YARN-4108 Lazy Preemption Design - V2

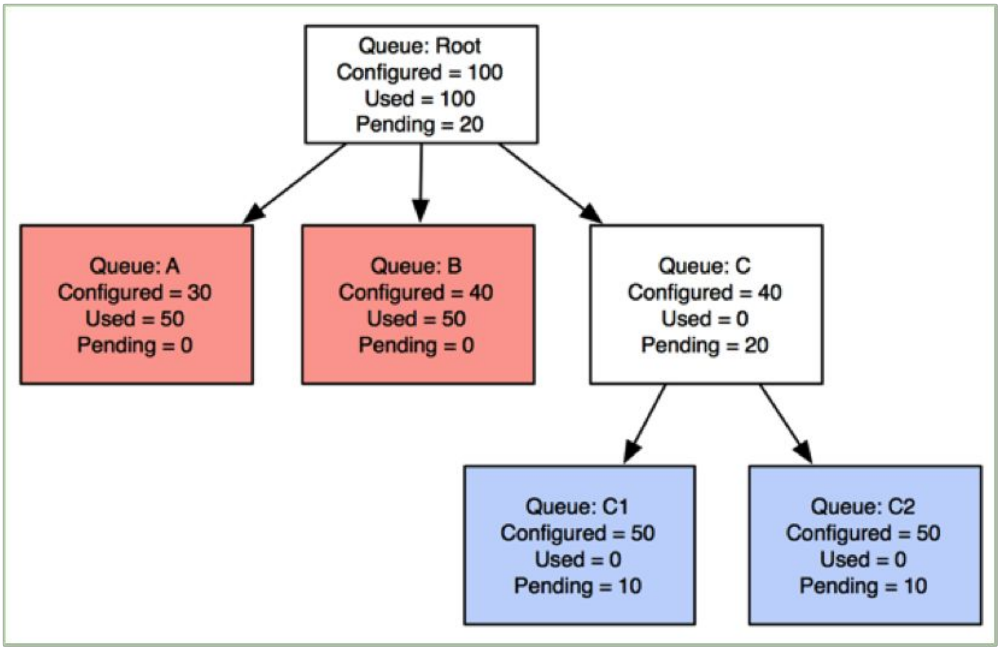
Authors: Wangda Tan with inputs from Ram Venkatesh, Carlo Curino, Chris Douglas.
Last Modified Date: Dec 15, 2015

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:



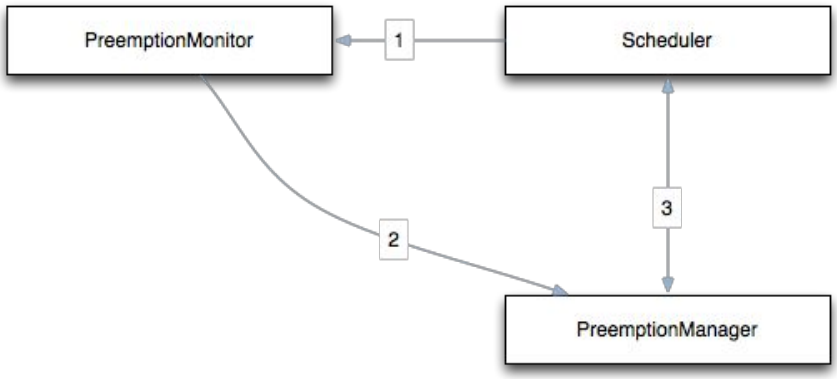
- Preemption monitor calculates how much resource to preempt from queues according to queues' resource usages.
- Once preemption monitor finalizes how much resources to preempt, it selects containers from queues to preempt based on many factors:
    - One of the most important considerations is, app 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

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

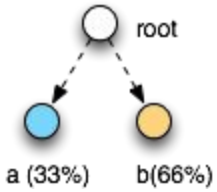


- **Phase 1** PreemptionMonitor periodically pulls scheduler data (1) and calculates ideal share of each queue. (Just like original PreemptionPolicy), ideal share of each queue will be sent to PreemptionManager. (2)
- **Phase 2** In scheduler's logic, we will add a "dryrun" flag to indicate the allocation 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).
    - Scheduler's dryrun means we don't change any state within scheduler, but we will update statuses in PreemptionManager (which containers to preempt).
    - In another word, containers-to-preempt are decided together with scheduler doing resource allocation (a.k.a node heartbeat).

Example

An example to explain how lazy preemption works.

Cluster status:
Two queues, a and b, a's configured capacity is 33% and b's configured capacity is 66%.

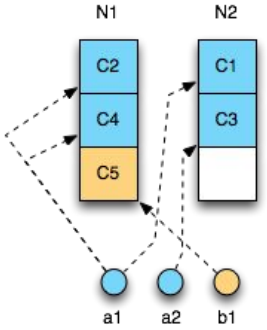


There're two applications (a1/a2) in a, and one application in b (b1).
a1 has 3 containers, c1/c2/c4
a2 has 1 container, c3
b1 has 1 container, c5;

Containers c1-c5 are 1G mem.
In addition, b1 has a pending resource request for one container, size = 3G.

There're two nodes, n1/n2 in the cluster, each of them has 3G mem.

Relationship between containers/applications/nodes see below:



Calculate ideal resource allocation:
In existing ProportionalCPP algorithm, queue=b should preempt 2G resource from queue=a.
We will send max-preemptable information to PreemptionManager:

a = -2G b = 2G

Scheduler's dryrun and select containers to preempt:
When dryrun=true, and N1 doing heartbeat:

1. Available resource in N1 is 0, **normally we won't continue, but now we will assume there are some resources available so we can continue.**
2. CapacityScheduler goes to queue=b, then application=b1, then traverses to the 1 * 3G resource request.
3. It will do all checks, such as queue-limit/user-limit/locality-delay. Assume now it is ready to allocate the 3G resource request:
    a. Scheduler asks PreemptionManager, is it possible to get 3G resource in N1?
    b. PreemptionManager gets running container info, only c2/c4 are preemptable, so it returns false to tell scheduler there's no way to get 3G resource in N1.

When dryrun=true, and N2 doing heartbeat:

1. Same as N1
2. Same as N1
3. Same as N1
    a. Scheduler asks PreemptionManager, is it possible to get 2G resource in N2. (N2 has 1G mem available)
    b. PreemptionManager gets running container info, c1/c3 are preemptable, so it marks c1/c3 to-be-preempted, and return true to tell scheduler it can allocate new container.
    c. PreemptionManager puts c1/c3 to to-be-preempted container list

d. RM sends to-be-preempted container list back to AM.

More details

Scheduler's dryrun logic

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.
- 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)

Who/How to select which containers to be preempted?

In existing PCPP, it selects containers according to FIFO order of apps and containers globally:

- Let's say a queue has 5 apps: a1-a5, sorted by submission order. When doing preemption, containers of a5 will be selected first.
- Containers within an app will be selected according to sequence of when container is allocated, recent allocated containers will be selected first.

Now we will select container in when doing resource allocation (a.k.a node heartbeat).

Who selects to-preempt containers?

- Instead of selected by PCPP, with this proposal, PreemptionManager selects which containers to be preempted.

How to select?

- This should be supported by pluggable policy. Different scenarios need different ways to select to-be-preempted containers. For example, we can select containers based on priority, fairness of application/users, etc.

Topics for discussion

Is it possible to combine batch preemption and surgical preemption?

Avoid doing surgical preemption (preemption in scheduler's logic) for every container, we could only do surgical preemption for hard-to-allocate (such as large-sized, hard locality, etc.) resource requests.

For other resource requests, we can use existing PCPP logic to do preemption.

Potential benefit of combining the two approach is to avoid adding too much overhead to scheduler: we don't have to run scheduler allocation test for every resource request.

Kill container at NM only when new container request arrives.

When a container is marked "can be killed" from PreemptionPolicy. Instead of killing container directly at NM. RM sends a special "preempt container" message to NM. NM marks the "preempt container" and only kills these container process when the new launched container is ready to go. (Finished localization, etc.).

Typically we need several seconds for AM to receive/launch container, and another several seconds for NM to do preparation such as localization, etc. for the new container. This approach could improve cluster utilization.

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