[Design] YARN-4390: Select preemption candidates based on resource requests

Wangda Tan with input from Eric Payne, Carlo Curino and Sunil G.

Problems

A little bit background:

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

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.

Proposal

I propose to add a new scheduler candidate selection logic based on reserved containers:

Currently, schedulers in YARN reserve resources for picky requests, such as large resource request, hard locality, complex constraint (such a node has GPU && JDK9 && SSD, will be added after YARN-3409).

To make surgical preemption could happen, we can select candidates based on reserved containers. Because once a container is reserved, we're pretty sure that the node can be used by the pending resource request after it frees resources.

Instead of changing existing preemption candidates selection logic, we can make preemption candidates selected by multiple steps:

- Calculate ideal allocation for queues, we can get a guideline about how to fix high-level imbalances between queues
- According to reserved containers, do surgical preemption candidates selection, maximize allocations of reserved containers.
- After we have done surgical preemption, we can select rest preemption candidates.

Prerequisites / potential issues we need to be aware of this approach:

1) Picky resource requests need to be reserved before we can preempt resources for them

If we assumes surgical preemption only happens on reserved container, we have to make sure resource reservation should happen in expected cases.
YARN-4280 is one of the issues that scheduler cannot reserve large resource when utilization is high.

2) Preemption policy should consider reserved resources while calculating ideal resource allocation

For example, there are two queues, A and B, if their usage is as below:

Queue	Configured	Used	Reserved	Pending	Ideal
A	20	35	0	0	35
B	20	0	5	0	5

Queue B under its configured resource, its only resource request is reserved. So computed ideal resource of B = its reserved resource.
In existing preemption logic, preemption cannot happen because it equally treat reserved/used resources.
We need to update ideal resource allocation logic to make preemption could still happen in above cases.

3) Be able to find better places for container reservation according to resource request

When preemption policy look at nodes, it should be able to find better places for container reservation. For example:

- When a node (required resource=node's total resource) is reserved by queue-A, and the node is used by a long running service container that we cannot preemption. We may need to move resource reservation to a different node because there's no SLA guarantee at all if we reserve container on the node.

- A better node for reservation, for example: better locality, larger available memory, etc.

Alternative proposals

Do preemption together with container allocation

This is original proposal of YARN-4108. (See original design doc, patch). This proposal looks very promising. However, this approach also increases allocation cost a lot, and it changes lots of scheduler's internal allocation logic as well.

For other issues/concerns of this approach, please see comment.

Scan pending resource request to find best place

A similar approach is scan pending resource requests in preemption policy to find best places to allocate resources. In another word, for an underutilized queue, we should be able to compute how much resources of each apps should get from other over-utilized queues. (Ideal allocation of each apps).

However, it could be very hard to be pre-computed:
- When FIFO/Priority ordering policy (default CS ordering policy) is enabled , we need to consider user limits and AM limits. One app in front of a queue may not be able to get more resources if user-limit is hit.
- When fair ordering policy is enabled, we have to re-sort apps after allocate resource to next least satisfied app.

If we cannot precisely get the **set of** resource requests which need to be satisfied, we cannot find best places to preempt containers.

Plan

1) Refactor existing preemption policy implementation so we can easier add new preemption candidates selection logic. (YARN-4822)
2) Implement preemption-candidates-selection logic for reserved containers. (YARN-4390)
3) Improve preemption candidates selection logic so preemption policy (or scheduling edit policy) can request a better place for reservation. (To file a JIRA)

Related JIRAs

1) YARN-4280, CapacityScheduler reservations may not prevent indefinite postponement on a busy cluster

2) ~~YARN-4108~~, CapacityScheduler: Improve preemption to only kill containers that would satisfy the incoming request