YARN-3409 : API changes

By Naganarasimha G R, with inputs from Sunil & Wangda.

Last Modified Date : Jul 24 2017

Mapping Attributes on Node

Though we need to maintain a superset of Node Attribute Labels to ensure attribute name - type mapping is preserved unique, we plan to maintain it internally and not to expose that to user side.

Users should be able to map node to attribute labels and if there is any type mismatch then the operation would fail.

1. CLI API interface changes :

yarn node–attributes [-update|-replace|-remove]
<"node1=attribute1[:attributeVal[:attributeType]],label2[:value]
node2=label1[:value],label2[:value]"> [-failOnUnknownNodes]

1. *Replace all attributes mapped to a node with new attributes mapping*
2. *Update or add specific attribute mapping(with val) to the existing map of attributes*
3. *Remove a single attribute mapped to a node(s)*

Note :

- *On "update" attributeType will not be considered to be modified if there is already mapping exist in RM.*
- *On "update and remove" attributes needs to be present for a node.*
- *Hostname or the IP (based on how the nodes are configured in RM) is sufficient, port mapping is not required.*

Open points

Do we need to support delete of a given attribute from all nodes. (as we do not support add and remove a given attribute). If so would it be ok to accept it as part of remove without specifying a nodeId?

2. API proto changes :

```
enum NodeAttributeTypeProto {
    STRING = 1;
 }
message NodeAttributeProto {
        optional string attributeName = 1;
        optional NodeAttributeTypeProto attributeType = 2;
        optional String attributeValue = 3;
}
```

Introduce a new method to be supported for Node attribute Labels in admin Service

```
service ResourceManagerAdministrationProtocolService {
…
 rpc mapAttributesToNodes(NodesToAttributesMappingRequestProto) returns
(NodesToAttributesMappingResponseProto);
}
message NodesToAttributesMappingRequestProto {
        optional string operationName = 1 [default= replace];
        repeated NodeIdToAttributesProto nodeToAttributes = 2;
        optional bool failOnUnknownNodes = 3;
}
message NodeIdToAttributesProto {
  optional NodeIdProto nodeId = 1;
  repeated NodeAttributeProto nodeAttributes = 2;
  }
```

This seems to be suitable for storing as well as compatibility wise hence we choose among other alternatives which we could think.

Alternate Proposal 1:

we can have one more field which says the given label partition or attribute

```
enum NodeLabelTypeProto {
        Partition, Attribute
}
message NodeLabelProto {
        optional string name = 1;
        optional bool isExclusive = 2 [default = true];
        optional NodeAttributeTypeProto attributeType = 3;
        optional String attributeVael =4;
        optional NodLabelTypeProto labelType = 5 [default= Partition];
}
```

Alternate Proposal 2:

```
message NodeLabelProto {
        optional string name = 1;
        optional bool isExclusive = 2 [default = true];
        optional NodeAttributeTypeProto attributeType = 3;
        optional String attributeVal = 3;
}
```

Based on attributeType == null determine whether its partition or Attribute

3. REST api interface changes :

i. Support Replace of node to attributes mapping

@PUT

http://<rm http address:port>/ws/v1/cluster/`replace-attributes-on-nodes`

```
NodesToAttributesMapping {
        List<NodeToAttributesEntry> nodeToAttributeLabels
}
NodeToAttributeLabelsEntry {
        Nodeid nodeId
```

ArrayList<NodeAttributeInfo> nodeAttributes;

}

NodeAttributeInfo{

String attributeName;

AttributeType attributeType;

String attributeValue;

}

ii. Support add or update of node to attributes mapping

@POST

http://<rm http address:port>/ws/v1/cluster/update-attributes-on-nodes

DOM sent across is same as "replace-attributes-on-nodes"..

iii. Support remove of node to attributes mapping

@DELETE

http://<rm http address:port>/ws/v1/cluster/remove-attributes-on-node

DOM sent across is same as "replace-attributes-on-nodes".