Automatic and Asynchronous YARN Nodes Graceful Decommissioning

Summary

YARN-4676 implements an automatic, asynchronous and flexible mechanism to graceful decommission YARN nodes. After user issues the refreshNodes request, ResourceManager automatically evaluates status of all affected nodes to kick out decommission or recommission actions. RM asynchronously tracks container and application status related to DECOMMISSIONING nodes to decommission the nodes immediately after there are ready to be decommissioned. Decommissioning timeout at individual nodes granularity is supported and is dynamically updatable. The logic naturally supports multiple independent graceful decommissioning “sessions” where each one involves different sets of nodes with different timeout settings. Such support is ideal and necessary for graceful decommission request issued by external cluster management software instead of human.

NodesListManager detects and handles include and exclude list changes

"yarn rmadmin -refreshNodes -g [timeout in seconds]" notifies NodesListManager to detect and handle include and exclude hosts changes. NodesListManager loads excluded hosts from the exclude file as specified through the "yarn.resourcemanager.nodes.exclude-path" configuration in yarn-site.xml.

NodesListManager inspects and compares status of RMNodes in resource manager and the exclude list, and apply necessary actions based on following rules:

- Recommission DECOMMISSIONED or DECOMMISSIONING nodes that are no longer excluded;
- Gracefully decommission excluded nodes that are not already in DECOMMISSIONED nor DECOMMISSIONING state;
- Forcefully decommission excluded nodes that are not already in DECOMMISSIONED state should -g flag is not specified.
- Accordingly, RECOMMISSION, GRACEFUL_DECOMMISSION or DECOMMISSION RMNodeEvent will be sent to the RMNode.

Per-Node decommission timeout support

Hosts could be subject to shutdown/terminate with short notice. For example, EC2 SPOT instance could be shutdown with 2-minute termination notice. In such cases, we like to graceful decommission such nodes with a small timeout that is before the anticipated termination time. It will avoid Hadoop re-actively deal with LOST instances where stop all containers could subject to long timeout delay.

Further, we support the flexibility to dynamically adjust the timeout of DECOMMISSIONING nodes. **HostsFileReader** now supports optional timeout value after each hostname (or ip). **NodesListManager** detect and update decommission timeout on individual RMNode as necessary. DecommissioningNodesWatcher evaluates timeout based on the dynamic decommission timeout on individual RMNode.

Automatic and asynchronous tracking of decommissioning nodes status

DecommissioningNodeWatcher inside ResourceTrackingService tracks DECOMMISSIONING nodes status automatically and asynchronously after client/admin made the graceful decommission request. ResourceTrackerService handles node registration and frequent heart beat with latest contain status so it is a natural place to hook up automatic decommissioning status tracking logic. ResourceTrackerService now tracks DECOMMISSIONING nodes to decide when, after all running containers on the node have completed, will be transitioned into DECOMMISSIONED state (NodeManager will be told to shutdown).

Under MR application, a node, after completes all its containers, may still serve it map output data during the duration of the application for reducers. A fully graceful mechanism would keep such DECOMMISSIONING nodes until all involved applications complete. It could be however undesirable under long-running applications scenario where a bunch of "idle" nodes would stay around for long period of time. DecommissioningNodesWatcher balance such concern with a timeout policy --- a DECOMMISSIONING node will be DECOMMISSIONED no later than DECOMMISSIONING_TIMEOUT regardless of running containers or applications. If running containers finished earlier, it continues wait up to DECOMMISSIONING_TIMEOUT for the applications to finish. Following are the sub status of a decommissioning node:

- NONE --- Node is not in DECOMMISSIONING state.
- WAIT_CONTAINER --- wait for running containers to complete.
- WAIT_APP --- wait for running application to complete (after all containers complete)
- TIMEOUT --- Timeout waiting for either containers or applications to complete
- READY --- Nothing to wait, ready to be decommissioned
- DECOMMISSIONED --- The node has already been decommissioned

Status of all decommissioning node are logged periodically (every 20 seconds) in human friendly format to resource manager logs.

To be efficient, DecommissioningNodesWatcher skip tracking application containers on a particular node before the node is in DECOMMISSIONING state. It only tracks containers once the node is in DECOMMISSIONING state. DecommissioningNodesWatcher basically is no cost when no node is under DECOMMISSIONING. This sacrifices the possibility that an idle node once host containers of a still running application.

When decommissioning node TIMEOUT, it will be decommissioned regardless. Proper events will be send to ensure the node be deactivated and owning tasks will be rescheduled as necessary.