Tessera Stack Cluster Installation Manual

tesseradata.org

October 2014

Contents

1 Overview

The Tessera stack comprises the components RHIPE, Datadr, and Trelliscope. This is a guide to installing all of these components on a multi-node cluster. In particular, installing and running RHIPE requires Hadoop, an embarrassingly parallel computational platform. We have tested the Tessera stack on Debian Linux Derivative - Ubuntu (Version 10.04 or 12.04).

Installion of the Tessera stack should proceed in the following steps:

1. Installation Step

   The following packages (listed in order of installation) are required to run the Tessera stack successfully.

   - System Update and Packages
   - Sun/Oracle Java - 6 or 7 (For 7 use any release before subversion 51)
   - Protocal Buffer - 2.4.1
   - Cloudera Hadoop - cdh3u5
   - R - 3.0.1
   - Binutils (specific to Ubuntu) and rJava - 0.96
   - RHIPE - 0.74
   - R Packages: codetools, MASS, lattice, boot, shiny, and devtools
   - Datadr
   - Trelliscope
   - Rstudio
   - Shiny Server

   Note: Some of these packages are updated quite frequently. Please verify that they are compatible with the Tessera stack and Hadoop before installing.

2. Configuration Step

   - Host File
   - Hadoop Core
   - Hadoop MapReduce
   - Hadoop HDFS

3. Hadoop Cluster Starting

   The details of installing each component are described in the subsequent sections.

2 Installation Step

2.1 System Update and Packages

The steps to install the packages must be done on all the machines of the cluster, unless specified. All the packages are installing using command line.

Firstly, system packages must be updated.

```
sudo apt-get update
sudo apt-get -y upgrade
```

Now install package config package

```
sudo apt-get install pkg-config
```

2.2 Java

First install the Sun/Oracle Java repository and then install Java package.

```
apt-get install -y python-software-properties
add-apt-repository ppa:webupd8team/java
apt-get update
apt-get install -y oracle-java6-installer
```

Now add the path to the Java package to your environment

```
echo -e "JAVA_HOME=/usr/lib/jvm/java-6-oracle  \
JAVA_BIN=$JAVA_HOME/bin \
PATH=$PATH:$JAVA_HOME:$JAVA_BIN \
export JAVA_HOME    \
export JAVA_BIN \
export PATH" >> /etc/profile
```

2.3 Protocol Buffers

Download, compile, and install the protocol buffers.

```
wget http://protobuf.googlecode.com/files/protobuf-2.4.1.tar.gz
sudo tar -xzf protobuf-2.4.1.tar.gz
cd protobuf-2.4.1
sudo ./configure
sudo make
sudo make install
sudo ldconfig
```

2.4 Hadoop

Cloudera Hadoop installation is simply done through Aptitude package installer.

```
sudo apt-get install -y hadoop-0.20 hadoop-0.20-conf-pseudo \
hadoop-0.20-datanode hadoop-0.20-doc hadoop-0.20-native \
hadoop-0.20-pipes hadoop-0.20-sbin hadoop-0.20-tasktracker \
hadoop-0.20-namenode hadoop-0.20-secondarynamenode hadoop-0.20-jobtracker
```

Next, add the Hadoop home, binaries, and libraries to environment path.

```
echo -e "export HADOOP=/usr/lib/hadoop-0.20 \
export HADOOP_HOME=$HADOOP  \
export HADOOP_BIN=$HADOOP/bin   \
export HADOOP_LIB=$HADOOP/lib   \
export HADOOP_CONF_DIR=$HADOOP/conf \
export PATH=$HADOOP:$HADOOP_BIN:$HADOOP_HOME:$HADOOP_LIB:$PATH" >> /etc/bashrc
```

2.5 R

```
sudo apt-get install -y r-base r-base-dev r-recommended \
r-cran-rodbc ess
```

2.6 Binutils and rJava

Binutils and rJava are packages required by RHIPE. Binutils can be installed using the aptitude package installer.

```
sudo apt-get install binutils-gold
```

Before installing rJava ensure that `JAVA_HOME` environment variable is pointing to the Sun/Oracle installation of Java. If not, log out and log back into the system. rJava needs to be downloaded and installed using R.

```
wget http://cran.r-project.org/src/contrib/rJava_0.9-6.tar.gz
sudo R CMD INSTALL rJava_0.9-6.tar.gz
```

2.7 RHIPE

RHIPE is also to be downloaded and installed using R.

```
wget https://raw.github.com/saptarshiguha/RHIPE/master/code/RHIPE_0.73.tar.gz
R CMD INSTALL RHIPE_0.73.tar.gz
```

2.8 R Packages

The R packages codetools, MASS, lattice, boot, shiny, and devtools are required for installing Trelliscope and Datadr. They are installed by using install.packages command in R.

```
sudo R -e "install.packages('codetools', repos='http://cran.rstudio.com/')"
sudo R -e "install.packages('MASS', repos='http://cran.rstudio.com/')"
sudo R -e "install.packages('lattice', repos='http://cran.rstudio.com/')"
sudo R -e "install.packages('boot', repos='http://cran.rstudio.com/')"
sudo R -e "install.packages('shiny', repos='http://cran.rstudio.com/')"
sudo R -e "install.packages('devtools', repos='http://cran.rstudio.com/')
```

openssl package is also required and can be installed with apititude.

```
sudo apt-get -y install libcurl4-openssl-dev
```

2.9 Datadr and Trelliscope

Datadr and trelliscope is installed using `install_github` function and R.

```
sudo R -e "options(repos = 'http://cran.rstudio.com/');
library(devtools); install_github('datadr', 'hafen')"
sudo R -e "options(repos = 'http://cran.rstudio.com/');
library(devtools); install_github('trelliscope', 'hafen')"
```

2.10 Rstudio

Rstudio is an optional package. It can be used in lieu of command line R. It needs to be installed only on the Namenode/Jobtracker machine.

```
wget http://download2.rstudio.org/rstudio-server-0.98.507-amd64.deb
     -O /tmp/rstudio-server.deb
sudo apt-get -y install gdebi-core
sudo gdebi --n /tmp/rstudio-server.deb
```

2.11 Shiny Server

Cmake needs to be compiled for shiny server.

```
wget http://www.cmake.org/files/v2.8/cmake-2.8.12.2.tar.gz
tar xzf cmake-2.8.12.2.tar.gz
cd cmake-2.8.12.2
./configure
make
```

Shiny-server code is copied from github and compiled. Configuration file is placed in `/etc/shiny-server/`

```
git clone https://github.com/rstudio/shiny-server.git
sudo mkdir -p /etc/shiny-server
sudo cp shiny-server/config/default.config /etc/shiny-server/shiny-server.conf

# Get into a temporary directory in which we'll build the project
cd shiny-server
mkdir tmp
cd tmp

# Add the bin directory to the path so we can reference node
DIR=`pwd`
PATH=$PATH:$DIR/../bin/

# See the "Python" section below if your default python version is not 2.6 or 2.7.
PYTHON=`which python`

# Check the version of Python. If it's not 2.6.x or 2.7.x, see the Python section below.
$PYTHON --version

# Use cmake to prepare the make step. Modify the "--DCMAKE_INSTALL_PREFIX"
# if you wish the install the software at a different location.
../../cmake-2.8.12.2/bin/cmake -DCMAKE_INSTALL_PREFIX=/usr/local -DPYTHON="$PYTHON" ../
# Get an error here? Check the "How do I set the cmake Python version?" question below

# Recompile the npm modules included in the project
make -j4
mkdir ../build
```

```
(cd .. && bin/npm --python="$PYTHON" rebuild)

# Need to rebuild our gyp bindings since 'npm rebuild' won't run gyp for us.
(cd .. && ext/node/lib/node_modules/npm/node_modules/node-gyp/bin/node-gyp.js
    --python="$PYTHON" rebuild)

# Install the software at the predefined location
sudo make install

# POST INSTALL
# Place a shortcut to the shiny-server executable in /usr/bin
sudo ln -s /usr/local/shiny-server/bin/shiny-server /usr/bin/shiny-server
# Create shiny user.
# On some systems, you may need to specify the full path to 'useradd'
sudo useradd -r -m shiny

# Create log, config, and application directories
sudo mkdir -p /var/log/shiny-server
sudo mkdir -p /srv/shiny-server
sudo mkdir -p /var/lib/shiny-server
sudo chown shiny /var/log/shiny-server

#copy shiny examples
sudo mkdir /srv/shiny-server/examples
sudo cp -R /usr/local/lib/R/site-library/shiny/examples/* /srv/shiny-server/examples
sudo chown -R shiny:shiny /srv/shiny-server/examples
sudo -u shiny nohup shiny-server &
```

3 Configuration Step

3.1 Host File

Host file and Hadoop configuration files must be modified to reflect the cluster specific parameters. Host names and host IP addresses of all machines in the RHIPE cluster must be added to host file `/etc/hosts` on each machine of the cluster.

```
xxx.xxx.xxx.001 node001
xxx.xxx.xxx.002 node002
...
xxx.xxx.xxx.020 node020
```

`xxx.xxx.xxx.001` represents the IP address of a machine on the cluster. `node001` is the hostname of the machine. IP address and hostname can be added/removed to the hosts file as and when machines are added and removed from the cluster. The machine's fully qualified domain name or FQDN can also be used as hostnames.

3.2 Hadoop Core

Hadoop `core-site.xml` file is placed at `/etc/hadoop-0.20/conf`.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://node001:8020</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>hadoopdir-tmp-1-${username},hadoopdir-tmp-2-${username}</value>
</property>
<property>
<name>fs.checkpoint.dir</name>
<value>hadoopdir-ck-1,hadoopdir-ck-2</value>
</property>
<property>
<name>fs.trash.interval</name>
<value>0</value>
</property>
```

- `fs.default.name` defines your name node.
- `hadoop.tmp.dir` defines your temporary storing directory for jobs.
- `fs.checkpoint.dir` is where your hdfs checkpoint files are stored.
- `fs.trash.interval` is the amount of time deleted files are stored, in seconds.

3.3 Hadoop MapReduce

Hadoop `mapred-site.xml` is used to configure job and task trackers.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>node002/node001:8021</value>
</property>
<property>
<name>mapred.system.dir</name>
<value>/mapred/system/</value>
</property>
<property>
<name>mapred.local.dir</name>
<value>/hadoop/data/1/tmp/,/hadoop/data/2/tmp/</value>
</property>
<property>
<name>mapred.job.tracker.handler.count</name>
<value>n</value>
</property>
```

```
<property>
<name>tasktracker.http.threads</name>
<value>n</value>
</property>
<property>
<name>mapred.tasktracker.map.tasks.maximum</name>
<value>n/nodes</value>
</property>
<property>
<name>mapred.tasktracker.reduce.tasks.maximum</name>
<value>n/nodes</value>
</property>
```

- `mapred.job.tracker` is the hostname of the job tracker node.
- `mapred.system.dir` is where map reduce stores its system files.
- `mapred.local.dir` is where it stores its working files.
- `mapred.job.tracker.handler.count` is the number of job threads.
- `tasktracker.http.threads` is equal to the number of job threads.
- `mapred.tasktracker.map.tasks.maximum` and `mapred.tasktracker.reduce.tasks.maximum` are equal to the number of threads divided by number of nodes.

4 Hadoop Cluster Starting

The hadoop cluster must be started in the following order

On the Namenode and Jobtracker designated machine start the following. If Namenode and Jobtracker processes are to be run on seperate machines, then run these commands seperately.

```
service hadoop-0.20-namenode start && service hadoop-0.20-jobtracker start
```

On all the machines run the Datanode and Tasktracker processes. If the Datanode and Tasktracker processes are to be run on seperate machines, then start the processes seperately.

```
service hadoop-0.20-datanode start && service hadoop-0.20-tasktracker start
```

5 Installation Notes

5.1 Node Design

- The Datanode service running on the machines manages the HDFS.
- The Namenode service manages all the datanodes and all blocks of the data. It also manages data redundency. For example, when a disk fails Namenode is the service responsible for moving the data around to other nodes to ensure minimum level of replication.
- The Tasktracker service manages the different tasks that are run on each node.
- The Jobtracker service manages these Tasktrackers. It keeps information about which tasktracker is running a job and which of them are not. It also allocates jobs to all the tasktrackers and is responsible for breaking up the jobs into tasks and distributing it across the cluster.

- It must also be noted that in many configurations, the Namenode and the Jobtracker are run on the same machine. There are a few situations where these two are run separately.
- Most times the machine(s) that is/are running Namenode and Jobtracker services also run the Datanode and the Tasktracker service. This is mainly to extract all available computing power.
- Under conditions when the machine(s) that is/are running Namenode and Jobtracker services are not very powerful then the Datanode and the Tasktracker services are not run on them.
- The decision to have Datanode and Tasktracker on the Namenode and/or Jobtracker is completely dependent on the requirements of each environment.
- On each machine (Datanode and Tasktracker) majority of the available processor cores are used for Hadoop and usually a minimal number is kept of other system tasks. For example, on a 4 core Intel processor, 3 cores are set aside for Hadoop/Map/Reduce, while one core is allocated for system tasks. Similarly, on a 16 Core AMD processor, anywhere between 12 - 14 cores can be set aside for Hadoop and the rest for system tasks.

5.2 RAID and Redundancy Design

- RAID configuration are usually not used for HDFS. That is mainly because Hadoop itself handles redundancy.
- But if there is an opportunity to use RAID then it could definitely be used for `/boot` and `/partition`. This is mainly to ensure that when one of the hard drives go down the RAID configured partition on the other hard drive can seamlessly take over.
- The minimum required redundancy configuration for data on Hadoop is 3. This means that each data block is copied at 2 other locations other than its primary location. Hadoop has shown that with 3 copies of each data block high availability is possible.