Rack awareness in Hadoop

This document displays how to configure the rack awareness in a Hadoop cluster. This requires 1 Namenode and 3 datanodes in the cluster. This will help you understand the rackawareness in a better way.

In Hadoop, Both HDFS and YARN are rack-aware. meaning if you inform them as to where the cluster nodes are located relative to each other, then both HDFS and YARN can make use of this information to make the Hadoop environment better.

This is important as a Rack mostly has a single switch. If this switch fails then all the servers connected to it become unavailable. Also the rack may have a single power supply for all equipments within the rack. Thus failure of this power supply also makes nodes unavailable.

HDFS uses rack awareness for fault-tolerance purposes. It ensures that it places one block replica on a different rack. Thus, if a network switch fails and an entire rack goes down, you are still guaranteed access to the data.

The ResourceManager capitalizes on its rack awareness to optimally allocate resources to clients by steering them to the nodes that are closest to the data. The NameNode and the ResourceManager daemons obtain the rack information by invoking an API, which also resolves DNS names to a rack ID.

Hadoop with the default replication factor of 3 , places data blocks in only two unique racks rather than three different racks.

,By default, even if you place nodes to multiple racks in your cluster, Hadoop assumes that all nodes belong to the same rack.

Check the current rack information using following command

hdfs dfsadmin -printTopology

Also create a file in HDFS. For this first create a file on your Linux system.

nano ~/test.txt

Type some text in the file. Save the file. Then copy this file to HDFS using following command.

hdfs dfs -mkdir /demo

hdfs dfs -put ~/test.txt /demo

Now check on which nodes the file is copied. For this use following command

hdfs fsck /demo/test.txt -files -blocks -racks -locations

Configure Rack awareness in Hadoop

1. Go to /usr/local/hadoop/etc/hadoop directory.

2. Create a file by name **rack-info.py**.

3. Type following python script in the file.

```
#!/usr/bin/env python
import sys
DEFAULT_RACK = "/default/default"
HOST_RACK_FILE = "/usr/local/hadoop/etc/hadoop/rack-map.txt"
host_rack = {}
for line in open(HOST_RACK_FILE):
        (host, rack) = line.split()
        host_rack[host] = rack
for host in sys.argv[1:]:
        if host in host_rack:
                print host_rack[host]
        else:
                print DEFAULT_RACK
```

Save the file.

4. Provide execute permission to the above file.

chmod u + x rack-info.py

5. Now create the **rack-map.txt** file mentioned in the above script. This file holds the mapping of nodes and the rack.

Add following in the file.

```
192.168.208.170 /default/rack0
192.168.208.171 /default/rack0
192.168.208.172 /default/rack1
DN1 /default/rack0
DN2 /default/rack0
DN3 /default/rack1
```

Save the file.

(*** Note:- Make sure you type your node IP address and hostnames)

6. Now configure the property in core-site.xml so that Hadoop will execute the script and get the rack information about the nodes.

<property>
<name>net.topology.script.file.name</name>
<value>/usr/local/hadoop/etc/hadoop/topology.py</value>
</property>

Save the file.

7. Now stop the DFS service.

stop-dfs.sh

8. Then start the DFS service.

start-dfs.sh

9. Now check the current rack information using following command

hdfs dfsadmin -printTopology

Also create a file in HDFS. For this first create a file on your Linux system.

nano ~/newtest.txt

Type some text in the file. Save the file. Then copy this file to HDFS using following command.

hdfs dfs -mkdir /demo
hdfs dfs -put ~/newtest.txt /demo

Now check on which nodes the file is copied. For this use following command

hdfs fsck /demo/newtest.txt -files -blocks -racks -locations

Did you find any difference in the output of the above command.
Try creating some more files.