YARN-796: Node-labels - Requirements & Design doc - V1

Authors: Wangda Tan, Vinod Kumar Vavilapalli with inputs from Arun C Murthy, Ram Venkatesh, Bikas Saha, Junping Du

Last Modified Date: July 7, 2014

What is a node-label?

- Label of a node represents a non-tangible-resource (see next) available on the node.

Use cases

- YARN currently doesn't have the ability to segregate nodes in a cluster based on features like the Operating System, processor architecture etc. So, applications requiring to run only on specific nodes in the cluster cannot do so inside YARN. Node-labels can help applications flock to nodes of their choice.

- Many applications need specific hardware such as GPU, FPGA etc to run against. Many a times, only a subset of a cluster will have this hardware installed due to several reasons like cost and non-pervasive usage. We can use node-labels to tag machines with such specialized hardware and let users run applications against them.

- Admins may want to logically partition (not physically) a large cluster to several smaller units, each partitioned unit can be used by one or more teams, like node[1-5] for marketing team, node[6-10] for financial team. This is a use-case that has so far been antithetical to the philosophy of YARN to elastically share cluster resources (and not

hard-partition them) with as high utilization and throughput as possible, but in sites where it is the only acceptable solution, node-labels can be used, if not strongly suggested.

Requirements

Definitions

- Label of a node represents a non-tangible-resource (like arch, os, etc.) available on the node.
- Tangible-resources (like bandwidth/memory/disk) shouldn't be a part of node-labels and should be explicitly modeled as a YARN resource.

Top level requirements

- One node can have multiple admin-specified labels (os, arch, dept etc.).
    - We should have limits on the number of labels per node so as to ease the management of system's book-keeping requirements
    - We should also optionally define a list of cluster-level acceptable labels to avoid configuration mistakes.
        - This list should be dynamically changeable just like node-to-label mappings
- There can be nodes that have no labels associated with them.
- One label can be associated with multiple nodes.
- There can be labels that have no nodes associated with them.
- Labels should be persistent across cluster component restarts (ResourceManager, NodeManager etc.) and cluster upgrades.
- Applications should be able to request for containers on nodes by specifying a list of labels.
- **Admin tools**:
    - Admins should be able to associate/un-associate labels with nodes dynamically.
    - Admins should be able to list labels associated with specified nodes and nodes associated with specified labels.
    - Admins should be able to remove a label entirely from a cluster.
        - Once removed, that label will be no longer associated with any nodes.
    - Security and access controls for managing Labels
        - Admins should be able to control which queues/users/groups can use which labels via ACLs
        - Only admins will be allowed to create new labels and add/remove the labels associated with a node.

    - Admins should be able to do all of the above ***dynamically*** at run-time without component restart.
- **Management**:
    - Admins shouldn't logically see the underlying storage formats of node-label configurations
    - Label definitions and configuration management should be scheduler-agnostic

Requirements from admin's point of view

- Must have
    - Admin can list/associate/un-associate labels for each node dynamically.
        - Need APIs such as,
            - CLI
            - REST APIs
    - Access Control Lists:
        - Admins can set Access Control Lists for restricting users/groups to use only a subset of the nodes corresponding to a subset of the labels.
        - Example: Only some of users/groups should be able to take advantage of nodes with labels GPU and/or FAT_MEMORY
- May be
    - Queue level labels (It's a separated section, see next)

Requirements on Node-labels configuration

- Decentralized configuration
    - Support per-node configuration (set in yarn-site.xml) instead of just the centralized configuration (as below) to support **legacy** decentralized way of configuration
        - Changing labels is very complex in the legacy way. Admins may be forced to manually change configs and restart nodes.
- Centralized configuration
    - Admins and end-users shouldn't ideally even know of the configuration files listing node-labels. The only endpoints should be CLI or REST.
        - RM can store it in whatever format it wants and wherever it wants
    - CLI/REST APIs
        - Set labels on a node(s)
            - Even take as input a specific format of a file
        - Add/remove labels on node(s)
        - Get labels on a node(s)
        - RM should have a list of node-labels at boot-up time. Two options
            - The CLI utility to set/modify labels has a mode which doesn't need RM to be running

            - We can pass it to the RM via yarn-site.xml when starting, similar to the present day config files
        - For a given label
            - Get the list of nodes marked with that label
            - Get/set ACLs for the label
- Admins have a choice of either doing a decentralized configuration or the centralized configuration - one cannot mix and match both.
- Restrictions
    - Label for each node should start with an alpha-numeric character. Character case will be ignored when comparing two labels. Letter, number, "-", "_" are only valid characters in a label name.
        - This is to avoid the pain of rendering by UIs (potentially also security holes and inconvenience on the web-UIs)
        - Should we limit max length of node labels?

Requirements from a user's point of view

- **User can specify dependency on node-labels via ResourceRequests**
    - *At application level*: During application-submission, user should be able to specify a fixed list of labels to apply for all containers of that application.
        - ResourceManager uses the app-level label requirements as requirement for all containers including the ApplicationMaster
    - *At container level*: Every time some requests are made by the AM, user should be able to set node-label requirements at some ResourceRequest levels as described below.
    - Force admins to use node-labels at queue-level (see end of this design doc).
- **Strength of the label-requirement**
    - Labels can logically be soft or hard requirements in general.
    - To simplify, to start with, we will only support **hard** requirement on node-labels. That is -- when a resource request asked for a node with label "foo", it will only be allocated to nodes with label "foo". [1], [2]
- **Request validity**
    - Labels are only allowed for rack & ANY (*) ResourceRequests
        - It doesn't make sense to have labels against Node specific ResourceRequests against nodes and result in InvalidResourceRequestException

    - ResourceRequest with a non-existent label (point in time) is illegal: results in InvalidResourceRequestException
    - ResourceRequest with a label without appropriate ACL results in IllegalResourceRequestException
    - A ResourceRequest will be rejected if any one of node-labels specified has #ref=0 (no node in the cluster contains the label at that point in time).

- **Number of labels per request**: Multiple node-labels can be specified as requirements at same time. To simplify the problem, now node-labels can only be connected by AND operation. [1]

- **Nodes' exclusivity with labels**: There are multiple possibilities for the exclusivity of labels:
    - **Exclusive labels**: Once a label is set against a node, that node is ***only*** usable by applications of users/groups who are part of that label's ACLs
        - In case, there are multiple labels against a node, then that node is **exclusive** to the union of the users/groups in all the ACLs.
    - **Exclusive labels with delayed scheduling:** When a node has a serials of labels (say x,y,z), but no application requested resources with such labels for a period of time (say threshold to allocate any resource on nodes) , the node can be scheduler arbitrarily.
    - **Exclusive labels through preemption:** Either (1) or (2) but those speculative containers will be preempted to give way to applications with node label ACLs.

Topics for discussion

Labels at queue-level

Some of the management of node-labels can potentially be simplified in certain sites by having admins specify labels against a queue. Some notes

- All applications submitted to a given queue will only run on nodes which have specific labels configured on the queue
    - This is a way of hard partitioning the cluster both in terms of the nodes and the queues
    - All the dimensions about exclusivity apply here.
- Admin can specify labels for scheduler queues, different schedulers pick their own way of configuration.
- Labels for queues can be updated dynamically

Respect node labels when preemption resources

Preemption logic in existing schedulers need to be aware of node-labels

Respect node-labels when calculate headroom

Calculation of per-application headroom needs to take node-labels into account.

Forward compatibility

- In case all the nodes in a cluster are assigned one or more labels, existing applications that didn't depend on labels may be starved because they don't set any label in ResourceRequest.
    - Maybe a configuration option to user: If no label set in ResourceRequest, should we automatically add a default label for such user/group/queue?

Support locality within a set of nodes selected via labels,

Applications may want to specify node-locality within the context of node-labels.

References

[1] Choosy: Max-Min Fair Sharing for Datacenter Jobs with Constraints, Ali Ghodsi et al.
[2] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das. Modeling and Synthesizing Task Placement Constraints in Google Compute Clusters. In ACM SoCC, 2011