Label Based Scheduling

Yuliya Feldman, Swapnil Daingade

Introduction

This is a design to follow up on JIRA: https://issues.apache.org/jira/browse/YARN-796
As some of the comments suggest that there is an interest in providing labels not just to mark nodes but also to schedule YARN applications on particular nodes.

Goal

1. Provide a robust mechanism to run application containers on particular set of nodes
2. Provide a robust mechanism for Scheduler's Queues to schedule resources only on particular set of nodes
3. Provide mechanism to resolve conflicts between Queue and Application defined set of nodes to run containers on

Definitions

- **Node Label** - label that describes a node. Node can have multiple labels
- **Label expression** - logical combination of labels (using && - and, || - or, ! - not)
- **Queue Label** - Label expression that determines on which nodes applications that belong to that queue may run
- **Application Label** - Label expression that determines on which nodes application wants it's containers to run
- **Queue Label Policy** - Label Policy configured on the Queue that defines rules for resolving conflicts between application label and queue label

Use case

Cluster A uses Fair Scheduler to schedule applications on it's nodes.
Cluster Admin defined multiple queues including nested ones to divide resources among different organizations, so each org has it's share of resources to use.
In addition cluster A is heterogeneous in different aspects:

- it has different types of hardware (some nodes have more memory, some more CPU, some better network bandwidth)
- data in DFS broken into topologies that are tied to particular set of nodes/racks
- other differences

Cluster admin label all the nodes in the cluster according to their differences:

- nodes with large memory - "red"
- nodes with high CPU - "blue"
- nodes with high network bandwidth - "green"
- nodes on rackA - "rackA"
- nodes on rackB - "rackB"

Since "red" nodes can be on rackA, or rackB cluster admin can assign multiple label to them:

Node Regex	**Label1**	**Label2**	**...**	**LabelN**
rackA_red	red	rackA		
rackA_blue	blue	rackA		
rackA_green	green	rackA		
rackB_red	red	rackB		
rackB_blue	blue	rackB		

If we want all the applications submitted to Queue 1 run on “rackA_red” or “rackA_blue” nodes Cluster Admin will just define label expression while configuring Queue 1 as: “(red || blue) && rackA”.
In addition Cluster Admin configures a Queue Policy that defines scheduling behavior in case submitted application defines it’s own label expression. Admin can define that:

- Queue Label Expression always wins
- Application Label Expression always wins
- There is an AND condition between the above
- There is an OR condition between the above

Now User B submits an application to Queue 1, but he really cares about CPU, as application is CPU intensive and he wants it to run on high CPU nodes only. So he applies “blue && rackA” Label Expression on the Application.
Luckily cluster admin sets up a Queue Policy as “AND”, so resulting set of the nodes where Application will be scheduled is: “rackA_blue” since result of evaluation of Queue and Application logical expressions is “blue && rackA”
In case cluster admin sets up a Queue Policy as “OR” resulting label expression is going to be: “((red || blue) && rackA) || (blue && rackA)” which will result in “(red || blue) && rackA”

High Level Design considerations

Currently there is a mechanism to prevent containers to be run on certain nodes by blacklisting them by the application (list of blacklisted nodes can be supplied by AppMaster to ResourceManager)

We plan to piggyback on this “blacklisting” mechanism to “blacklist” nodes not only if they are explicitly provided in the ResourceRequest but also based on Queue Labels, Application Labels and rules to apply them

Design

Node Labels, Label Expressions, Policies

Node Labels

Original JIRA suggests each node to specify it’s own labels and provide this info to ResourceManager.

We suggest centralized location for node labels such as file stored on DFS that all the YARN daemons have access to and it does not require to be synchronized across whole cluster and/or clients

Labels can be specified in following format:

perfnode. big, fast,physical*
scale. virtual , “slow”*
perfnode30 ‘bad’

Where

the first field is regular expression for the node(s) - it would be time consuming to specify each and every node here
the following fields are comma separated and represent set of labels that are applied to the node/group of nodes

Label Expressions, Policies

Label expression - is a logical expression that can consist of one or more node labels (example: big || virtual && !bad).

Application Label expression defines set of nodes on which application “wants” to run.
Queue Label expression defines set of nodes on which queue “wants” all the applications that are scheduled on that queue to run on.

Some of those Label Expressions could be contradictory especially while taking into consideration both Application and Queue Label Expressions.
In order to resolve those conflicts we introduce “Queue Label Policy”.

Queue Label Policy

Every queue that is configured to deal with label expressions can define a Label Policy to interact with Application Label Expression

Following are proposed values for the policy:

```
PREFER_QUEUE("PREFER_QUEUE"), // Queue label expression always wins
PREFER_APP("PREFER_APP"), // App label expression always wins
AND("AND"),  // Use && on Queue and App label expressions  ← default
OR("OR");  // Use || on Queue and App label expressions
```

This way we can resolve conflicts between Application and Queue requirements to schedule on particular set of nodes

Label Manager

Label Manager is a new service that will be introduced to run as part of ResourceManager (RM).

Responsibilities of this service will be:

1. Loading node labels from DFS and maintain internal map of node to list of labels for that node
2. Update internal node to labels mapping based on changes to a file - will reupload file with labels every “node.labels.monitor.interval” (default to 2 mins.) if last time file was modified is greater than last time it was uploaded
3. Construct ready to evaluate label logical expression (for both queue and application)
4. Construct combined logical expression based on Application Label Expression, Queue Label Expression and Queue Label Policy
5. Evaluate logical expression against the node

Following product is proposed to be used to evaluate expressions:
http://java.net/projects/eval/pages/Home (Eval - A Simple Expression Evaluator for Java)

Admin APIs

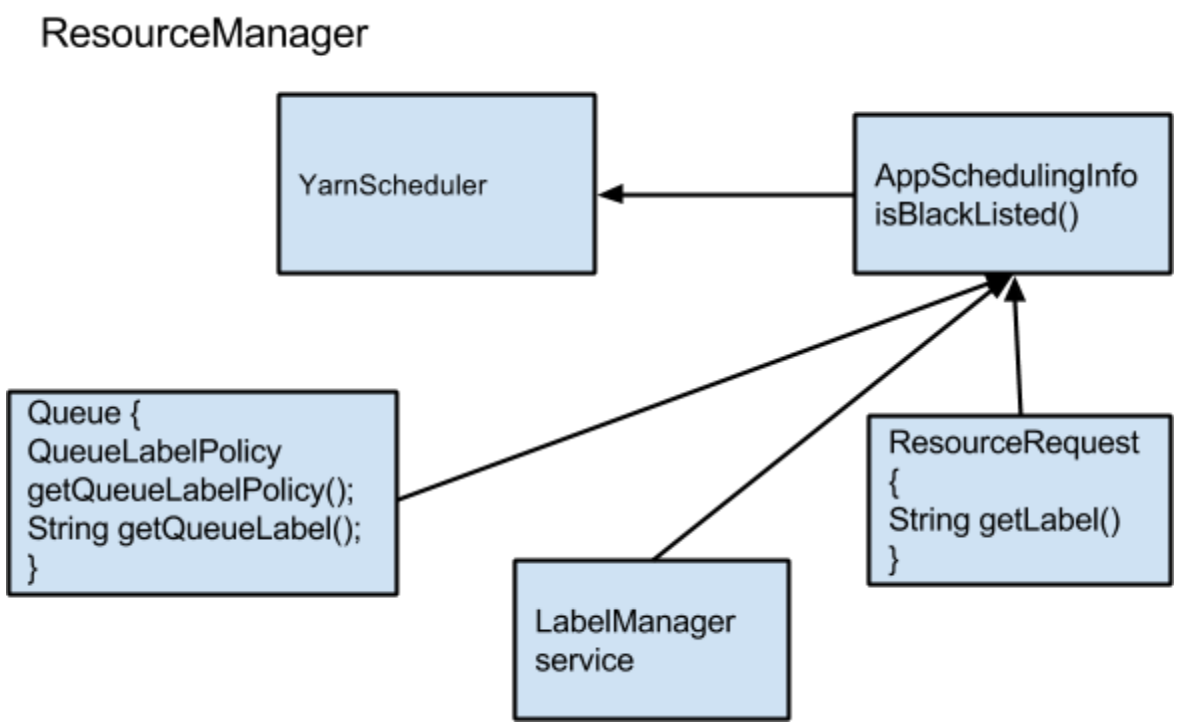
New admin APIs will be added to view Node labels and immediately refresh Node labels:

yarn rmadmin -showlabels
yarn rmadmin -refreshlabels

To continue supporting jobs specific APIs, the following will do the same:

mapred job -showlabels
mapred job -refreshlabels

Flow interactions



AppSchedulingInfo class has all the information about ResourceRequests and Queue for the application so it can make intelligent decision whether to blacklist a node based on combination of AppicationLabelExpression, QueueLabelExpression and QueueLabelPolicy

High level changes to existing classes and interfaces

1. ResourceRequest and it's proto gets new field: "String label" - that represents application label expression
2. ApplicationSubmissionContext also gets new field "String label" - that represents application label expression, so any application while submitting to ResourceManager can setLabel() while creating ApplicationSubmisisonContext
3. YarnRunner sets label on ApplicationSubmissionContext for MapReduce applications
4. ResourceManager manages new service: LabelManager (init, start, stop)
5. Queue interface has new enum: QueueLabelPolicy
6. Queue interface is updated with two new methods:
    a. String getLabel() - to get Queue Label Expression
    b. QueueLabelPolicy getLabelPolicy() - to get QueueLabelPolicy
    c. All implementations of Queue interface will implement those methods

7. All schedulers classes include refresh of Queue Labels and QueueLabelPolicy
8. AppSchedulingInfo class method “isBlacklisted(String resource)” is updated to decide whether this resource has to be blacklisted based on combination of AppicationLabelExpression, QueueLabelExpression and QueueLabelPolicy

LabelManager service configuration

LabelManager needs to configure following properties in yarn-site.xml to support label based scheduling
1. Required - if using label based scheduling

```
<property>
  <name>node.labels.file</name>
  <value>/labels.file</value>
  <description> Location of the file that contain node labels on DFS
  </description>
 </property>
```

2. Optional (default to 2 min.)

```
 <property>
  <name>node.labels.monitor.interval </name>
  <value>120000</value>
  <description> Interval to check labels file for updates
  </description>
 </property>
```

MapReduce Job submission configuration

To configure Application Label Expression for MapReduce Job(s) following property needs to be set either in mapred-site.xml or on command line:

```
<property>
  <name>mapreduce.job.label </name>
  <value>good || fast</value>
  <description> Label expression for Map/Reduce job
  </description>
</property>
```

Example of wordcount job

hadoop jar hadoop-mapreduce-examples*.jar wordcount
-Dmapreduce.job.queuename=Marketing.CustomerDataAnalysis.FastLanes
-D**mapreduce.job.label="good || fast"** input output

Schedulers Queues Label and Label Policies Configuration

Capacity Scheduler

Capacity Scheduler configuration is located in capacity-scheduler.xml

Following are new properties that can be defined for a particular queue to support Label Expression and/or Label Policy (Example is given to set Label Expression and Queue Label Policy for Queue "alpha")

```
<property>
  <name>yarn.scheduler.capacity.root.alpha.label-policy</name>
  <value>PREFER_QUEUE</value>
    <description>
     The ACL of who can submit jobs to the default queue.
    </description>
  </property>

<property>
  <name>yarn.scheduler.capacity.root.alpha.label</name>
  <value>fast||big</value>
    <description>
     The ACL of who can submit jobs to the default queue.
    </description>
  </property>
```

In case Queue "alpha" has children queues that do not have labels set on them label and label policy from parent queue will apply.
In case children queues have their own labels and/or label policies, they will be used

Below is an example for "alpha.a1" Label configuration

```
property>
  <name>yarn.scheduler.capacity.root.alpha.a1.label-policy</name>
  <value>PREFER_APP</value>
    <description>
```

```
     The ACL of who can submit jobs to the default queue.
    </description>
  </property>

<property>
  <name>yarn.scheduler.capacity.root.alpha.a1.label</name>
  <value>slow</value>
    <description>
     The ACL of who can submit jobs to the default queue.
    </description>
</property>
```

Fair Share Scheduler

Fair Share Scheduler can be configured in any xml file as long as file name is defined in yarn-site.xml
To configure label expressions and/or label policies additional properties - label, labelPolicy - are used. Below is an example of a Queues configuration in FairScheduler where label and labelPolicy can be defined on any level - the closest will be used.

```
<queue name="Marketing">
   <minShare>8192</minShare>

   <queue name="WebsiteLogsETL">
     <weight>1.0</weight>
   </queue>

   <queue name="CustomerDataAnalysis">
     <weight>2.0</weight>
      <labelPolicy>OR</labelPolicy>
      <label>good</label>
     <queue name="FastLanes">
      <weight>3.0</weight>
      <maxShare>6096</maxShare>
      <labelPolicy>PREFER_QUEUE</labelPolicy>
      <label>abc</label>
     </queue>
     <queue name="Regular">
     </queue>
   </queue>

  </queue>
```

Note: While making changes to capacity-scheduler.xml or fair share scheduler configuration xml file, use xml encoding for following special characters.
“ - "
& - &
‘ - '
< - <
> - >

Considerations

I. Any YARN Application can use labels as long as Application Label Expression is set on ApplicationSubmissionContext.
   A. YARNRunner will be updated to use it by default - meaning M/R use case will be taken care of, but any other YARN application in order to take advantage of the labels will have to set application label on the ApplicationSubmissionContext based on their configuration property name or by other means.

II. There are performance considerations to be taken into account
   A. Since “isBlacklisted(String resource)” method is used very often reevaluation of the label expressions on every call may have performance implications. There are following mitigations that can be considered
      1. ResourceRequest label expression - can not be changed while application is running, so may not need to be reconstructed again
      2. Queue Label Expression, Queue Label Policy as well as Node labels do not change frequently. If we assume that during life of the application none of those changes can take effect on the application, we can set a node as blacklisted on a first evaluation so that node won’t be reevaluated again

III. FairScheduler preemption and LabelBased Scheduling coordination should be taken into consideration - not yet

IV. When invalid label expression (consists of label(s) that are not present in the labels file) is used to define for Queue or Application it will be ignored as if no label was set. RM logs will have errors about usage of invalid labels

V. If no node that satisfies final label evaluation is available Application will be waiting to be submitted.