Hadoop and Harp installation

Guide

This instruction is only tested on linux and mac OS. If you are using windows, we suggest you to install an Ubuntu system on a virtualization software (eg. VirtualBox) with at least 4GB memory in it.

Hadoop

Hadoop is an open source framework for distributed storage and processing of large datasets on a commodity cluster. Hadoop utilizes the Hadoop Distributed File System (HDFS) for data storage and the MapReduce model for computational processing. Hadoop 2.6 also includes the YARN resource management platform which manages multiple nodes within the cluster and is responsible for task scheduling.

Preparation

1. Create Directories

We'll assume the following directory structure, where the **software** directory holds unzipped software, and **zips** contains the software tarballs.

```
/home/ubuntu
  ├── software
  ├── zips
```

If you don't see this structure, you can create it using the following command.

```
mkdir ~/software
mkdir ~/zips
```

2. Vim

If you have vim installed, skip this step. Else, install vim, the command-line text editor using the following command.

```
sudo apt-get install vim
```

3. SSH and Rsync

Test with the following command. If it works (you will need to input the password), skip this step.

```
ssh localhost
```

If SSH and Rsync are not already installed in the environment, use the following commands to install

them.

```
sudo apt-get install ssh
sudo apt-get install rsync
```

4. Java

If it's installed already, skip this step. Or, download Oracle JDK 8 and extract the archive using the following steps.

```
cd ~/software
wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie"
http://download.oracle.com/otn-pub/java/jdk/8u91-b14/jdk-8u91-linux-x64.tar.gz

tar xzf jdk-8u91-linux-x64.tar.gz
mv jdk-8u91-linux-x64.tar.gz ~/zips
```

5. Hadoop

Download and extract the latest Hadoop binary into your machine. These are available at http://hadoop.apache.org/releases.html. The following commands will download and extract Hadoop version 2.6.0.

```
cd ~/software
wget http://www-eu.apache.org/dist/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
tar -xzvf hadoop-2.6.0.tar.gz
mv hadoop-2.6.0.tar.gz ~/zips
```

6. Environment Variables

Set the following environment variables (you can set the variables at the **top** of the ~/.bashrc file). You can use the following command to open and edit the ~/.bashrc file.

```
vim ~/.bashrc
```

Add the the following lines to the beginning of the file.

```
export JAVA_HOME=~/software/jdk1.8.0_91
export HADOOP_HOME=~/software/hadoop-2.6.0
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$HADOOP_HOME/bin:$JAVA_HOME/bin:$PATH
```

Now run the following command in order to make sure the changes are applied.

```
source ~/.bashrc
java -version
```

You should see an output similar to the one given below.

```
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

8. Verify Hadoop

Check if you can successfully run the following Apache Hadoop command.

```
hadoop
```

You should see the following output.

```
Usage: hadoop [--config confdir] COMMAND
       where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
  checknative [-a|-h]  check native hadoop and compression libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
  classpath            prints the class path needed to get the
  credential           interact with credential providers
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  trace                view and modify Hadoop tracing settings
or
  CLASSNAME            run the class named CLASSNAME

Most commands print help when invoked w/o parameters.
```

Set up Password-less SSH

Test if you can SSH to **localhost** without requiring a password.

```
ssh localhost
```

If the above requires a password then setup password-less SSH using the following commands.

```
ssh-keygen -t rsa
(hit enter to all the options)

cd ~/.ssh
```

```
cat id_rsa.pub >> authorized_keys
```

Then try to SSH again and if that's successful hit exit to terminate that SSH connection

```
ssh localhost
exit
```

Hadoop Configuration

Modify the following files in Apache Hadoop distribution.

1. core-site.xml

```
vim   $HADOOP_HOME/etc/hadoop/core-site.xml
```

```
<configuration>
      <property>
            <name>fs.default.name</name>
            <value>hdfs://localhost:9010</value>
       </property>

      <property>
            <name>hadoop.tmp.dir</name>
            <value>/tmp/hadoop-${user.name}</value>
            <description>A base for other temporary directories.</description>
      </property>
</configuration>
```

2. hdfs-site.xml

```
vim $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

```
<configuration>

      <property>
            <name>dfs.replication</name>
            <value>1</value>
      </property>
      <property>
            <name>dfs.namenode.http-address</name>
            <value>localhost:50070</value>
      </property>
      <property>
            <name>dfs.namenode.secondary.http-address</name>
            <value>localhost:50190</value>
      </property>
```

```
</configuration>
```

3. mapred-site.xml

```
vim $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

    <!-- set application master memory size -->
    <property>
        <name>yarn.app.mapreduce.am.resource.mb</name>
        <value>512</value>
    </property>
    <property>
        <name>yarn.app.mapreduce.am.command-opts</name>
        <value>-Xmx256m -Xms256m</value>
    </property>
<configuration>
```

4. yarn-site.xml

```
vim $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>localhost</value>
    </property>
    <property>
        <name>yarn.resourcemanager.address</name>
        <value>localhost:8132</value>
    </property>
    <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>localhost:8130</value>
    </property>
    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>localhost:8131</value>
    </property>
    <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>localhost:8133</value>
    </property>
```

```
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>localhost:8080</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>

    <!-- check the memory size under the node manager -->
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>4096</value>
    </property>

   <!-- no memory checking -->
   <property>
        <description>Whether virtual memory limits will be enforced for containers.</description>
        <name>yarn.nodemanager.vmem-check-enabled</name>
        <value>false</value>
    </property>

   <!-- set the min and max container allocation -->
   <property>
        <name>yarn.scheduler.minimum-allocation-mb</name>
        <value>512</value>
   </property>
   <property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
        <value>2048</value>
   </property>
</configuration>
```

Start Daemons

1. Format the file system next.

```
hdfs namenode -format
```

If you can see information like below, the format process should be successful.

```
xx/xx/xx xx:xx:xx INFO util.ExitUtil: Exiting with status 0
xx/xx/xx xx:xx:xx INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at xxx.xxx.xxx.xxx
```

2. Launch NameNode daemon and DataNode daemon

```
$HADOOP_HOME/sbin/start-dfs.sh
```

The log is in the $HADOOP_LOG_DIR directory (default is $HADOOP_HOME/logs). You can always check with logs to debug if any error appears. One of the possible errors is port conflict. So be careful when you assign ports in configurations above.

3. Check if the daemons started successfully.

```
jps
```

You should see the following with xxxxx replaced to actual process IDs.

```
xxxxx NameNode
xxxxx SecondaryNameNode
xxxxx DataNode
xxxxx Jps
```

You can browse the web interface for the NameNode at http://localhost:50070

4. Start ResourceManager daemon and NodeManager Daemon

```
$HADOOP_HOME/sbin/start-yarn.sh
```

5. Verify the daemons started successfully

```
jps
```

You should see the following with xxxxx replaced by actual process IDs.

```
xxxxx NameNode
xxxxx SecondaryNameNode
xxxxx DataNode
xxxxx NodeManager
xxxxx Jps
xxxxx ResourceManager
```

You can browse the web interface for the ResourceManager at http://localhost:8088

Example

1. Make the Hadoop Distributed File System (HDFS) directories.

```
hdfs dfs -mkdir -p .
hdfs dfs -mkdir input
```

2. Copy the input files into HDFS.

In this example, we use files in $HADOOP_ HOME/etc/hadoop/ directory as input files.

```
hdfs dfs -put $HADOOP_ HOME/etc/hadoop/* input
```

3. Run the "grep" example provided.

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar grep input
output 'hadoop'
```

4. View the output files on HDFS.

```
hdfs dfs -cat output/*
```

Or copy the output files to the local filesystem.

```
hdfs -get output output
cat output/*
```

You should see the output as follows.

```
167         hadoop
```

Stop Daemons

If you are done, you can stop all daemons by using this code:

```
$HADOOP_ HOME/sbin/stop-dfs.sh
$HADOOP_ HOME/sbin/stop-yarn.sh
```

Harp

Harp is a collective communication libaray + a set of programming interfaces for machine learning applications. It has following features:

1. Hadoop 2.6.0 plugin
2. Hierarchal data abstraction (arrays/objects, partitions/tables)
3. Pool based memory management
4. Collective + event-driven programming model (distributed computing)
5. Dynamic Scheduler + Static Scheduler (multi-threading)

Preparation

1. Download Harp3-Project

Download **Harp3-Project** and put it under **software** directory. Then the directory is /home/ubuntu/software/Harp3-Project-master.

It's available at https://github.iu.edu/IU-Big-Data-Lab/Harp3-Project

2. ant

```
cd ~/software
wget http://www-us.apache.org/dist/ant/binaries/apache-ant-1.9.7-bin.tar.gz
tar xzf apache-ant-1.9.7-bin.tar.gz
mv apache-ant-1.9.7-bin.tar.gz ~/zips
```

3. Environment variables

```
vim .bashrc
export HARP3_PROJECT_HOME=~/software/Harp3-Project-master
export HARP3_HOME=$HARP3_PROJECT_HOME/harp3
export ANT_HOME=~/software/apache-ant-1.9.7
export PATH=$ANT_HOME/bin:$PATH
```

4. stop Hadoop

If hadoop is still running, stop it first, because you will modify hadoop configuration files.

```
$HADOOP_PREFIX/sbin/stop-dfs.sh
$HADOOP_PREFIX/sbin/stop-yarn.sh
```

COMPILATION & INSTALLATION

1. compile harp

```
cd $HARP3_HOME
ant
```

2. install harp to hadoop

a) Copy harp-0.3.0-hadoop-2.6.0.jar and fastutil-7.0.13.jar from $HARP3_HOME/lib/ into $HADOOP_HOME/share/hadoop/mapreduce

```
cp $HARP3_HOME/lib/harp-0.3.0-hadoop-2.6.0.jar $HADOOP_HOME/share/hadoop/mapreduce
cp $HARP3_HOME/lib/fastutil-7.0.13.jar $HADOOP_HOME/share/hadoop/mapreduce
```

b) Edit $HADOOP_HOME/etc/hadoop/mapred-site.xml, add java opts settings for map-collective tasks. For example:

```
<!-- set map-collective container memory size -->
<property>
    <name>mapreduce.map.collective.memory.mb</name>
    <value>512</value>
</property>
<property>
    <name>mapreduce.map.collective.java.opts</name>
    <value>-Xmx256m -Xms256m</value>
</property>
```

To develop Harp applications, remember to add the following property in job configuration:

```
jobConf.set("mapreduce.framework.name", "map-collective");
```

EXAMPLE

1. copy example/harp-app-hadoop-2.6.0-kmeans.jar to $HADOOP_HOME

We have kmeans compiled for you. Copy harp3-app-hadoop-2.6.0-kmeans.jar to $HADOOP_HOME

```
cp $HARP3_PROJECT_HOME/harp3-app/example/harp3-app-hadoop-2.6.0-kmeans.jar $HADOOP_HOME
```

2. Start Hadoop

```
$HADOOP_HOME/sbin/start-dfs.sh
$HADOOP_HOME/sbin/start-yarn.sh
```

3. Run Kmeans job

Use the following command to run harp kmeans:

```
cd $HADOOP_HOME

hadoop jar harp3-app-hadoop-2.6.0-kmeans.jar edu.iu.kmeans.KMeansLauncher <num of points> <num of centroids> <vector size> <num of point files per worker> <number of map tasks> <num threads> <number of iteration> <work dir> <local points dir>
```

<num of points>: the number of data points you want to generate randomly
<num of centriods>: the number of centroids you want to clustering the data to
<vector size>: the number of dimension of the data
<num of point files per worker>: how many files which contain data points in each worker
<number of map tasks>: number of map tasks
<num threads>: how many threads to launch in each worker
<number of iteration>: the number of iterations to run
<work dir>: the root directory for this running in HDFS

<local points dir>: the harp kmeans will firstly generate files which contain data points to local directory. Set this argument to determine the local directory.

For example:

```
hadoop jar harp3-app-hadoop-2.6.0-kmeans.jar edu.iu.kmeans.KMeansLauncher 1000 10 100 5 2 2 10 /kmeans /tmp/kmeans
```

To fetch the results, use the following command:

```
hdfs dfs –ls /
```