt, it will update to-be-preempted containers.
- How to calculate to-be-preempted resource and how to select containers to be preempted should come in pluggable ways.
- Preemption manager could decide which nodes need dryrun check to avoid excessive preemption check. (For example, if we have enough resources available on the node, we don't need do dryrun)

Topics for discussion

Preemption within a queue

We should support preemption within a queue when fair-sharing / priority policy enabled.

Some thoughts:

- PreemptionMonitor can calculate how much resources can be used/preempted for entities within a queue, such as user/app.

- With above, PreemptionManager can select to-be-preempted containers within a queue as well
- Preemption within a queue will be triggered if the demanding app cannot get resource from other queues. (avoid applications within a queue shooting each other).

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