Setup Distributed Computing Cluster Based on Hadoop, Zookeeper and HBase

Yong Li

CONTENTS

I. HADOOP + ZOOKEEPER + HBASE ENVIRONMENT

A. Introduction

Powered by today's high-performing distributed systems and the cloud computing technology, services and applications are increasingly hosted by servers in data centers and delivered to end users through the Internet infrastructure. To provide reliable and scalable services while guaranteeing high throughput, the huge amount of computing tasks as well as storage requirement from the users are usually distributed among multiple servers in clusters managed by distributed software frameworks such as Hadoop, Haystack, HBase, Spanner, etc. This report discusses the working flow and setups for distributed computing platform built atop clusters of commodity servers using Hadoop framework.

The Apache Hadoop is a framework that allows the distributed processing of large data sets across clusters of machines using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than relying on a single hardware device or machine to deliver high-availability, the Hadoop framework is designed to detect and handle failures at the application layer, thus delivering a highly available service on top of a cluster of computers, each of which may subject to failures.

B. Software Structures

This section discuss how Hadoop works together with Zookeeper and HBase to form a platform for distributed computing and storage.

Hadoop consists of a common library, a distributed file system (i.e. HDFS) which provides reliable and scalable data storage and a MapReduce framework (a high-performance parallel programming model similar to the Google MapReduce).

The common library is a set of useful shell commands, utilities and APIs for management and development on the Hadoop platform.

The HDFS features a master-slave architecture. A HDFS cluster consists of a single NameNode (NameNode is the name of the master node as well as the daemon running on the master node) and several DataNodes (DataNode is the name of the slave nodes as well as the daemons running on the slave nodes). The NameNode, with various kinds of metadata (e.g. EditLog and FsImage), manage the file system namespace/structure and regulate accesses to files by clients. While the DataNodes manage the low-level storage and perform the block creation, deletion and replication upon the instructions from the NameNode.

The Hadoop MapReduce component also has the similar structure with a JobTracker being the master and a TaskTracker being the slave in a cluster. The JobTracker is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The TaskTracker executes the tasks as directed by the master.

To manage and process large data set with billions of rows $\times$ millions of columns on top of clusters of servers, it is desirable to utilize a distributed database such as the Google Bigtable, which leverages the underlying parallel hardware resources. With Hadoop, the distributed database system HBase provides such capabilities. A table in HBase consists of a collection of splits, called regions, where each region stores a range partition of the key space. Its architecture comprises a three layered B+ tree structure; the regions storing data constitute the lowest level. The two upper levels are special regions referred to as the ROOT and META regions. The entire running instance of HBase includes several HRegionServers which serve subset of regions and a HMaster which is responsible to assign the regions to each HRegionServer when you start HBase. Zookeeper (daemon name QuorumPeerMain) is used to coordinate and synchronize the HBase daemons running distributely in the system.

C. General Configuration Suggestions

The scale of the Hadoop-based distributed system can drastically vary from a system with only a few servers to large clusters of thousands of servers, depending on the hosted applications and services. Typically, the NameNode (the master for HDFS) is running on a dedicated machine to serve the namespace/ file structure for a cluster with tens to a few hundreds of severs. Another machine is used to run the JobTracker, the master of the MapReduce framework. On each of other slave nodes in the cluster, DataNode and TaskTracker can run simultaneously to serve computing tasks as well as data block storage requests assigned by the master(s).

Generally, you should run ZooKeeper (QuorumPeerMain) on an odd number of servers (not necessarily exclusively) and the exact number of ZooKeepers to run depends on the desired level of reliability, tolerance to system failures and the scale of the DataNodes/ HRegionServers. Configure each ZooKeeper server with around 1GB of RAM and a dedicated disk. For heavily loaded clusters, run ZooKeeper servers on separate machines from region servers.

In a smaller cluster with less than 20 machines, it is possible to run the NameNode and JobTracker on top of the same physical server, as will be introduced in Section II.

More technical details about the Hadoop, Zookeeper and HBase can be found at Apache Hadoop web page: http://hadoop.apache.org/.

II. Setup and Configuration for An Experimental Hadoop-based Cluster

This section presents hardware and software configurations of a small cluster built for studying the functionalities and performance of the Hadoop-based computing environment. The cluster comprises 2 virtual machines running Ubuntu 11.04 OS within VMware Workstation 7.14 and they are connected using virtual network. One of the machine serves as the master and runs the NameNode (master daemon for HDFS), JobTracker (master daemon for MapReduce), QuorumPeerMain(daemon for ZooKeeper) and HMaster (master daemon for HBase). It also works as a slave for HDFS and MapReduce framework thus running DataNode (slave daemon for HDFS) and TaskTracker (slave daemon for MapReduce) as well. The other machine works as the slave and runs the DataNode, TaskTracker and HRegionServer (slave daemon for HBase). We will elaborate on how to install and configure such an environment in the following sections.

A. *Prerequisite*

Here is a list of software/system that is necessary for running Hadoop:

- **GNU/Linux**. Linux is supported as a development and production platform. Hadoop has been demonstrated on GNU/Linux clusters with 2000 nodes. In our case we use Ubuntu 11.04.
- **GNU build-essential**. For the potential interested developers, it is convenient to have GNU build tool chains. You can obtain and install build-essential (for GNU C, C++ compiler).
- **JavaTM 1.6.x**, preferably from Sun, must be installed. Here we use JDK binary installation package jdk-6u26-linux-i586.bin
- **tsocks** or its equivalents. It is needed in an environment where apps have to go through proxy or socks server for Internet access.
- **ssh** and **openssh-server** (sshd). ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons.
- **NTP**. The clocks on cluster members should be in basic alignments. Some skew is tolerable but wild skew could generate odd behaviors. Run NTP on your cluster, or an equivalent.

Here are the steps to install the required software and packages:

1) Download and install Ubuntu 11.04 using the installation image (ISO file) from the Ubuntu official site.

2) Download and install GNU build tool chain:

   ```
    sudo apt-get install build-essential
   ```

3) Download and install the newest JDK for Linux. If the Internet is not available yet, please download from another machine with Internet access and copy the installation package using a flush drive. Alternatively, you may install **tsocks** first to get your machines online and then use *apt-get install* command to install java automatically.
4) After you install java, you should put the following code into the startup script for your shell. In our case it is the *.bashrc* file located at the home directory:

   ```
   #Yong's customized settings
   JAVA_HOME=/home/yongli/Install/jdk1.6.0_26
   export JAVA_HOME
   PATH=/home/yongli/Install/jdk1.6.0_26/bin:$PATH
   export PATH
   ```

5) Install tsocks. You can download tsocks at http://sourceforge.net/projects/tsocks. Unzip it by

   ```
   tar -xzvf tsocks-versionnumber.tar.gz
   ```

   Then *cd* to the unziped directory and install it by

   ```
    ./configure
     sudo make
     sudo make install
   ```

   Now when you run *apt-get* through tsocks:

   ```
   sudo tsocks apt-get install mysql-server
   ```

   it will complain that

   ```
    ERROR: ld.so: object '/usr/lib/libtsocks.so'
   from LD\_PRELOAD cannot be preloaded: ignored
   ```

   To address this issue, copy the lib file to the path where program can find:

   ```
   sudo cp libtsocks.so /usr/lib
   ```

   The above step is needed for some other Linux installation. Please check the installation directory or other possible directories to find the necessary lib files and copy them to the right place. Use

   ```
   find /(or other path) -name filename
   ```

   to locate an interested file. After the lib file being copied, create the configuration file for tsocks by

   ```
   sudo cp tsocks.conf.complex.example /etc/tsocks.conf
   ```

   and edit it as:

```
#your local network without proxy:
local = 192.168.111.0/255.255.255.0
local = 172.16.120.0/255.255.255.0
local = 10.0.0.0/255.0.0.0
#path {
#    reaches = 150.0.0.0/255.255.0.0
#...
#}
#the address/name for your socks server:
server = your.proxy.server
# Server type (5 or 4):
server_type = your socks type
# Change the port number based on your environment:
server_port = your socks server port
```

After the above steps, you may still have problems with the *apt-get* command. If that is the case, please clicked system− >preferences− >network proxy, and check the "manual proxy configuration" box, and put your.socks.server as the socks host and your port number as the port. Click "apply system-wide" and then close the window.
Now you should now be able to install openssh-server using apt-get install:

```
sudo tsocks apt-get install openssh-server
```

6) Download and install openssh-server using the above command. Please remove tsocks from the above command if there is no socks or proxy server in your environment. After your install it, *sshd* should be automatically started. You may check if *sshd* is running using *ps -ef | grep sshd*.
   After *sshd* running on the machines in your cluster, you should test if these machines can talk with each other through *ssh*.
7) Install and configure *NTP* using *apt-get install* so that the time of the machines in the cluster is synchronized.

B. Installation and Configuration Hadoop

In this document we choose Ubuntu11.04, hadoop0.20.203, zookeeper 3.3.3 and HBase0.90.3 to setup a testing platform comprising two virtual machines (please refer the first paragraph in Section II for a description of the platform). Please notice that there might be data loss problems for the HBase0.90.3 working with hadoop0.20.203, as stated in the hadoop documentation page:

"This version(0.90.3) of HBase will only run on Hadoop 0.20.x. It will not run on hadoop 0.21.x (nor 0.22.x). HBase will lose data unless it is running on an HDFS that has a durable sync. Hadoop 0.20.2 and Hadoop 0.20.203.0 DO NOT have this attribute. Currently only the branch-0.20-append branch has this attribute[1]. No official releases have been made from the branch-0.20-append branch up to now so you will have to build your own Hadoop from the tip of this branch. Michael Noll has written a detailed blog, Building an Hadoop 0.20.x version for HBase 0.90.2, on how to build an Hadoop from branch-0.20-append."

It is important that you are aware of this problem. However, the installation and setup introduced in this document also applies to other Hadoop versions.

Follow the below steps for configuring the hadoop environment:

1) Finish the installation and setups in Section II-A. Notice you should do this for all the machines in your cluster.
2) Edit /etc/hostname to properly and clearly name each of the machines in the cluster. In our two-machine cluster, we have one machine named *master*, serving as the master for HDFS (runing NameNode daemon), MapReduce (running JobTracker daemon), Zookeeper (running QuorumPeerMain daemon) and HBase (running HMaster daemon), and another machine named *slave-01*, serving as HDFS and HBase slave. Since we only have two machines, the *master* machine also serve as a second DataNode, as stated before.
3) Edit /etc/hosts to provide local DNS/routing information. In our case for example, the /etc/hosts file on *master* machine has the content as follows:
   ```
   127.0.0.1 localhost
   192.168.111.142 master
   192.168.111.141 slave-01
   ```
   For *slave-01*, the hosts file looks like:
   ```
   127.0.0.1 localhost
   192.168.111.141 slave-01
   192.168.111.142 master
   ```
   Please remember to update the hosts files if you reboot your machines and the IP addresses are re-assigned by a DHCP server. Use *ifconfig* command to find out the IP address if necessary.
4) On all the machines, download hadoop-0.20.203.tar.gz from Apache hadoop website. Extract hadoop archive file to /home/Install/ directory. After the extraction, there should be a path like /home/Install/hadoop-0.20.203.0/.
5) You now need do a SSH Key Authentication since hadoop master uses passwordless ssh access to control and coordinate with the slaves. Make sure you are the hadoop user for all of these commands. If you have not yet installed Hadoop and/or created the hadoop user you should do that first.
   First from the master node check that you can ssh to the localhost without a passphrase: *ssh localhost* If you cannot ssh to localhost without a passphrase, execute the following commands:

```
$ ssh-keygen -t dsa -P "" -f ~.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

On your master node try to ssh again (as the hadoop user) to your localhost and if you are still getting a password prompt then:

```
$ chmod go-w $HOME $HOME/.ssh
$ chmod 600 $HOME/.ssh/authorized_keys
$ chown `whoami` $HOME/.ssh/authorized_keys
```

Now you need to copy your public key to all of your "slave" machine (don't forget your secondary name node. If slave's hadoop user does not have a .ssh directory you should create it (*mkdir /.ssh*)):

```
$ scp ~/.ssh/id_dsa.pub slave1:~/.ssh/master.pub
```

Now login (as the hadoop user) to your slave machine. While on your slave machine add your master machine's hadoop user's public key to the slave machine's hadoop authorized key store:

```
cat ~/.ssh/master.pub >> ~/.ssh/authorized_keys
```

Now, from the master node try to ssh to slave. *ssh slave-01* If you are still prompted for a password (which is most likely) then it is very often just a simple permission issue. Go back to your slave node again and as the hadoop user run this

```
$ chmod go-w $HOME $HOME/.ssh
$ chmod 600 $HOME/.ssh/authorized_keys
$ chown `whoami` $HOME/.ssh/authorized_keys
```

Try again from your master node. *ssh slave-01* And you should be good to proceed. Repeat for all Hadoop Cluster Nodes.

6) *cd* to the hadoop installation directory and create three directories: *dfsname*, *dfsdata* and *tmp*. The dfsname directory is used to store the namespace and logs (this is specified by the dfs.name.dir property in hdfs-site.xml, as you will see in the following steps). The dfsdata folder is the local filesystem path of a DataNode where it should store its data blocks (this is specified by the dfs.data.dir property in hdfs-site.xml). The tmp directory is the base for other temporary directories used by Hadoop. Hadoop has its own default directories to store those files. However, the three directories are necessary if you installed hadoop using an account other than the root. Without the these directories, you will experience problems starting the HDFS master daemons (NameNode). To diagnose, please check the log files in the log directory in the hadoop installation directory.
7) Put the following code in HadoopInstallationDir/conf/core-site.xml on both *master* and *slave-01* machines:

```
<configuration>
```

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>HadoopInstallationDir/tmp</value>
  <description>A base for other temporary
              directories.</description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://master:9002</value>
  <description>The name and port numer of
  the default file system. </description>
</property>
</configuration>
```

Notice the port number for HDFS does not have to be 9002.

8) Put the following code in HadoopInstallationDir/conf/mapred-site.xml on both *master* and *slave-01* machines:

```
<configuration>
<property>
  <name>mapred.job.tracker</name>
  <value>master:9001</value>
  <description>The host and port that the MapReduce
  job tracker runs at.  If "local", then jobs are run
  in-process as a single map and reduce task.</description>
</property>
</configuration>
```

Notice the port number for MapReduce does not have to be 9001.

9) Put the following code in HadoopInstallationDir/conf/hdfs-site.xml on both *master* and *slave-01* machines:

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
  </description>
```

```
</property>

<property>
  <name>dfs.name.dir</name>
  <value>HadoopInstallationDir/dfsname</value>
  <description>Directory to Store NameSpace and Logs.
  </description>
</property>

<property>
  <name>dfs.data.dir</name>
  <value>HadoopInstallationDir/dfsdata</value>
  <description>Directory to Store Data.
  Comma separated list of paths on the local filesystem
  of a DataNode where it should store its blocks
  </description>
</property>

</configuration>
```

10) Add hadoop to environment variable by modifying /.bashrc
```
 export HADOOP_HOME=HadoopInstallationDir
 export PATH=$HADOOP_HOME/bin:$PATH
```
This step is not mandatory not for your convenience.

11) Modify HadoopInstallationDir/conf/masters only on *master* machine as:
```
master
```
Modify HadoopInstallationDir/conf/slaves only on *master* machine as:
```
master
slave-01
```

12) Format the NameNode only once and only on *master* machine
```
HadoopInstallationDir/bin/hadoop namenode -format
```
If there is confirmation prompt, type *Y* to proceed. Please use the upper case.

13) Download and extract install zookeeper only on *master* node:
```
tar -xzvf zookeeper-3-3.2.tar.gz
```
Then copy conf/zoo_sample.cfg to conf/zoo.cfg:
```
cp conf/zoo_sample.cfg confg/zoo.cfg
```

Change the following line in conf/zoo.cfg:

```
 dataDir=/home/hadoop/zookeeper/snapshot
```

14) Download and install Hbase on all nodes:

```
tar -xzvf hbase-0.90.3.tar.gz
```

Then edit HBaseInstallationDir/conf/hbase-site.xml and put the following code (on all machines) to configure HBase as fully-distributed:

```
<configuration>
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://master:9002/Install/hbase-0.90.3</value>
    <description> The directory shared by region servers.
    </description>
  </property>
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
    <description>The mode the cluster will be in.
     Possible values are false: standalone and pseudo-
     distributed setups with managed Zookeeper
     true: fully-distributed with unmanaged Zookeeper
     Quorum (see hbase-env.sh)
    </description>
  </property>
  <property>
      <name>hbase.zookeeper.quorum</name>
      <value>master</value>
      <description>Comma separated list of servers in
      the ZooKeeper Quorum.
      If HBASE_MANAGES_ZK is set in hbase-env.sh this
      is the list of servers which we will start/stop
      ZooKeeper on.
      </description>
    </property>
</configuration>
```

Notice the hbase.rootdir property should be modified according to your case. I installed HBase in /home/Install/hbase-0.90.3/ and use the port number 9002 as the HDFS port, therefore the hbase.rootdir should be

```
hdfs://master:9002/Install/hbase-0.90.3
```

15) Modify environment variables in HBaseInstallationDir/conf/hbase-env.sh on all machines:

```
export JAVA_HOME="$HOME/Install/jdk1.6.0_26"
export HBASE_IDENT_STRING=$HOSTNAME
export HBASE_MANAGES_ZK=false
```

Change the JAVA_HOME based on your JDK installation. We want ZooKeeper to run separately from HBase and set HBASE_MANAGES_ZK to false.

16) Overwrite HBaseInstallationDir/conf/regionservers on both machines as

```
slave-01
```

If you want *master* to server as a HRegionServer as well, you can add *master* to the HBaseInstallationDir/conf/regionservers on both machines:

```
slave-01
master
```

17) Copy HadoopInstallationDir/haoop-core-versionnumber.jar to HBaseInstallationDir/lib/ on both machines. This is very important to fix version difference issue. Pay attention to its ownership and mode(755). If later on when you start HBase, the HMater daemon is not running, please check the log in the hbase installation directory. If the log indicates there is configuration exception, please copy HadoopInstallationDir/lib/commons-configuration-1.6.jar to HBaseInstallationDir/lib/ on both machines. This step is needed in our case.

18) Start zookeeper:

```
 cd ~/zookeeper
 bin/zkServer.sh start
```

We can test if zookeeper is running correctly by typing

```
 ~/zookeeper/bin/zkCli.sh Cserver 127.0.0.1:2181
```

19) Start hadoop cluster:

```
HadoopInstallationDir/bin/start-dfs.sh
HadoopInstallationDir/bin/start-mapred.sh
```

20) Start Hbase:

```
HBaseInstallationDir/bin/start-hbase.sh
```

21) Use HBase shell:

```
HBaseInstallationDir/bin/hbase shell
```

22) check all daemons are running by *jps*. The following Java daemons should be displayed on the *master*:

```
QuorumPeerMain
DataNode
```

```
HMaster
JobTracker
NameNode
TaskTracker
```

The following Java daemons should be displayed on the *slave-01*:

```
DataNode
HRegionServer
TaskTracker
```

23) Later on, stop the multi-node cluster by typing following code only on *master*:

```
HBaseInstallationDir/bin/stop-hbase.sh
HadoopInstallationDir/bin/stop-mapred.sh
HadoopInstallationDir/bin/stop-dfs.sh
ZooKeeperInstallationDir/bin/zkServer.sh stop
```