Hadoop TP

Installing the environment:

To install with docker the Hadoop image with Python 2.6 run the following command:

```
docker run -it --name myhadoop sequenceiq/hadoop-docker:2.7.0 /etc/bootstrap.sh -bash
```

If already created and running:

```
docker exec -it myhadoop /bin/bash
```

Setting Hadoop in pseudo-distributed mode

1. The Hadoop executables are located in directory $HADOOP_PREFIX. Check where this directory is located and make it the current directory.

   `echo $HADOOP_PREFIX` (-> /usr/local/hadoop) (-> echo tells you the content of an environment variable in Hadoop)

   `cd /usr/local/hadoop`

2. With a quick look at configuration files: etc/hadoop/core-site.xml and etc/hadoop/hdfs-site.xml

   • check that we are using HDFS, and not the local filesystem as in standalone mode.

   • check the default replication factor.

   Within /usr/local/hadoop, you run the following command:

   `cat etc/hadoop/core-site.xml` (-> tells you the content of a file in Hadoop)

   The content is:

   <configuration><property><name>

   dfs.replication (-> default replication factor = 1)

   </name><value>1</value>

   </property></configuration>

   `cat etc/hadoop/hdfs-site.xml`

   The content is:

   <configuration>

   <property>

<name>fs.defaultFS</name> (-> default file system = HDFS)

<value>hdfs://8d6961b2aa71:9000</value>

</property>

</configuration>

3. Then check which Java processes are running with jps. You should see: Jps, DataNode, NameNode, SecondaryNameNode, NodeManager, ResourceManager.

```
bash-4.1# pwd
/usr/local/hadoop
bash-4.1# cat etc/hadoop/hdfs-site.xml
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
bash-4.1# cat etc/hadoop/core-site.xml
  <configuration>
      <property>
          <name>fs.defaultFS</name>
          <value>hdfs://8d6961b2aa71:9000</value>
      </property>
  </configuration>
bash-4.1# jps
1083 Jps
575 ResourceManager
131 NameNode
406 SecondaryNameNode
226 DataNode
672 NodeManager
bash-4.1#
```

Managing files with HDFS

1. Move the text file words.txt from your computer to the container. docker cp words.txt myhadoop:/.

Run this command within the Windows Command Line, not within WSL.

docker cp "C:\Users\pmoll\Desktop\Master\Semester 3\Distributed Systems for Massive Data Management\TPs\TP_Hadoop\tp-hadoop-en\tp-hadoop-en\words.txt" myhadoop:

```
bash-4.1# ls
bin   dev   home   lib64   mnt   proc   sbin      srv   tmp   var
boot  etc   lib    media   opt   root   selinux   sys   usr   words.txt
bash-4.1# pwd
/
bash-4.1#
```

2. Create a (distributed) directory in hdfs bin/hadoop fs -mkdir /rep.

Run the same code(within the /usr/local/hadoop), I don't know how to check where it is.

bin/hadoop fs -mkdir /rep or bin/hdfs dfs -mkdir /rep

3. Copy the file from the local file system to hdfs:

bin/hadoop fs -put /words.txt /rep/words.txt. (run the code in /usr/local/hadoop)

To check if it was created the /rep directory and now contains words.txt:

bin/hadoop fs -ls /rep

Hadoop Streaming: running a MapReduce job defined with executables

- JAR File: The job uses the hadoop-streaming-2.7.0.jar file. Hadoop Streaming allows you to create and run MapReduce jobs with any executable or script as the mapper and reducer. It's a utility that comes with Hadoop.

- Input Directory: The input data for the MapReduce job is taken from the /rep directory in HDFS. This directory should contain your input data files.

- Output Directory: The output of the MapReduce job will be written to outputdir0 in HDFS. If this directory already exists, the job will fail, as Hadoop does not overwrite existing directories or files. You need to ensure that outputdir0 does not exist before running the command.

- Input Format: It specifies org.apache.hadoop.mapred.KeyValueTextInputFormat as the input format, which treats each line of input as a key-value pair separated by a tab. This is useful for processing text files where each line has a key and a value.

- Mapper: The mapper is set to org.apache.hadoop.mapred.lib.IdentityMapper, which is a built-in Hadoop function that simply outputs each input key-value pair as output without any changes. Essentially, it's a pass-through operation.

- Reducer: The reducer is specified as an external command /usr/bin/wc, which is a Unix/Linux command that counts lines, words, and bytes in its input (from the mapper in this case). This setup uses Hadoop Streaming's capability to use non-Java programs as reducers.

- So, what this MapReduce job does is:

1. It reads key-value pairs from text files in the /rep directory on HDFS.
2. Passes these key-value pairs directly through the IdentityMapper, making no changes to the input data.
3. Then, it pipes the output of the mapper to the Unix/Linux wc command, which counts the number of lines, words, and bytes in the output from the mapper.
4. Finally, it writes the output of the wc command, which is the count of lines, words, and bytes, to the outputdir0 directory on HDFS.