Proposal for “Gracefully Decommission of NodeManager” (YARN-914)

Scenarios and Use Cases

One of important design goals for YARN is easier and more smoothly for operation and maintenance on a cluster with thousands of nodes. A major challenge for daily hadoop cluster operation is upgrade of software stacks on each node.
Some of software upgrades, e.g. YARN/HDFS components, happens in the hadoop layer and doesn’t involve the reboot of node or JVMs. These upgrades can be well addressed by Rolling Upgrade feature that get supported by YARN since 2.6 release based on feature of RM/NM work preserving. The NM daemon can be shutdown temporarily for upgrade and bring back without losing running containers and states. However, there are also other cases that software upgrades happen below the layer of hadoop and have to involve the reboot of nodes (e.g. OS upgrade) or relaunch of JVMs (e.g. upgrade of JDK or JRE).

For these scenarios, operation flow today is to decommission these nodes first, then do recommission after finish of upgrades which has many outstanding issues:

- Regular running containers get lost and will have to be reassigned with new attempts (waste of time and resources)
- AM container (even closed to be finished) could get lost and reassigned to other node with increase of App attempts. In some cases, this reassignment can happen again and again that potentially cause app failure.
- Without any notification and timeout mechanism, AM running on a decommissioning node can hardly find ways to migrate its own states to a new node that already upgraded. AM has to pay more price to recovery its state from somewhere to survival from regular upgrades but not failure casually.
- Current operation flow (based on node decommission) restraint the number of nodes being upgraded at the same time if user want their applications can keep running which cause the upgrade of large cluster seems like an endless process.

Thus, we need a new mechanism that could decommission nodes in a more “graceful” way.

Key Requirements

- NodeManager can be decommissioned after running app/containers against the node get finished rather than immediately. We call the node in stage of “decommission in progress” before it get decommissioned finally.
- A new CLI is provided to wait nodes’ new decommission process which accept a timeout as input. After timeout, it will call today’s forcefully decommission on nodes still in “decommission_in_progress”(or all nodes).
- If CLI get interrupted, then it won’t keep track of timeout to forcefully decommissioned left nodes. However, nodes in “decommission in progress” will still get terminated later (after) except admin call recommission CLI on these nodes explicitly.

- When NM in “decommission_in_progress”

➔ Containers can keep running to the end, NodeManager’s services (like shuffle) still serve as a normal node, but no new containers get allocated on this NM.

➔ NodeManager will be terminated (decommissioned) if no live containers and no running apps consume any services on this NM.

➔ AMs who has running containers on this NM will get noticed, especially AM itself run against on this node. AM is supposed to handle properly for this situation later.

➔ NM in the state of “decommission in progress” can be rollback to normal nodes by recommission on this node.

➔ Before NM decommissioned, the timeout can be updated to shorter or longer

- (Optional) Notify admin (in log or UI) that NMs are decommissioned gracefully (with all related apps completed) or get timeout that help admin to set better timeout value for future.

- The command line to “gracefully” decommission nodes keep almost the same as the old one as “yarn rmadmin -refreshNodes” with a new option “-g” (gracefully or gently), no new configuration get added at this moment.

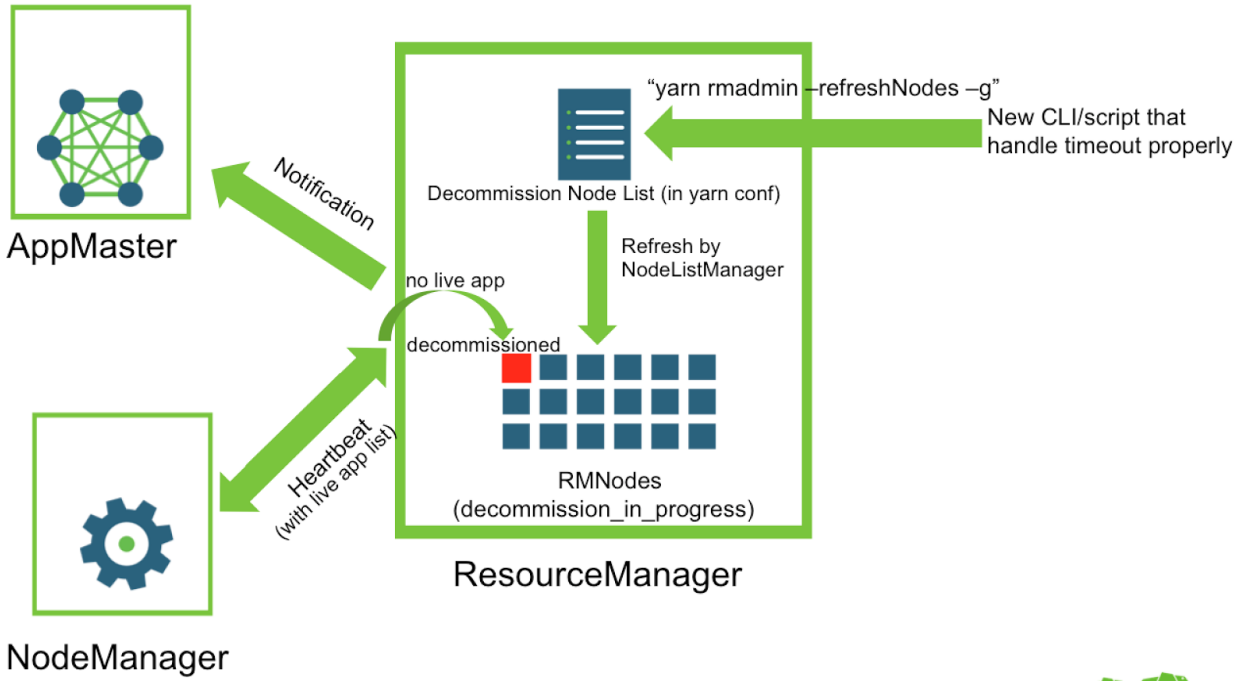
- UI (and CLI output) changes for listing NM’s status, a new state of “decommission_in_progress” will be added.

- Works well with other cases, e.g. NM being restart with work preserving (Rolling Upgrade) or RM in transition between active and standby (RM HA). Nodes in stage of “decommission_in_progress” need to get stored in persistent storage to survival from RM failed over (tracked by YARN-2567).

Design

1. Architecture

The overall architecture for this feature is as following:



When triggered by user's call on new decommission CLI, it will call RMAdmin CLI to notify ResourceManager to decommission nodes gracefully. NodeListManager running inside RM will load node list that specified in a configuration file (specified in "yarn.resourcemanager.nodes.exclude-path") and send out events to notify related RMNodes (state machines within RM to represent NM) that they are being decommissioned. The states of these nodes will be marked as **decommission_in_progress** and their resources will be deduced from YARN cluster (by calling APIs provided by YARN-291 for node's resource adjust in runtime). This state change only happens in RM side, and NM doesn't need to aware any of it.

Then, whenever next heartbeat comes from these NMs, ResourceTrackerService in RM will check that if live app list is empty from heartbeat messages. If so, update RMNode state to decommissioned and send back shutdown response to these NMs just like what we do today.

In the mean time, AM can pull the states of related NMs that in decommissioning (may leverage existing preemption framework) to get understand potentially resource change.

2. More Details

New state and state transition event for RMNode
A new state of “decommission_in_progress” will be added and can transition from “running” state triggered by a new event, may call “gracefully_decommission”. This new state can be transit to state of “decommissioned” when ResourceTrackerService detect that all apps get completed from the heartbeat of NM.
Also, it can back to “running” if user decides to cancel previous decommission by calling recommission on the same node.

Resource update from RMNode decommission/recommission
When node transit from RUNNING to DECOMMISSION_IN_PROGRESS, it will send RMNodeResourceUpdateEvent to update its resource to 0 (and record the previous value for rollback later). If recommission is triggered, it add previous resource value back with another RMNodeResourceUpdateEvent.

New parameter for “-refreshNodes” in RMAdmin CLI
When the “refreshNodes” with an additional parameter like “-g” (means gracefully or gently) get triggered in RMAdmin CLI, it will call a new method call regreshNodesGracefully() in NodeListManager which will send the new RMNode event - DECOMMISSIONING to related RMNodes to make them transite to new state of “decommission_in_progress”.

Persist new RMNode state of decommission_in_progress
RMNode in state of “decommission_in_progress” need get persisted to across RM failed over (see more details in YARN-2567)

New CLI/script for monitoring gracefully decommission process with timeout

New configurations
No.

New UI page or contents
TBD

3. Open Questions

RM in failed over (with HA enabled) when gracefully decommission is just triggered. We should make sure the new active RM can carry on the action forward. (We will persistent RMNode state to across RM failed over in YARN-2567)

NM restart with work preserving happens before or after NM being decommissioned (not finished yet). What behavior is expected here?

AM could get notified that undergoing NM is being decommissioned, how AM deal with it (like migrate its states)?

With containers of long running services, the timeout may not help but only delay the upgrade/reboot process. Shall we skip it and decommission directly in this case? (No need so far, but add some diagnostic info is better)

Another possibility is to track decommission timeout in RM side, instead of NM side - a new decommission services proposed above. Which way is better? (We decide to track all things in RM side, the timeout even be moved outside of core of YARN)

4. Work items BREAK DOWN

Plan to file several sub JIRAs as below:

New state and state transition event for RMNode gracefully decommission

Resource update properly, especially to make RMNode keep track of old resource for possible rollback

New parameter for RMAdmin CLI

New CLI/script for monitor gracefully decommission process with timeout

Add diagnostic info for long running services

Provide AM notification

Minimum UI changes