# YARN-3409 : Add constraint node labels- Requirements & Design doc

**Authors** : Naganarasimha, Chong chen, Lei Guo & Varun Saxena with inputs from Wangda tan, Devaraj Kavali

**Last Modified Date** : Nov 18 2016



# What is Constraint Node Labels ?

As mentioned in YARN-2492, Node labels (including Partitions and Constraints) are non-tangible (consumable) resource associated with the node.

# How is it different from existing Node Labels ?

Existing Node Labels i.e. Partitions are used for logically partitioning the cluster, we can more like treat it as “Resource Pools” in traditional scheduling terms. As its partitioning the cluster based on these labels, a node can belong to only ONE partition label. Thus using existing labels we will not be able characterize a node based on many of its attribute and select a node based on expression of these attributes during scheduling.

# Usecases

**Hardware Constraints :** Not all resources can be isolated but we still need to support scheduling of tasks on the nodes based on the resources like GPU, FPGA, SSD, (dual) network cards, # of disks, InfiniBand etc. Further Tasks can depend on multiple of these resources hence cannot be supported by existing “Partition Labels” concept.

**Task Constraints :** In many scenarios, applications requires its containers to be run on nodes having specific Operating system versions, processor architecture, software library versions etc..

In many cases applications would require combination of multiple such attributes of the nodes. We will not be able to effectively acheive this only with Partitions, hence we require Constraint Labels.

# Requirements

## Top Level Requirements

- Bring in the notion of ConstraintNodeLabel (OS, Architecture, GPU ...) where in one node can have multiple Constraint labels, but can only belong to one partition (label). Or even a node not have any kind of labels also.
- One Constraint label can be associated with multiple nodes.
- Constraint Labels are not related to any Queue capacity/resource planning.
- We need to support varied type (boolean, long, String, Float, custom class) of Constraint Labels so that required type of comparison can be done. Custom Constraint Type can be used when we want to define custom comparisons, for example different softwares will have different formats to represent the software (library) version. So if its not based on ascii then user can define Custom Constraint Type which defines what all comparisions can be done and also how to evaluate the comparisons for the given software version.
- As we are supporting a type for a constraint label we need to have a unique list of Constraint labels at the RM side which can be treated as cluster Constraint labels and node will have subset of these labels
- Constraint Labels should be persistent across cluster components restarts and cluster upgrades, similar to the existing Partition labels.

## Mapping of Constraint Node-Labels to Nodes

Similar to existing configurations we can support the 3 approaches i.e.

**Central :** Node to Constraint labels mapping can be done through RM exposed CLI/RPC i.e. admin controlled.

**Distributed :** Node to Constraint labels mapping will be set by a configured Node Labels Provider in NM. Similar to existing support we need to support through scripts, which is very critical in dynamically identifying the attributes of the node.

**Delegated-Centralized :** Node to labels mapping will be set by a configured Node Labels Provider in RM. Similar to existing feature but not compulsion

And would require to support additional kind :

**Distributed-Constraints :** Similar to “Distributed” but only difference is we **only** get Constraint labels from the NM side. We might need this as Admin might want to specify the partition labels for the node and get the attributes of the nodes from script (in NM).

# Admin Requirements

- Similar to existing support for partitions, we should be able to do the following operations
  - → Should be able add new cluster level Constraint Labels along with specifying its types.
  - → Should be able to delete the Cluster level constraint label when not in use by any node.
  - → Should be able to map Nodes to constraint Labels ( in **Central mode**)
- Similar to existing support for partitions, we should be able to see the constraint labels using "yarn cluster" with additional options in CLI
- Similar to existing support for partitions, we should be able to fetch from REST api and as well as in the web UI in the nodes page (may be in separate table).
- There would NO accessibility settings for the queues or users for the Constraint labels. (another difference with Partitions).
- Constraint label Naming restrictions : Same naming convention as Partitions i.e. Label for each node should start with an alphanumeric character. Character case will be ignored when comparing two labels. Letter, number, "", "_" are only valid characters in a label name.

# YARN User Requirements

## ResourceRequests modifications

- This requires some discussion but here will just specify the best option i can think but will discuss others in later sections. Each container Request (including AM) should be able to specify the constraintLabelExpression.

```
ResourceRequest {
        Priority priority,
        String hostName,
        Resource capability,
        int numContainers,
        boolean relaxLocality,
        String labelExpression,
        String constraintLabelExpression, // New modification in the interface
        ExecutionTypeRequest executionTypeRequest
}
```

  Main advantage of this approach being that its segregates itself from existing "partitions" and hence less confusion for user to specify the complex constraint expression pattern.

- App Level constraint expression similar to partitions label expression, to be supported for constraint labels too i.e. expression mentioned as part of ApplicationSubmissionContext will be considered for containers of the apps unless overidden by the individual ResourceRequest.

- Similar to partitions admins should be able to enforce constraint label expression at queue - level.

## Constraint Label Expression support

- Default Types which are planned to be supported : Boolean, Double, Long, String.

  ex. ***Boolean Constraints*** **:** HAS_GPU, Windows, Linux, Mac, *HAS_SSD*

  ***String Constraints :*** *ARCHITECTURE, JDK_TYPE*

  ***Long Constraints :*** *NUM_OF_DISKS, NUM_OF_INTF, MAX_MEM*

  ***Double Constraints :*** *GLIBC (version), JDK_VERSION*

- Comparison Operators within an expression :

  → Exists based Operators : !<label>, <label>

  - *HAS_GPU* => run processing intensive tasks in the nodes which has GPU's
  - *!Windows* => run container on nodes which are not having "Windows" OS usually required when we need to run binary compatable to specific OS

  → Equality-based : ==, !=

  - *ARCHITECTURE* == X86_64 => to run a task on the nodes which supports 64-bit computing
  - *JDK_VERSION != 1.6* => run container on nodes which has any JDK apart from 1.6 version, to avoid particular version of software.

  → Numerical comparisons : >,>=,<,<=

  - *NUM_OF_DISKS >= 3* =>run container on nodes which has more local disks. *Spark kind of tasks doing a lot of data processing would like to have more disks to process the data parallelly*
  - *NUM_OF_INTF >= 2* => run container on nodes which has dual networks *(generally required for AM's to be run in dual plane so that its accessible for clients)*
  - *MAX_MEM > 64 => run container on nodes which has more memory (so that there will be chances for resizing the container even though the current allocation is Less , suitable for spark )*

  → Set-Based : IN, NOT_IN

  - JDK_TYPE IN (OPEN_JDK, ORACLE_JDK) && GLIBC NOT_IN ( 1.2, 2.0)

- Multiple constraint expressions will be supported with each of them combined with '&&' or '||'
- '(' and ')' are supported to segregate the expressions and define the order. Expressions are evaluated from left to right and inner most expression first.
  - (*NUM_OF_DISKS >= 3 || HAS_SSD)* && !Windows

- In all the expression, if the constraint label for a node doesn't exist then it will evaluated to false. Except for '!' and NOT_IN
- If a given ConstraintType doesnt support a particular comparison operator, initial evaluation of the expression fails and expression will be thrown.

# Topics for discussion

## ResourceRequest modifications:

Its seems to be initially complicated for users to understand the concept of what are Partitions and Constraints and supported expression operators, so we feel it's ideal to separate them so that there is less confusion while forming expression. However disadvantage of it is :

- We almost need to add ConstraintLabelExpression each place where nodeLabelExpression is currently supported, like ResourceRequest, CS' QueueDefaultNodeLabelExpression, ApplicationNodeLabelExpression.
- Applications like MR are already exposing 4 partition node label expression configurations now if we support separate config for Constraints then it would sum to almost 8.

Alternate approaches.

- **Option 2**: Support a delimiter in existing NodeLabelExpression (say ';' delimiter between partition label expression & Constraint label expression) to identify them separatly.

  *<partition expression>;<Constraint Expression>*. Similar to acls just that the delimiter is different
- **Option 3**: Parse it similar to ConstrainLabelExpression with special handling explicitly for Partitions (like a Custom Constraint Type with a reserved label for Partition).

IMO i would first choose Option 1 (i.e. new field for Constraint label expression ), because other approaches seems to be more of an alternate approach than a concrete solution. Further initial intution might be for option 3 but if user specify like the expression below then its not possible to satisfy with any resource and it would not be easily feasible to detect and inform that the expression will not be satisfied. Hence option 3 doesn't fit in.

((<Partition1> || HAS_SSD) && ((<Partition2> || HAS_GPU) ),

Though the expression is valid no containers will be assigned as given node can never belong to both the Partitions.

## Separate configuration to enable disable constraint labels

I think existing configuration to enable labels should be sufficient.

# References :

Many of the existing distributed systems already support different variety of labels/ attributes of the node and expressions to support evaluation

**LSF** : http://www.ibm.com/support/knowledgecenter/SSETD4_9.1.3/lsf_admin/select_string.html

**MESOS** : http://mesos.apache.org/documentation/latest/attributes-resources/

though they might be exactly the same as of our solution but definitely we need support multiple kinds of attributes of the nodes.