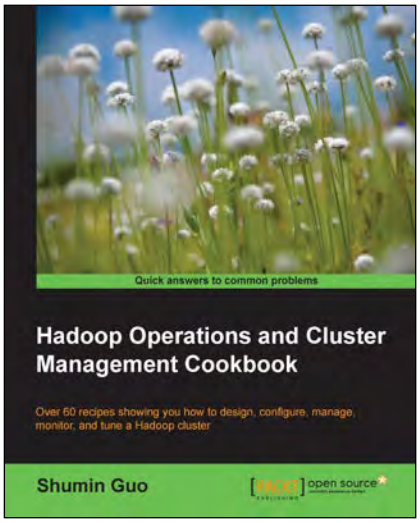
[PACKT] open source
community experience distilled
PUBLISHING

Hadoop Operations and Cluster Management Cookbook

Shumin Guo



Chapter No. 3
"Configuring a Hadoop Cluster"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.3 "Configuring a Hadoop Cluster"

A synopsis of the book's content

Information on where to buy this book

About the Author

Shumin Guo is a PhD student of Computer Science at Wright State University in Dayton, OH. His research fields include Cloud Computing and Social Computing. He is enthusiastic about open source technologies and has been working as a System Administrator, Programmer, and Researcher at State Street Corp. and LexisNexis.

I would like to sincerely thank my wife, Min Han, for her support both technically and mentally. This book would not have been possible without encouragement from her.

Hadoop Operations and Cluster Management Cookbook

Today, many organizations are facing the Big Data problem. Managing and processing Big Data can incur a lot of challenges for traditional data processing platforms such as relational database systems. Hadoop was designed to be a distributed and scalable system for dealing with Big Data problems. A Hadoop-based Big Data platform uses Hadoop as the data storage and processing engine. It deals with the problem by transforming the Big Data input into expected output.

Hadoop Operations and Cluster Management Cookbook provides examples and step-by-step recipes for you to administrate a Hadoop cluster. It covers a wide range of topics for designing, configuring, managing, and monitoring a Hadoop cluster. The goal of this book is to help you manage a Hadoop cluster more efficiently and in a more systematic way.

In the first three chapters, you will learn practical recipes to configure a fully distributed Hadoop cluster. The subsequent management, hardening, and performance tuning chapters will cover the core topics of this book. In these chapters, you will learn practical commands and best practices to manage a Hadoop cluster. The last important topic of the book is the monitoring of a Hadoop cluster. And, we will end this book by introducing steps to build a Hadoop cluster using the AWS cloud.

What This Book Covers

Chapter 1, Big Data and Hadoop, introduces steps to define a Big Data problem and outlines steps to build a Hadoop-based Big Data platform.

Chapter 2, Preparing for Hadoop Installation, describes the preparation of a Hadoop cluster configuration. Topics include choosing the proper cluster hardware, configuring the network, and installing the Linux operating system.

Chapter 3, Configuring a Hadoop Cluster, introduces recipes to configure a Hadoop cluster in pseudo-distributed mode as well as in fully distributed mode. We will also describe steps to verify and troubleshoot a Hadoop cluster configuration.

Chapter 4, Managing a Hadoop Cluster, shows you how to manage a Hadoop cluster. We will learn cluster maintenance tasks and practical steps to do the management. For example, we will introduce the management of an HDFS filesystem, management of MapReduce jobs, queues and quota, and so on.

Chapter 5, Hardening a Hadoop Cluster, introduces recipes to secure a Hadoop cluster. We will show you how to configure ACL for authorization and Kerberos for authentication, configure NameNode HA, recover from a failed NameNode, and so on.

Chapter 6, Monitoring a Hadoop Cluster, explains how to monitor a Hadoop cluster with various tools, such as Ganglia and Nagios.

Chapter 7, Tuning a Hadoop Cluster for Best Performance, introduces best practices to tune the performance of a Hadoop cluster. We will tune the memory profile, the MapReduce scheduling strategy, and so on to achieve best performance for a Hadoop cluster.

Chapter 8, Building a Hadoop Cluster with Amazon EC2 and S3, shows you how to configure a Hadoop cluster in the Amazon cloud. We will explain steps to register, connect, and start VM instances on EC2. We will also show you how to configure a customized AMI for a Hadoop cluster on EC2.

3
Configuring a Hadoop Cluster

In this chapter, we will cover:

- Choosing a Hadoop version
- Configuring Hadoop in pseudo-distributed mode
- Configuring Hadoop in fully-distributed mode
- Validating Hadoop installation
- Configuring ZooKeeper
- Installing HBase
- Installing Hive
- Installing Pig
- Installing Mahout

Introduction

After finishing all the preparing tasks, we are ready to configure a Hadoop cluster in this chapter. First, we will give you some tips on choosing a proper Hadoop release version. Then we will show you how to configure a Hadoop cluster in pseudo-distributed and fully-distributed mode. Pseudo-distributed mode is a very good starting point if you have no experience configuring a Hadoop cluster. In this mode, we will configure all the Hadoop daemons to run on a single machine, which can give us the first feeling of a working Hadoop cluster while minimizing configuration difficulties. Next, we will show you how to validate a Hadoop cluster. The importance of validating a Hadoop cluster configuration will never be overemphasized. We typically use this step to confirm that the Hadoop cluster is running as expected. The last few recipes will show you how to install a few components in the cluster.

Choosing a Hadoop version

As an open source project, Hadoop has been under active development over the past few years. New versions are being released regularly. These new releases either fix bugs contributed by the community, leading to a more stable Hadoop software stack, or add new features for the purpose of more full-fledged and enterprise-level distribution.

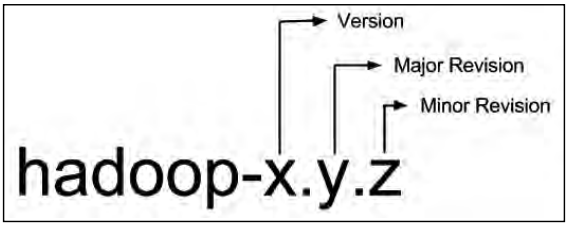
In this section, we are going to review the history of releases of Hadoop, pointing out features of these releases. More importantly, we will give tips on choosing a proper Hadoop distribution.

Getting ready

In general, the **release version number** of a Hadoop distribution consists of three parts: the **version** number, the **major revision** number, and the **minor revision** number.

Sometimes the revision number can have a fourth part, for example, 0.20.203.0, but this is relatively rare.

A Hadoop release name can be described with the following figure:



How to do it...

The following table shows features of major Hadoop releases:

Feature\Version	2.0.x	1.1.x	0.23.x	0.20.x
Stable		Yes		Yes
MRv1		Yes		Yes
MRv2	Yes		Yes	
Kerberos security	Yes	Yes	Yes	
HDFS federation	Yes		Yes	

Feature\Version	2.0.x	1.1.x	0.23.x	0.20.x
NameNode HA	Yes		Yes	
HDFS append	Yes	Yes	Yes	
HDFS symbolic links	Yes	Yes	Yes	

The table tells us that Hadoop is evolving rapidly, with new features such as security, **HDFS federation**, and **NameNode HA** being added over time. Another lesson we can learn from the table is that the most recent stable release, Version 1.1.x, does not contain all the features. And although release Version 2.0.x is the most feature-rich Hadoop release, it is still in alpha state requiring further improvements.

So, which version should you choose for your deployment? Generally, we need to consider two properties: stability and features. For a production deployment, we definitely want to deploy a stable release and we want to use the release that contains all the required features. Clearly, our current optimal and only choice is Version 1.1.x or specifically Version 1.1.2 as of this book's writing.

See also

- More information about Hadoop releases can be found at `http://hadoop.apache.org/releases.html`

Configuring Hadoop in pseudo-distributed mode

Pseudo-distributed mode refers to a Hadoop cluster configuration that contains only one node. This mode can be helpful for debugging and validation purposes. In this recipe, we will outline steps to configure Hadoop in pseudo-distributed mode.

Getting ready

Before configuring Hadoop in pseudo-distributed mode, we assume that we have a machine, for example, the master node of the Hadoop cluster, with Linux installed. We also assume that all the necessary tools have been installed and properly configured.

- The most important dependent software is Java, which is the programming language and library that Hadoop is based on. To check that Java has been properly installed, we can use the following command:

```
$ java -version
```

You should have output similar to the following:

```
java version "1.7.0_13"
Java(TM) SE Runtime Environment (build 1.7.0_13-b20)
Java HotSpot(TM) 64-Bit Server VM (build 23.7-b01, mixed mode)
```

- If you have installed OpenJDK other than the Oracle's official Java, the output will be similar to the following:

```
Java version "1.7.0_09-icedtea"
OpenJDK Runtime Environment (fedora-2.3.4.fc17-x86_64)
OpenJDK 64-Bit Server VM (build 23.2-b09, mixed mode)
```

If you have installed OpenJDK, please refer to the *Installing Java and other tools* recipe from *Chapter 2*, *Preparing for Hadoop installation*.

- Download the desired Hadoop distribution. In this book, we assume that we're using Hadoop release 1.1.2. To download a Hadoop release, please visit the following URL:

  `http://www.apache.org/dyn/closer.cgi/hadoop/common/`

  Choose the proper mirror site (or use the suggested link on top of the mirror). Start downloading by clicking on the proper Hadoop release. We suggest downloading a `.gzip` archived file with the filename ending with `tar.gz`.

- Alternatively, we can download a Hadoop release with the following command under Linux:

```
wget http://mirror.quintex.com/apache/hadoop/common/hadoop-1.1.2/
hadoop-1.1.2.tar.gz -P ~
```

- Last, we assume that the `ssh` password-less login has been properly configured.

How to do it...

Perform the following steps to configure Hadoop in pseudo-distributed mode:

1. Copy the Hadoop archive to the `/usr/local` directory:

```
sudo cp hadoop-1.1.2.tar.gz /usr/local
```

2. Decompress the Hadoop package archive:

```
cd /usr/local
sudo tar xvf hadoop-1.1.2.tar.gz
```

The uncompressed archive file will contain the following files and folders:

```
CHANGES.txt   c++                    hadoop-examples-1.1.2.jar
lib
LICENSE.txt   conf                   hadoop-minicluster-1.1.2.jar
libexec
NOTICE.txt    contrib                hadoop-test-1.1.2.jar
sbin
README.txt    hadoop-ant-1.1.2.jar   hadoop-tools-1.1.2.jar
share
bin           hadoop-client-1.1.2.jar ivy
src
build.xml     hadoop-core-1.1.2.jar  ivy.xml
webapps
```

The folder contains several `.jar` files and folders such as `bin`, `sbin`, and `conf`. The `.jar` files `hadoop-core-1.1.2.jar` and `hadoop-tools-1.1.2.jar` contain the core classes of Hadoop. The files `hadoop-examples-1.1.2.jar` and `hadoop-test-1.1.2.jar` contain sample MapReduce jobs.

The `conf` folder contains cluster configuration files, the `bin` folder contains commands and scripts to start and stop a cluster, and the `sbin` folder contains scripts to perform specific tasks.

3. Make a soft link for Hadoop root directory:

```
sudo ln -s hadoop-1.1.2 hadoop
```

4. Use your favorite text editor to open the file `~/.bashrc` and add the following contents:

```
export JAVA_HOME=/usr/java/latest
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$JAVA_HOME/bin:HADOOP_HOME/bin
```

We are assuming Oracle Java has been installed under the `/usr/java/latest` directory.

5. Reload the configuration file `~/.bashrc` with the following command:

```
. ~/.bashrc
```

6. Use your favorite text editor to open the file `$HADOOP_HOME/conf/hadoop-env.sh`, and change the `JAVA_HOME` environment variable to the following:

```
export JAVA_HOME=/usr/Java/latest
```

7. Use your favorite text editor to open the file `$HADOOP_HOME/conf/core-site.xml`, and add the following content:

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
  </property>

  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:54311</value>
  </property>

  <property>
    <name>hadoop.tmp.dir</name>
    <value>/hadoop/tmp/</value>
  </property>
</configuration>
```

8. Use your favorite text editor to open the file `$HADOOP_HOME/conf/hdfs-site.xml`, and add the following content to the file:

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

  <property>
    <name>dfs.data.dir</name>
    <value>/hadoop/data/</value>
  </property>
</configuration>
```

9. Use your favorite text editor to open the file `$HADOOP_HOME/conf/mapred-site.xml`, and add the following content:

```
<configuration>
  <property>
    <name>mapred.system.dir</name>
    <value>/hadoop/mapred</value>
  </property>
</configuration>
```

10. Ask `localhost` to run the SecondaryNameNode daemon with the following command:

```
sudo echo "localhost" > $HADOOP_HOME/conf/masters
```

11. Configure `localhost` as the single slave node with the following command:

```
sudo echo "localhost" > $HADOOP_HOME/conf/slaves
```

Use the following steps to start and stop a Hadoop cluster:

1. Format the HDFS filesystem from NameNode with the following command:

```
hadoop namenode -format
```

We will get output similar to the following:

```
13/02/14 01:43:12 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 1.1.2
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/
common/branches/branch-1.0 -r 1393290; compiled by 'hortonfo' on
Wed Oct  3 05:13:58 UTC 2012
************************************************************/
13/02/14 01:43:13 INFO util.GSet: VM type       = 64-bit
13/02/14 01:43:13 INFO util.GSet: 2% max memory = 17.77875 MB
13/02/14 01:43:13 INFO util.GSet: capacity      = 2^21 = 2097152
entries
13/02/14 01:43:13 INFO util.GSet: recommended=2097152,
actual=2097152
13/02/14 01:43:13 INFO namenode.FSNamesystem: fsOwner=shumin
13/02/14 01:43:13 INFO namenode.FSNamesystem:
supergroup=supergroup
13/02/14 01:43:13 INFO namenode.FSNamesystem:
isPermissionEnabled=true
13/02/14 01:43:13 INFO namenode.FSNamesystem: dfs.block.
invalidate.limit=100
13/02/14 01:43:13 INFO namenode.FSNamesystem:
isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s),
accessTokenLifetime=0 min(s)
```

```
13/02/14 01:43:13 INFO namenode.NameNode: Caching file names
occuring more than 10 times
13/02/14 01:43:13 INFO common.Storage: Image file of size 112
saved in 0 seconds.
13/02/14 01:43:14 INFO common.Storage: Storage directory /hadoop/
tmp/dfs/name has been successfully formatted.
13/02/14 01:43:14 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/127.0.0.1
************************************************************/
```

2. Start the HDFS daemons with the following command:

```
start-dfs.sh
```

We will get output similar to the following:

```
starting namenode, logging to /usr/local/hadoop/libexec/../logs/
hadoop-hduser-namenode-localhost.out
localhost: starting datanode, logging to /usr/local/hadoop/Hadoop/
libexec/../logs/hadoop-hduser-datanode-localhost.out
localhost: starting secondarynamenode, logging to /usr/local/
hadoop/libexec/../logs/hadoop-hduser-secondarynamenode-localhost.
out
```

The output shows that the following HDFS daemons have been started: NameNode, DataNode, and SecondaryNameNode.

3. Start the MapReduce daemons with the following command:

```
start-mapred.sh
```

The output will be similar to the following:

```
starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/
hadoop-hduser-jobtracker-localhost.out
localhost: starting tasktracker, logging to /usr/local/hadoop/
libexec/../logs/hadoop-hduser-tasktracker-localhost.out
```

The output shows that the JobTracker and TaskTracker MapReduce daemons have been started.

4. With the `jps` command, we can get a list of all running daemons as follows:

```
10984 SecondaryNameNode
11272 TaskTracker
11144 JobTracker
26966 NameNode
10855 DataNode
27183 Jps
```

So far, all the Hadoop daemons have been started.

5. Stop the MapReduce daemons with the following command:

```
stop-mapred.sh
```

6. Stop the HDFS daemons with the following command:

```
stop-hdfs.sh
```

How it works...

Under Unix-like operating systems, system runtime configurations and environment variables are specified via plain text files. These files are called run configuration files, meaning that they provide configurations when the program runs. For example, the `.bashrc` file under a user's home directory is the run configuration file for bash shell. It will be sourced (loaded) automatically every time when a bash terminal is opened. So, in this file, we can specify commands and environment variables for a running bash environment.

.bashrc OR .bash_profile

Under Linux, the bash shell has two run configuration files for a user, `.bashrc` and `.bash_profile`. The difference between the two files is that `.bash_profile` is executed for login shells, while `.bashrc` is for interactive, non-login shells. More specifically, when we log in to the system by entering username and password either locally or from a remote machine, `.bash_profile` will be executed, and a bash shell process is initialized. On the other hand, if we open a new bash terminal after logging into a machine or type the `bash` command from command line,the `.bashrc` file will be used for initialization before we see the command prompt on the terminal window. In this recipe, we used the `.bashrc` file, so that new configurations will be available after opening a new bash process. Alternatively, we can manually source a configuration file after it is created or changed with the `source` command.

The following table shows configuration files for configuring a Hadoop cluster in pseudo-distributed mode:

File	Description
`hadoop-env.sh`	Configures the environment variable used by Hadoop
`core-site.xml`	Configures parameters for the whole Hadoop cluster
`hdfs-site.xml`	Configures parameters for HDFS and its clients
`mapred-site.xml`	Configures parameters for MapReduce and its clients
`masters`	Configures host machines for SecondaryNameNode
`slaves`	Configures a list of slave node hosts

The following list explains the configuration files:

- `hadoop-env.sh` specifies environment variables for running Hadoop. For example, the home directory of Java installation `JAVA_HOME`, those related to Hadoop runtime options and cluster logging, and so on.
- `core-site.xml` specifies the URI of HDFS NameNode and MapReduce JobTracker. The `hdfs://localhost:54310` value of the `fs.default.name` property specifies the location of the default filesystem as HDFS on `localhost` using port `54310`. We can specify other filesystem schemes such as a local filesystem with `file:///home/hduser/hadoop`, Amazon web service S3 with `s3://a-bucket/hadoop`, and so on. The `localhost:54311` value of the `mapred.job.tracker` property specifies the URI of the cluster's JobTracker.
- `hdfs-site.xml` specifies the HDFS-related configurations. For example, `dfs.replication` configures the replication factor of data blocks on HDFS. For example, the value `2` specifies that each data block will be replicated twice on the filesystem. The `dfs.data.dir` property specifies the location of the data directory on the host Linux filesystem.
- `mapred-site.xml` specifies configurations for the MapReduce framework. For example, we can configure the total number of the `jvm` tasks, the number of the `map` slots, and reduce slots on a slave node, reduce the amount of memory for each task, and so on.
- The `masters` file specifies hosts that will run a SecondaryNameNode daemon. In our single node configuration, we put `localhost` in this file. A SecondaryNameNode daemon will be started on `localhost`, which has been verified with the `jps` command.
- The `slaves` file specifies slave nodes that run tasks controlled by task trackers. In our pseudo-distributed mode configuration, `localhost` is the only slave node in the cluster.

Hadoop provides a number of bash scripts for convenience of starting and stopping a cluster. The following table shows these scripts:

Script	Description
`start-dfs.sh`	This is the script to start HDFS daemons, including NameNode, SecondaryNameNode, and DataNode. A PID file will be created for each daemon process under the default folder `${hadoop.tmp.dir}`. For example, if the user `hduser` is used to run the script, the `/hadoop/tmp/hadoop-hduser-namenode.pid` file will be created for the NameNode daemon process.
`stop-dfs.sh`	This is the script to stop HDFS daemons. This command will try to find the PID files of the HDFS daemons, and kill the processes with the PID files. So, if the PID file is missing, this script will not work.
`start-mapred.sh`	This is the script to start MapReduce daemons, including the JobTracker and TaskTracker daemons. Similar to `start-hdfs.sh` script, PID files will be created for each daemon process.
`stop-mapred.sh`	This is the script to stop Hadoop MapReduce daemons. Similar to `stop-dfs.sh` script, the script will try to find the PID files and then kill those processes.
`start-all.sh`	It is equal to `start-dfs.sh` plus `start-mapred.sh`.
`stop-all.sh`	It is equal to `stop-dfs.sh` plus `stop-mapred.sh`.

There's more...

Currently, Hadoop is also available in the `rpm` format. So, we can use the following command to install Hadoop:

```
sudo rpm -ivh http://www.poolsaboveground.com/apache/hadoop/common/
stable/hadoop-1.1.2-1.x86_64.rpm
```

The locations of installed files will be different from the tarball method, and we can check the file layout with the following command:

```
rpm -ql hadoop
```

Then we can use the following command to configure a Hadoop cluster in single node:

```
sudo hadoop-setup-single-node.sh
```

See also

- The *Configuring Hadoop in fully distributed mode* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- The *Validating Hadoop installation* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*

Configuring Hadoop in fully-distributed mode

To configure a Hadoop cluster in **fully-distributed mode**, we need to configure all the master and slave machines. Although different from the pseudo-distributed mode, the configuration experience will be similar. In this recipe, we will outline steps to configure Hadoop in fully-distributed mode.

Getting ready

In this book, we propose to configure a Hadoop cluster with one master node and five slave nodes. The hostname of the `master` node is `1` and the hostnames of the slave nodes are `slave1`, `slave2`, `slave3`, `slave4`, and `slave5`.

Before getting started, we assume that Linux has been installed on all the cluster nodes and we should validate password-less login with the following commands on the master node:

```
ssh hduser@slave1
ssh hduser@slave2
...
```

Unlike the pseudo-distributed mode, configuring a Hadoop cluster in fully-distributed mode requires the successful configuration of all the nodes in the cluster. Otherwise, the cluster will not work as expected.

We should be cautious about the interconnection of the cluster nodes. Connection problems might be caused by configurations of firewalls, network, and so on.

Assuming the `$HADOOP_HOME/conf/slaves` file contains hostnames of the slave nodes, we can use the following command to check the password-less login to all slave nodes from the master node:

```
for host in 'cat $HADOOP_HOME/conf/slaves'; do
  echo 'Testing ssh from master to node ' $host
  ssh hduser@$host
done
```

How to do it...

Use the following recipe to configure Hadoop in fully-distributed mode:

1. Log in to the master node from the administrator machine with the following command:

   ```
   ssh hduser@master
   ```

2. Copy the Hadoop archive to the `/usr/local` directory:

```
sudo cp hadoop-1.1.2.tar.gz /usr/local
```

3. Decompress the Hadoop archive:

```
cd /usr/local
sudo tar xvf hadoop-1.1.2.tar.gz
```

4. Make a proper soft link for Hadoop root directory:

```
sudo ln -s hadoop-1.1.2 hadoop
```

5. Use your favorite text editor to open the `~/.bashrc` file, and add the following content:

```
export JAVA_HOME=/usr/java/latest
export HADOOP_HOME=/usr/local/Hadoop
export PATH=$PATH:$JAVA_HOME/bin:HADOOP_HOME/bin
```

6. Open the `$HADOOP_HOME/conf/hadoop-env.sh` file with your favorite text editor and add the following content:

```
export JAVA_HOME=/usr/java/latest
```

7. Open the `$HADOOP_HOME/conf/core-site.xml` file with your favorite text editor and add the following content:

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://master:54310</value>
  </property>

  <property>
    <name>mapred.job.tracker</name>
    <value>master:54311</value>
  </property>
</configuration>
```

8. Open the `$HADOOP_HOME/conf/hdfs-site.xml` file with your favorite text editor and add the following content into the file:

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

  <property>
    <name>dfs.data.dir</name>
```

```
    <value>/hadoop/data/</value>
  </property>

  <property>
    <name>hadoop.tmp.dir</name>
    <value>/hadoop/tmp/</value>
  </property>
</configuration>
```

9. Open the `$HADOOP_HOME/conf/mapred-site.xml` file with your favorite text editor and add the following content:

```
<configuration>
  <property>
    <name>mapred.tasktracker.map.tasks.maximum</name>
    <value>6</value>
  </property>

  <property>
    <name>mapred.tasktracker.reduce.tasks.maximum</name>
    <value>6</value>
  </property>

  <property>
    <name>mapred.map.child.java.opts</name>
    <value>-Xmx512m</value>
  </property>

  <property>
    <name>mapred.reduce.child.java.opts</name>
    <value>-Xmx512m</value>
  </property>

</configuration>
```

10. Configure the `$HADOOP_HOME/conf/masters` file with the following command:

```
sudo echo "master" > $HADOOP_HOME/conf/masters
```

    This will configure the master node to run SecondaryNameNode.

11. Open the `$HADOOP_HOME/conf/slaves` file with your favorite text editor and add all the slave node hostnames into the file similar to the following:

```
slave1
slave2
slave3
...
```

12. Copy the configured Hadoop directory to all the slave nodes with the following command:

```
for host in 'cat $HADOOP_HOME/conf/slaves
  do
  echo 'Configuring hadoop on slave node ' $host
  sudo scp -r /usr/local/hadoop-1.1.2 hduser@$host:/usr/local/
  echo 'Making symbolic link for Hadoop home directory on host '
$host
  sudo ssh hduser@$host -C "ln -s /usr/local/hadoop-1.1.2 /usr/
local/hadoop"
done
```

The `for-loop` command will recursively copy the `/usr/local/hadoop-1.1.2` directory to each node specified in the `$HADOOP_HOME/conf/slaves` file, and a symbolic link is made on each node for the Hadoop directory. We can get the following output information:

```
Configuring hadoop on slave node slave1
Making symbolic link for Hadoop home directory on host
host slave1
Configuring hadoop on slave node slave2
Making symbolic link for Hadoop home directory on host
host slave2
Configuring hadoop on slave node slave3
Making symbolic link for Hadoop home directory on host
host slave3
Configuring hadoop on slave node slave4
Making symbolic link for Hadoop home directory on host
host slave4
...
```

13. Copy the bash configuration file to each slave node with the following command:

```
for host in cat $HADOOP_HOME/conf/slaves; do
  echo 'Copying local bash run configuration file to host ' $host
  sudo cp ~/.bashrc $host:~/
done
```

The `for-loop` command copies the bash run configuration file from the master node to all the slave nodes in the cluster. We can get the following output message:

```
Copying local bash run configuration file to host slave1
Copying local bash run configuration file to host slave2
Copying local bash run configuration file to host slave3
...
```

Use the following recipe to start a Hadoop cluster:

1. Format the HDFS filesystem on the master node with the following command:

   ```
   hadoop namenode -format
   ```

If this is the first time to format the HDFS, the command should finish automatically. If you are reformatting an existing filesystem, it will ask you for permission to format the filesystem. For example, the output information will contain a message similar to the following:

```
Re-format filesystem in /tmp/hadoop-shumin/dfs/name ?
(Y or N)
```

In such a case, we need to press *Y* to confirm the reformatting of the filesystem. Be cautious that all the data will be wiped out after you hit the *Enter* key.

2. Check the directory structure of the formatted NameNode with the following command:

   ```
   tree /hadoop/dfs/
   ```

   The output will be similar to the following:

   ```
   /hadoop/dfs/
   └── name
       ├── current
       │   ├── edits
       │   ├── fsimage
       │   ├── fstime
       │   └── VERSION
       └── image
           └── fsimage


   3 directories, 5 files
   ```

The tree listing shows the directory structure of a formatted HDFS filesystem which contains the filesystem `image` (in the `/hadoop/dfs/name/image` directory) and the current live `image` (mirrored to the `/hadoop/dfs/name/current` folder) in the main memory.

3. Start HDFS cluster daemons with the following command:

```
start-dfs.sh
```

We will get output similar to the following:

```
starting namenode, logging to /usr/local/hadoop/logs/hadoop-hdu
ser-namenode-master.out
slave1: starting datanode, logging to /usr/local/hadoop/logs/h
adoop-hduser-datanode-sslave1.out
slave2: starting datanode, logging to /usr/local/hadoop/logs/
hadoop-hduser-datanode-slave2.out
slave3: starting datanode, logging to /usr/local/hadoop/logs/
hadoop-hduser-datanode-slave3.out
slave4: starting datanode, logging to /usr/local/hadoop/logs/
hadoop-hduser-datanode-slave4.out
slave5: starting datanode, logging to /usr/local/hadoop/logs/
hadoop-hduser-datanode-slave5.out
master: starting secondarynamenode, logging to /usr/local/hadoop/
logs/hadoop-hduser-secondarynamenode-hadoop-master.out
```

The output message shows that NameNode and SecondaryNameNode daemons are started on the master node, and a DataNode daemon is started on each slave node.

4. Start the MapReduce cluster daemons with the following command:

```
start-mapred.sh
```

The output will be similar to the following:

```
starting jobtracker, logging to /usr/local/hadoop/logs/hadoop-
hduser-jobtracker-master.out
slave1: starting tasktracker, logging to /usr/local/Hadoop/logs/
hadoop-hduser-tasktracker-slave1.out
slave2: starting tasktracker, logging to /usr/local/Hadoop/logs/
hadoop-hduser-tasktracker-slave2.out
slave3: starting tasktracker, logging to /usr/local/Hadoop/logs/
hadoop-hduser-tasktracker-slave3.out
```

```
slave4: starting tasktracker, logging to /usr/local/Hadoop/logs/
hadoop-hduser-tasktracker-slave4.out
slave5: starting tasktracker, logging to /usr/local/Hadoop/logs/
hadoop-hduser-tasktracker-slave5.out
```

The output message shows that a JobTracker daemon is started on the master node and a TaskTracker daemon is started on each slave node.

5. On the master node, check the status of the Hadoop daemons with the following command:

   ```
   jps
   ```

   The output will be similar to the following:

   ```
   19512 NameNode
   19930 JobTracker
   19708 SecondaryNameNode
   20276 Jps
   ```

6. On a slave node, we can check the status of the daemon processes with the same command, and the output will be similar to the following:

   ```
   3949 Jps
   3639 TaskTracker
   3501 DataNode
   ```

The highlighted daemons in the previous two steps must be present. Otherwise there will be configuration problems. You can review the recipe *Validating Hadoop installation* for troubleshooting and debugging suggestions.

7. List all the available TaskTrackers with the following command:

   ```
   hadoop job -list-active-trackers
   ```

   The output message will be similar to the following:

   ```
   tracker_slave1:slave1/10.0.0.2:38615
   tracker_slave2:slave2/10.0.0.3:39618
   tracker_slave3:slave3/10.0.0.4:48228
   tracker_slave4:slave4/10.0.0.5:42954
   tracker_slave5:slave5/10.0.0.6:43858
   ```

8. Check the status of each node in the HDFS cluster with the following command:

```
hadoop dfsadmin -report
```

The output message will be similar to the following:

```
Configured Capacity: 13500319031296 (12.28 TB)
Present Capacity: 12015141961728 (10.93 TB)
DFS Remaining: 4067084627968 (3.7 TB)
DFS Used: 7948057333760 (7.23 TB)
DFS Used%: 66.15%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

-------------------------------------------------
Datanodes available: 5 (5 total, 0 dead)

Name: 192.168.1.14:50010
Decommission Status : Normal
Configured Capacity: 964306395136 (898.08 GB)
DFS Used: 590553788416 (550 GB)
Non DFS Used: 97300185088 (90.62 GB)
DFS Remaining: 276452421632(257.47 GB)
DFS Used%: 61.24%
DFS Remaining%: 28.67%
Last contact: Sat Feb 16 00:34:17 EST 2013

...

Name: 192.168.1.17:50010
Decommission Status : Normal
Configured Capacity: 964262363136 (898.04 GB)
DFS Used: 617057673216 (574.68 GB)
Non DFS Used: 81531011072 (75.93 GB)
DFS Remaining: 265673678848(247.43 GB)
DFS Used%: 63.99%
DFS Remaining%: 27.55%
Last contact: Sat Feb 16 00:34:15 EST 2013
```

The output shows that there are `5` DataNodes in the cluster, and the status of each DataNode such as capacity and percentage of usage is reported.

Use the following two steps to stop a running Hadoop cluster:

1. Stop the MapReduce daemons with the following command on the master node:

   ```
   stop-mapred.sh
   ```

   We will get an output message similar to the following:

   ```
   stopping jobtracker
   slave3: stopping tasktracker
   slave2: stopping tasktracker
   slave5: stopping tasktracker
   slave4: stopping tasktracker
   slave1: stopping tasktracker
   ```

The output shows that the JobTracker daemon on the master node and TaskTracker daemons on the slave nodes are being stopped.

2. Stop the HDFS daemons with the following command on the master node:

   ```
   stop-dfs.sh
   ```

   The output message will be similar to the following:

   ```
   stopping namenode
   slave3: stopping datanode
   slave4: stopping datanode
   slave2: stopping datanode
   slave1: stopping datanode
   slave5: stopping datanode
   localhost: stopping secondarynamenode
   ```

The output shows that the NameNode and SecondaryNameNode daemons on the master node and the DataNode daemons on the slave nodes are being stopped.

How it works...

The following table shows the properties used in this recipe:

Property	Description
`fs.default.name`	The URI of the default filesystem.
`mapred.job.tracker`	The URI of the JobTracker, for example, `localhost:54310`.
`dfs.replication`	Specifies how many nodes a block should be replicated to. The default value of this property is `3`.
`dfs.data.dir`	The local storage directory of data blocks on DataNodes.
`hadoop.tmp.dir`	A base directory for a number of other directories.
`mapred.tasktracker.map.tasks.maximum`	Max number of parallel map tasks that a TaskTracker daemon can run.
`mapred.tasktracker.reduce.tasks.maximum`	Max number of parallel reduce tasks that a TaskTracker daemon can run.
`mapred.map.child.java.opts`	The Java options for the `map` task child processes.
`mapred.reduce.child.java.opts`	The Java options for the reduce task child processes.

There's more...

Alternatively, we can use the following steps to configure a fully-distributed Hadoop cluster:

1. Download Hadoop `rpm` package on the administrator machine with the following command:

   ```
   wget http://www.poolsaboveground.com/apache/hadoop/common/stable/hadoop-1.1.2-1.x86_64.rpm -P ~/repo
   ```

2. Log in to the master node with the following command:

   ```
   ssh hduser@master
   ```

3. Use the following commands to install Hadoop on all nodes:

```
for host in master slave1 slave2 slave3 slave4 slave5; do
  echo 'Installing Hadoop on node: ' $host
  sudo rpm -ivh ftp://hadoop.admin/repo/hadoop-1.1.2-1.x86_64.rpm
done
```

4. Configure the Hadoop cluster by modifying the configuration files located in the `/etc/hadoop` folder.

See also

- The *Configuring Hadoop in pseudo-distributed mode* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- The *Validating Hadoop installation* recipe in *Chapter 3*, *Configuring a Hadoop cluster*

Validating Hadoop installation

The configuration of a Hadoop cluster is not done before the validation step. Validation plays an important role in the configuration of a Hadoop cluster; for example, it can help us figure out configuration problems.

The most straightforward way to validate a Hadoop cluster configuration is to run a MapReduce job from the master node. Alternatively, there are two methods to validate the cluster configuration. One is from web interface and the other is from the command line. In this recipe, we will list steps to validate the configuration of a Hadoop cluster.

Getting ready

To validate the configuration from the web interface, a web browser such as Firefox or Google Chrome is needed. Sometimes if a GUI web browser is not available, we can use a command line based web browser such as `elinks` and `lynx`. In this book, we assume to use `elinks` for illustration purpose.

We assume that `elinks` has been installed with the following command:

```
sudo yum install elinks
```

Start all the Hadoop daemons with the following commands:

```
start-dfs.sh
start-mapred.sh
```

How to do it...

Use the following steps to run a MapReduce job:

1. Log in to the master node with the following command:

   ```
   ssh hduser@master
   ```

2. Run a sample MapReduce job with the following command:

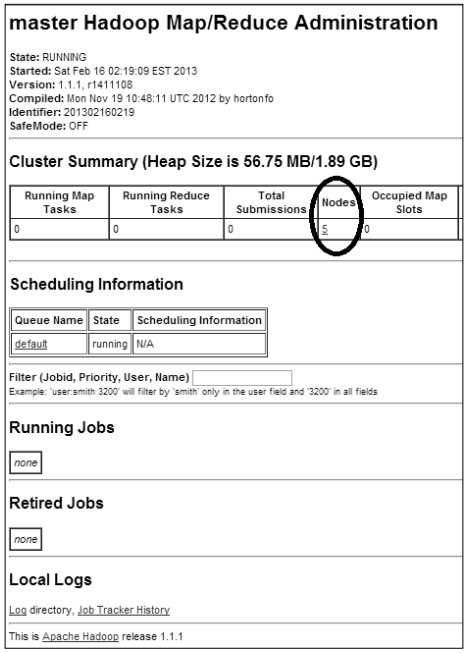
   ```
   hadoop jar $HADOOP_HOME/hadoop-examples*.jar pi 20 100000
   ```

In this command, `hadoop-examples*jar` is a `.jar` file that contains a few sample MapReduce jobs such as `pi`. Option `20` is the number of tasks to run and `100000` specifies the size of the sample for each task.

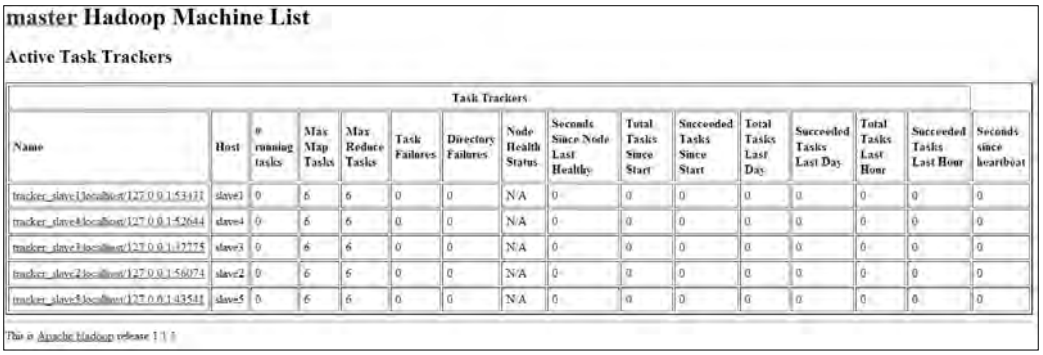
If this job finishes without any problem, we can say that the Hadoop cluster is working. But this is not enough, because we also need to make sure all the slave nodes are available for running tasks.

Use the following steps to validate Hadoop cluster configuration through a web user interface:

1. Open the `master:50030/jobtracker.jsp` URL with a web browser. The web page will be similar to the following screenshot:



2. Check the status of each slave node by clicking on the link, which leads us to a web page similar to the following screenshot:



master Hadoop Machine List

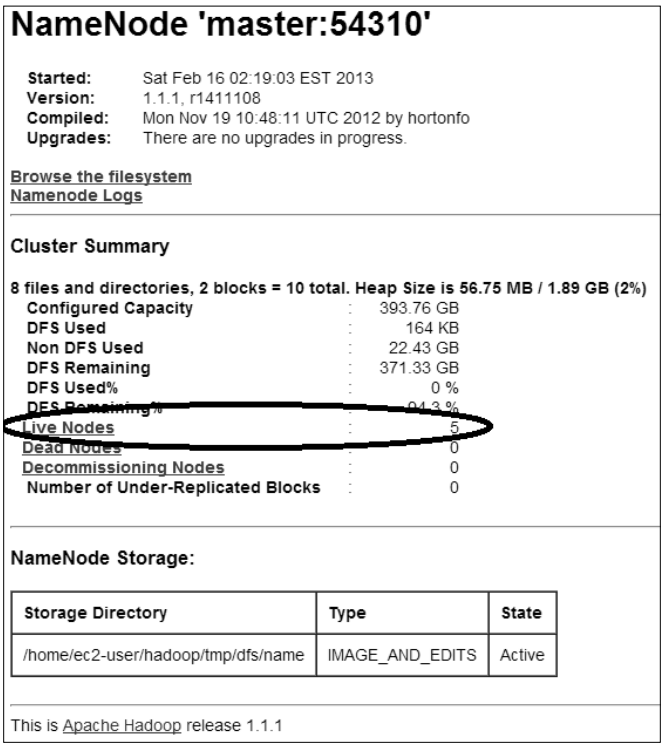
Active Task Trackers

Task Trackers															
Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day	Total Tasks Last Hour	Succeeded Tasks Last Hour	Seconds since heartbeat
tracker_slave1:localhost/127.0.0.1:53411	slave1	0	6	6	0	0	N/A	0	0	0	0	0	0	0	0
tracker_slave4:localhost/127.0.0.1:52644	slave4	0	6	6	0	0	N/A	0	0	0	0	0	0	0	0
tracker_slave3:localhost/127.0.0.1:37775	slave3	0	6	6	0	0	N/A	0	0	0	0	0	0	0	0
tracker_slave2:localhost/127.0.0.1:56074	slave2	0	6	6	0	0	N/A	0	0	0	0	0	0	0	0
tracker_slave5:localhost/127.0.0.1:43541	slave5	0	6	6	0	0	N/A	0	0	0	0	0	0	0	0

This is Apache Hadoop release 1.1.1

From this screenshot, we can easily check the status of the active TaskTrackers on the slave nodes. For example, we can see the count of failed tasks, the number of MapReduce slots, the heart beat seconds, and so on.

3. Check the status of slave DataNodes by opening the `master:50070` URL. The web page will be similar to the following screenshot:



NameNode 'master:54310'

Started: Sat Feb 16 02:19:03 EST 2013
Version: 1.1.1, r1411108
Compiled: Mon Nov 19 10:48:11 UTC 2012 by hortonfo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

8 files and directories, 2 blocks = 10 total. Heap Size is 56.75 MB / 1.89 GB (2%)

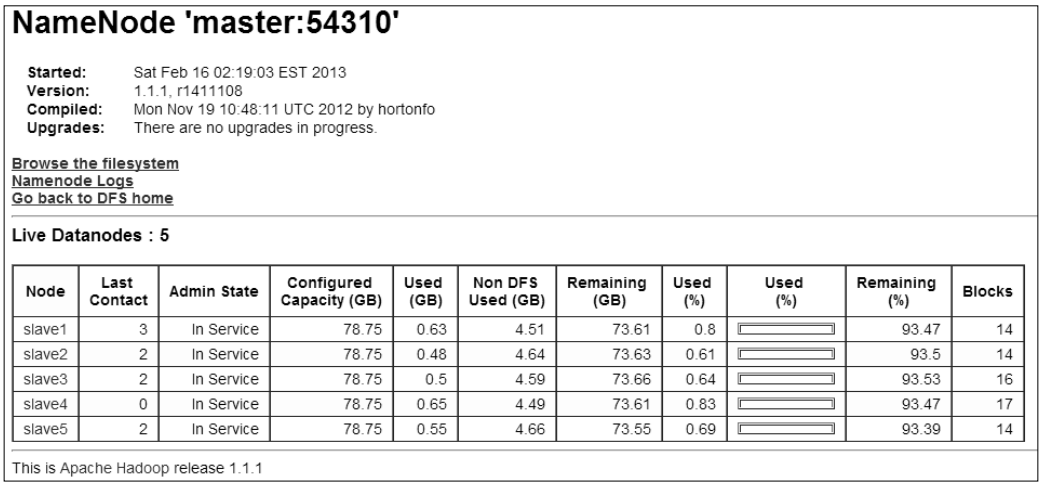
Configured Capacity	:	393.76 GB
DFS Used	:	164 KB
Non DFS Used	:	22.43 GB
DFS Remaining	:	371.33 GB
DFS Used%	:	0 %
DFS Remaining%	:	94.3 %
Live Nodes	:	5
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

NameNode Storage:

Storage Directory	Type	State
/home/ec2-user/hadoop/tmp/dfs/name	IMAGE_AND_EDITS	Active

This is Apache Hadoop release 1.1.1

4. By clicking on the **Live Nodes** link we can see the details of each node as shown in the following screenshot:



NameNode 'master:54310'

Started: Sat Feb 16 02:19:03 EST 2013
Version: 1.1.1, r1411108
Compiled: Mon Nov 19 10:48:11 UTC 2012 by hortonfo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 5

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
slave1	3	In Service	78.75	0.63	4.51	73.61	0.8		93.47	14
slave2	2	In Service	78.75	0.48	4.64	73.63	0.61		93.5	14
slave3	2	In Service	78.75	0.5	4.59	73.66	0.64		93.53	16
slave4	0	In Service	78.75	0.65	4.49	73.61	0.83		93.47	17
slave5	2	In Service	78.75	0.55	4.66	73.55	0.69		93.39	14

This is Apache Hadoop release 1.1.1

5. Run an example `teragen` job to generate 10 GB data on the HDFS with the following command:

```
hadoop jar $HADOOP_HOME/hadoop-examples-1.1.2.jar teragen
$((1024*1024*1024* 10/100)) teraout
```

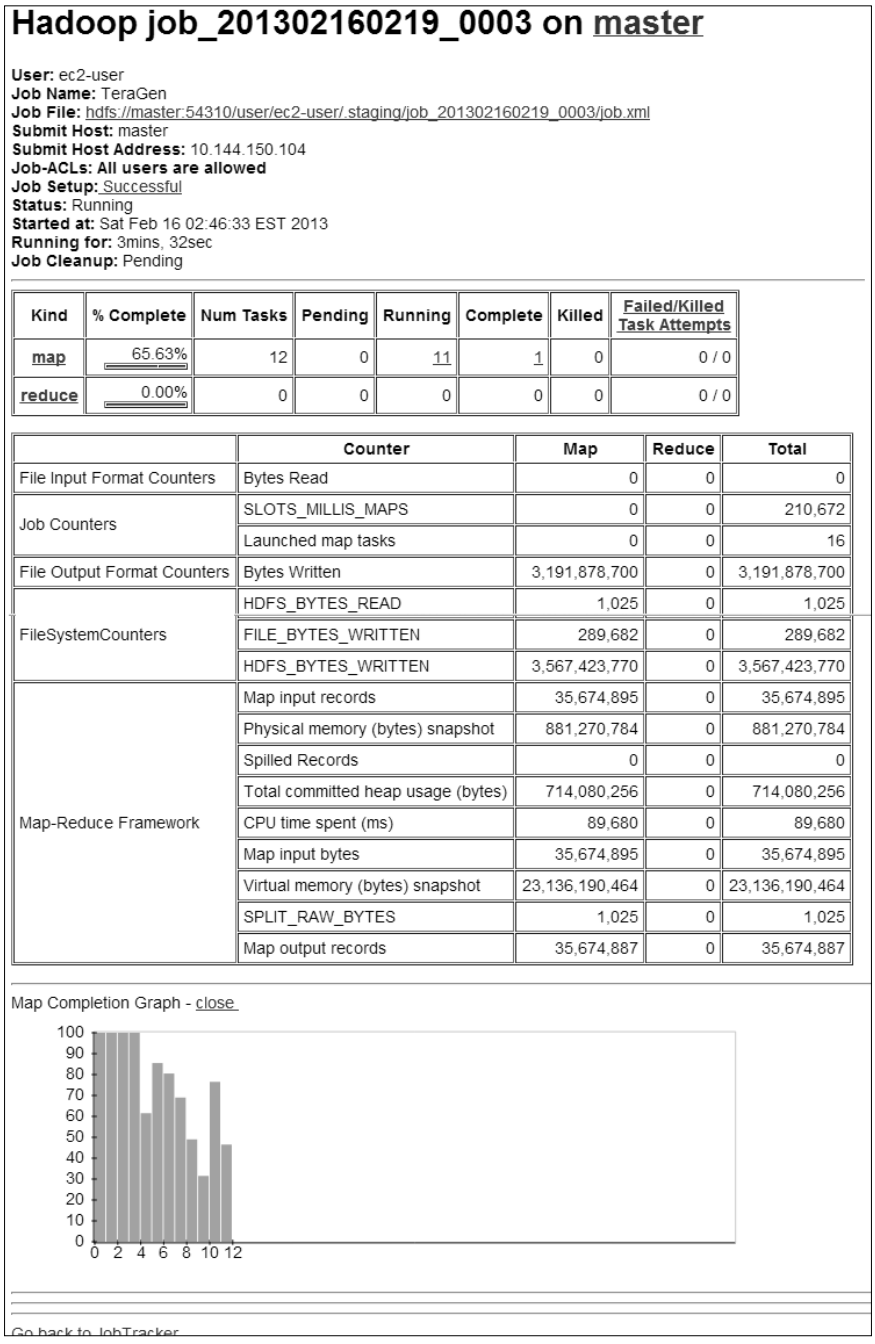
In this command, `hadoop-examples-1.1.2.jar` is the Java archive file which provides a number of Hadoop examples. The option `$((1024*1024*1024* 10/100))` tells us how many lines of data will be generated with the total data size 10 GB.

6. When the job is running, we can check the status of the job by opening the following URL:

```
http://master:50030/jobdetails.jsp?jobid=job_201302160219_0003&
refresh=30
```

In this URL, `job_201302160219_0003` is the job ID and `refresh=30` tells how often the web page should be refreshed.

The job status web page will be similar to the following screenshot:



Hadoop job_201302160219_0003 on master

User: ec2-user
Job Name: TeraGen
Job File: hdfs://master:54310/user/ec2-user/.staging/job_201302160219_0003/job.xml
Submit Host: master
Submit Host Address: 10.144.150.104
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Running
Started at: Sat Feb 16 02:46:33 EST 2013
Running for: 3mins, 32sec
Job Cleanup: Pending

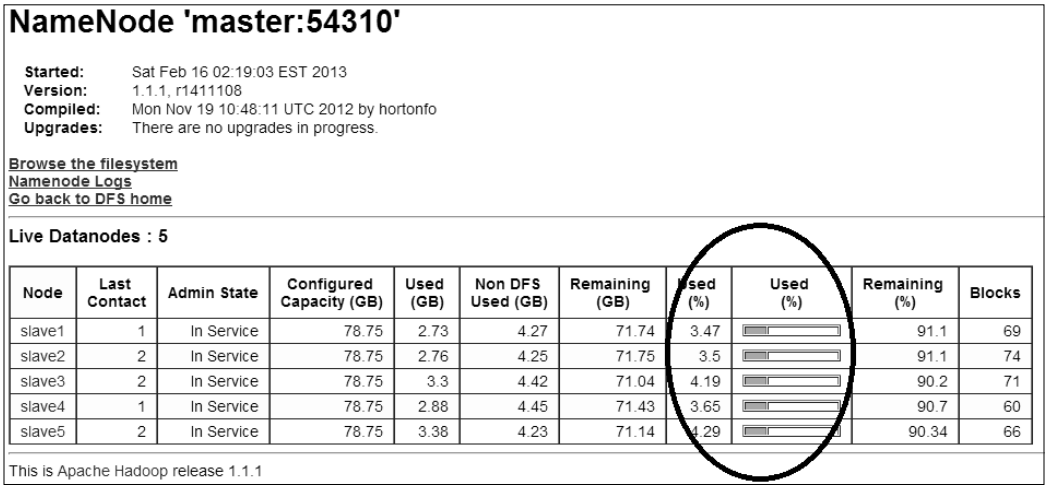
Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	65.63%	12	0	11	1	0	0 / 0
reduce	0.00%	0	0	0	0	0	0 / 0

	Counter	Map	Reduce	Total
File Input Format Counters	Bytes Read	0	0	0
Job Counters	SLOTS_MILLIS_MAPS	0	0	210,672
	Launched map tasks	0	0	16
File Output Format Counters	Bytes Written	3,191,878,700	0	3,191,878,700
FileSystemCounters	HDFS_BYTES_READ	1,025	0	1,025
	FILE_BYTES_WRITTEN	289,682	0	289,682
	HDFS_BYTES_WRITTEN	3,567,423,770	0	3,567,423,770
Map-Reduce Framework	Map input records	35,674,895	0	35,674,895
	Physical memory (bytes) snapshot	881,270,784	0	881,270,784
	Spilled Records	0	0	0
	Total committed heap usage (bytes)	714,080,256	0	714,080,256
	CPU time spent (ms)	89,680	0	89,680
	Map input bytes	35,674,895	0	35,674,895
	Virtual memory (bytes) snapshot	23,136,190,464	0	23,136,190,464
	SPLIT_RAW_BYTES	1,025	0	1,025
	Map output records	35,674,887	0	35,674,887

Map Completion Graph - close

Go back to JobTracker

7. After the `teragen` job finishes, we can check the node storage space usage by opening the URL `http://master:50070/dfsnodelist.jsp?whatNodes=LIVE`. The web page will be similar to the following screenshot:



NameNode 'master:54310'

Started: Sat Feb 16 02:19:03 EST 2013
Version: 1.1.1, r1411108
Compiled: Mon Nov 19 10:48:11 UTC 2012 by hortonfo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 5

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
slave1	1	In Service	78.75	2.73	4.27	71.74	3.47		91.1	69
slave2	2	In Service	78.75	2.76	4.25	71.75	3.5		91.1	74
slave3	2	In Service	78.75	3.3	4.42	71.04	4.19		90.2	71
slave4	1	In Service	78.75	2.88	4.45	71.43	3.65		90.7	60
slave5	2	In Service	78.75	3.38	4.23	71.14	4.29		90.34	66

This is Apache Hadoop release 1.1.1

Sometimes, a command-line based web browser can be handier than a GUI browser. For example, we can use the `elinks master:50030` command to check the status of MapReduce on the master node and use the `elinks master:50070` command to check the status of HDFS.

Use the following steps to validate the configuration of a Hadoop cluster from command line:

1. List all available TaskTrackers with the following command:

   ```
   hadoop job -list-active-trackers
   ```

   Example output is similar to the following:

   ```
   tracker_slave1:localhost/127.0.0.1:53431
   tracker_slave4:localhost/127.0.0.1:52644
   tracker_slave3:localhost/127.0.0.1:37775
   tracker_slave2:localhost/127.0.0.1:56074
   tracker_slave5:localhost/127.0.0.1:43541
   ```

   The output confirms that all the configured TaskTrackers are active in the Hadoop cluster.

2. Check the status of the HDFS cluster with the following command:

   ```
   hadoop fsck /
   ```

The output will be similar to the following:

```
FSCK started by hduser from /10.0.0.1 for path / at Sat Feb 16
03:03:44 EST 2013
.............................Status: HEALTHY
 Total size:    7516316665 B
 Total dirs:    15
 Total files:   31
 Total blocks (validated):      125 (avg. block size 60130533 B)
 Minimally replicated blocks:   125 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     2.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          5
 Number of racks:               1
FSCK ended at Sat Feb 16 03:03:44 EST 2013 in 12 milliseconds


The filesystem under path '/' is HEALTHY
```

The output gives us the same information as from the web interface, and the last line tells us that the `root` filesystem is `HEALTHY`.

How it works...

Hadoop provides commands and web interfaces for system administrators to check the status of the cluster. When we start Hadoop daemons, a built-in web server will be started and a number of prewritten `.jsp` script files are used to respond to the user's requests from a web browser. The `.jsp` files can be found under the `$HADOOP_HOME/webapps` directory. If you have programming experience, you can take advantage of the `.jsp` files to develop personalized Hadoop cluster management tools.

There's more...

In this part, we list a few typical Hadoop configuration problems and give suggestions on dealing with these problems.

Can't start HDFS daemons

There are many possible reasons that can cause this problem. For example, the NameNode on the master node has not been formatted, in which case, we can format the HDFS before starting the cluster with the following command:

```
hadoop namenode -format
```

Warning!

Be cautious when formatting the filesystem with this command. It will erase all the data on the filesystem. Always try other methods before using this one.

More generically, to troubleshoot this problem, we need to check that HDFS has been properly configured and daemons are running. This can be done with the following command:

```
jps
```

If the output of this command does not contain the NameNode and SecondaryNameNode daemons, we need to check the configuration of HDFS.

To troubleshoot the HDFS startup problem, we can open a new terminal and monitor the NameNode logfile on the master node with the following command:

```
tail -f $HADOOP_HOME/logs/hadoop-hduser-namenode-master.log
```

This command will show the content of the logfile in a dynamic way when a new log is appended to the file. If an error happens, we can get error messages similar to the following:

```
2013-02-16 11:44:29,860 ERROR org.apache.hadoop.hdfs.server.namenode.
NameNode: java.net.UnknownHostException: Invalid hostname for server:
master1
        at org.apache.hadoop.ipc.Server.bind(Server.java:236)
        at org.apache.hadoop.ipc.Server$Listener.<init>(Server.java:302)
        at org.apache.hadoop.ipc.Server.<init>(Server.java:1488)
        at org.apache.hadoop.ipc.RPC$Server.<init>(RPC.java:560)
        at org.apache.hadoop.ipc.RPC.getServer(RPC.java:521)
```

```
        at org.apache.hadoop.hdfs.server.namenode.NameNode.
initialize(NameNode.java:295)
        at org.apache.hadoop.hdfs.server.namenode.
NameNode.<init>(NameNode.java:529)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.
createNameNode(NameNode.java:1403)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.
java:1412)

2013-02-16 11:44:29,865 INFO org.apache.hadoop.hdfs.server.namenode.
NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at master/10.144.150.104
************************************************************/
```

Alternatively, the following command will give the same error:

```
hadoop jobtracker
```

The previous message shows that the hostname of the NameNode is wrong. It should be `master` instead of `master1`.

Cluster is missing slave nodes

Most likely, this problem is caused by hostname resolution. To confirm, we can check the content of the `/etc/hosts` file with the following command:

```
cat /etc/hosts
```

The output should be similar to the following:

```
10.0.0.1  master
10.0.0.2  slave1
10.0.0.3  slave2
10.0.0.4  slave3
10.0.0.5  slave4
10.0.0.6  slave5
```

If the IP address and hostname mapping does not exist or has been erroneously specified in this file, correcting the error can solve this problem.

MapReduce daemons can't be started

The following two reasons can cause this problem:

- The HDFS daemons are not running, which can cause the MapReduce daemons to ping the NameNode daemon at a regular interval, which can be illustrated with the following log output:

```
13/02/16 11:32:19 INFO ipc.Client: Retrying connect to server:
master/10.0.0.1:54310. Already tried 0           time(s); retry
policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
sleepTime=1 SECONDS)

13/02/16 11:32:20 INFO ipc.Client: Retrying connect to server:
master/10.0.0.1:54310. Already tried 1 time(s); retry policy is
RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1
SECONDS)

13/02/16 11:32:21 INFO ipc.Client: Retrying connect to server:
master/10.0.0.1:54310. Already tried 2 time(s); retry policy is
RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1
SECONDS)

13/02/16 11:32:22 INFO ipc.Client: Retrying connect to server:
master/10.0.0.1:54310. Already tried 3 time(s); retry policy is
RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1
SECONDS)

13/02/16 11:32:23 INFO ipc.Client: Retrying connect to server:
master/10.0.0.1:54310. Already tried 4 time(s); retry policy is
RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1
SECONDS).
```

To troubleshoot this problem, we can refer to tips from the *Can't start HDFS daemons* section.

- Configuration problems of MapReduce can cause the `MapReduce daemons can't be started` problem. Recall that we have configurations for the number of the `map` slots and reduce slots as well as the amount of memory in the `$HADOOP_HOME/conf/mapred-site.xml` file. Before starting a cluster, we need to make sure that the total amount of configured memory should be smaller than the total amount of system memory.

For example, suppose a slave host has 4 GB of memory, and we have configured six map slots, and six reduce slots with 512 MB of memory for each slot. So, we can compute the total configured task memory with the following formula:

6 x 512 + 6 x 512 = 6 GB

As 6 GB is larger than the system memory of 4 GB, the system will not start. To clear this problem, we can decrease the number of the `map` slots and reduce slots from six to three. This configuration gives us a total configured memory of 3 GB, which is smaller than the system total memory 4 GB, thus the MapReduce daemons should be able to start successfully.

See also

- The *Configuring Hadoop in pseudo-distributed mode* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- The *Configuring Hadoop in fully-distributed mode* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*

Configuring ZooKeeper

ZooKeeper provides highly reliable centralized service for maintaining configuration information, naming, and providing distributed synchronization and group services. In this recipe, we will outline steps to install ZooKeeper.

Getting ready

Make sure Hadoop has been properly configured. Please refer to the previous recipes in this chapter about installation of Hadoop on a cluster.

Log in to the master node from the Hadoop administrator machine as `hduser` with the following command:

```
ssh hduser@master
```

Download the ZooKeeper archive file with the following commands:

```
wget http://www.gtlib.gatech.edu/pub/apache/zookeeper/stable/zookeeper-
3.4.5.tar.gz -P ~/repo
```

How to do it...

Use the following steps to configure ZooKeeper:

1. Log in to the master node with the following command:

   ```
   ssh hduser@master
   ```

2. Copy the downloaded archive to `/usr/local` with the following command:

   ```
   sudo wget ftp://hadoop.admin/repo/zookeeper-3.4.5.tar.gz -P /usr/
   local
   ```

3. Decompress the file with the following command:

   ```
   cd /usr/local/
   sudo tar xvf zookeeper-3.4.5.tar.gz
   ```

4. Create a symbolic link with the following command:

   ```
   sudo ln -s /usr/local/zookeeper-3.4.5 /usr/local/zookeeper
   ```

5. Open the `~/.bashrc` file and add the following lines:

   ```
   ZK_HOME=/usr/local/zookeeper
   export PATH=$ZK_HOME/bin:$PATH
   ```

6. Load the configuration file with the following command:

   ```
   . ~/.bashrc
   ```

7. Create data and log directories for ZooKeeper with the following command:

   ```
   sudo mkdir -pv /hadoop/zookeeper/{data,log}
   ```

8. Create Java configuration file `$ZK_HOME/conf/java.env` with the following content:

   ```
   JAVA_HOME=/usr/java/latest
   export PATH=$JAVA_HOME/bin:$PATH
   ```

The filename `java.env` is mandatory. It will be loaded by ZooKeeper.

9. Create the `$ZK_HOME/conf/zookeeper.cfg` file and add the following lines to it:

   ```
   tickTime=2000
   clientPort=2181
   initLimit=5
   ```

```
syncLimit=2
server.1=master:2888:3888
server.2=slave1:2888:3888
server.3=slave2:2888:3888
server.4=slave3:2888:3888
server.5=slave4:2888:3888
server.6=slave5:2888:3888
dataDir=/hadoop/zookeeper/data
dataLogDir=/hadoop/zookeeper/log
```

The highlighted section makes every node know the other nodes in the ZooKeeper ensemble.

10. Configure ZooKeeper on all slave nodes with the following command:

```
for host in cat $HADOOP_HOME/conf/slaves; do
  echo 'Configuring ZooKeeper on ' $host
  scp ~/.bashrc hduser@$host:~/
  sudo scp -r /usr/local/zookeeper-3.4.5 hduser@$host:/usr/local/
  echo 'Making symbolic link for ZooKeeper home directory on '
$host
  sudo ssh hduser@$host -C "ln -s /usr/local/zookeeper-3.4.5 /usr/
local/zookeeper"
done
```

11. Start ZooKeeper on master node with the following command:

```
zkServer.sh start
```

12. Verify ZooKeeper configuration with the following command:

```
zkCli.sh -server master:2181
```

13. Stop ZooKeeper with the following command:

```
zkServer.sh stop
```

See also

- The *Installing HBase* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- Get more documentation about ZooKeeper from `http://zookeeper.apache.org/doc/r3.4.5/zookeeperAdmin.html`

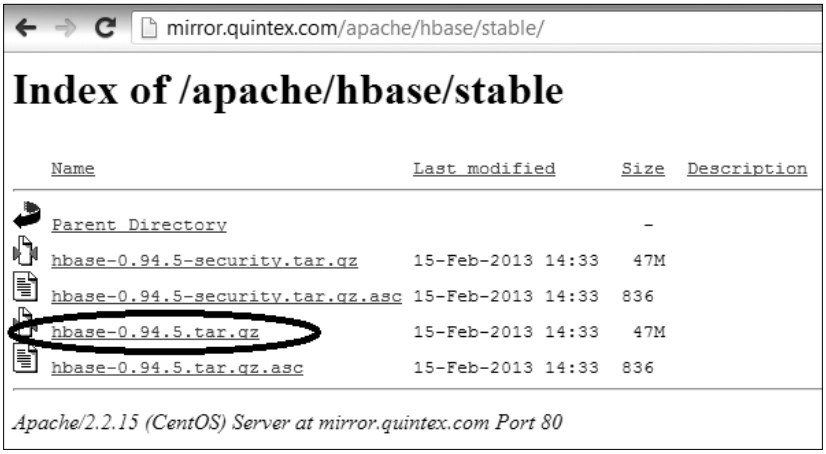
Installing HBase

HBase is the database based on Hadoop. It is a distributed, scalable Big Data storage system. In this section, we are going to list steps about installing HBase in our Hadoop cluster.

Getting ready

To install HBase, we assume that Hadoop has been configured without any issues.

Download HBase from a mirror site. Similar to downloading Hadoop, HBase is hosted on mirrors all over the world. Visit the link `http://www.apache.org/dyn/closer.cgi/hbase/`, and select the nearest mirror (the suggested mirror on the top is the optimal choice). After selecting the mirror, follow the link to select the HBase version; we suggest the stable version. For example, follow the link `http://mirror.quintex.com/apache/hbase/stable/` and you can see the downloadable files as shown in the following screenshot:



Click on the file link `hbase-0.94.5.tar.gz` to download the file to the administrator machine. Then, copy the file to the FTP repository with the following command:

```
cp hbase-0.94.5.tar.gz ~/repo
```

Alternatively, we can download the file with the following command:

```
wget http://mirror.quintex.com/apache/hbase/stable/hbase-0.94.5.tar.gz -P ~/repo
```

How to do it...

Use the following recipe to install HBase:

1. Log in to the master node from the administrator machine with the following command:

   ```
   ssh hduser@master
   ```

2. Decompress the HBase archive with the following commands:

   ```
   cd /usr/local
   sudo wget ftp://hadoop.admin/repo/hbase-0.94.5.tar.gz -P /usr/
   local
   sudo tar xvf hbase-0.94.5.tar.gz
   ```

3. Create a symbolic link with the following command:

   ```
   ln -s hbase-0.94.5 hbase
   ```

4. Use your favorite text editor to open the `~/.bashrc` file and append the following lines into the file:

   ```
   export HBASE_HOME=/usr/local/hbase
   export PATH=$HBASE_HOME/bin:$PATH
   ```

5. Open the `$HBASE_HOME/conf/hbase-env.sh` file and set `JAVA_HOME` as:

   ```
   export JAVA_HOME=/usr/java/latest
   ```

6. Open the `$HBASE_HOME/conf/hbase-site.xml` file with your favorite text editor and add the following contents to the file:

   ```
   <?xml version="1.0"?>
   <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
   <configuration>
     <property>
       <name>hbase.rootdir</name>
       <value>hdfs://master:54310/hbase</value>
     </property>

     <property>
       <name>hbase.cluster.distributed</name>
       <value>true</value>
     </property>

     <property>
       <name>hbase.tmp.dir</name>
       <value>/hadoop/hbase</value>
     </property>
   ```

```
  <property>
    <name>hbase.ZooKeeper.quorum</name>
    <value>master</value>
  </property>

  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/hadoop/zookeeper</value>
  </property>
</configuration>
```

7. Open the `$HBASE_HOME/conf/regionservers` file and add the following lines:

```
slave1
slave2
slave3
slave4
slave5
```

8. Link the HDFS configuration file to the HBase configuration directory with the following command:

```
sudo ln -s $HADOOP_HOME/conf/hdfs-site.xml $HBASE_HOME/conf/hdfs-site.xml
```

9. Replace the dependent `.jar` files for HBase with the following commands:

```
rm -i $HBASE_HOME/lib/hadoop-core*.jar $HBASE_HOME/lib/zookeeper-*.jar

cp -i $HADOOP_HOME/hadoop-core*.jar $HADOOP_HOME/lib/commons-*.jar $ZK_HOME/zookeeper-*.jar $HBASE_HOME/lib/
```

10. Configure all the slave nodes with the following commands:

```
for host in 'cat $HBASE_HOME/conf/regionservers'; do
  echo 'Configuring HBase on ' $host
  scp ~/.bashrc hduser@$host:~/
  sudo scp -r /usr/local/hbase-0.94.5 hduser@$host:/usr/local/
  echo 'Making symbolic link for HBase home directory on ' $host
  sudo ssh hduser@$host -C "ln -s /usr/local/hbase-0.94.5 /usr/local/hbase"
  echo 'Making symbolic link for hdfs-site.xml to the HBase configuration directory on ' $host
  sudo ssh hduser@$host -C "ln -s /usr/local/hadoop-1.1.2/conf/hdfs-site.xml /usr/local/hbase-0.94.5/conf/hdfs-site.xml"
done
```

11. Start HBase daemons with the following command:

```
start-hbase.sh
```

12. Connect to the running HBase with the following command:

```
hbase shell
```

13. Verify the HBase installation with the following HBase shell commands:

```
hbase(main):001:0> create 'test', 'c'
0 row(s) in 0.2410 seconds

hbase(main):001:0> put 'test', 'r1', 'c:a', 'v1'
0 row(s) in 0.0320 seconds

hbase(main):003:0> scan 'test'
ROW COLUMN+CELL row1 column=c:a, timestamp=124455459102, value=v1
r1
1 row(s) in 0.2130 seconds

hbase(main):006:0> disable 'test'
0 row(s) in 9.4210 seconds

hbase(main):007:0> drop 'test'
0 row(s) in 8.3412 seconds

hbase(main):010:0> exit
```

14. To stop HBase, use the following command:

```
stop-hbase.sh
```

    The following message will be given:

```
stopping hbase..............
```

How it works...

In the configuration, the `hbase.rootdir` property specifies the root directory of the HBase data storage, and the `hbase.zookeeper.property.dataDir` property specifies the root directory of the ZooKeeper data storage.

See also

- The *Installing ZooKeeper* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- More documentation about HBase can be found at: `http://wiki.apache.org/hadoop/Hbase`

Installing Hive

As a top-level abstraction language, **Hive** provides a handy tool for manipulating data storage on HDFS with SQL-like language. In this section, we will talk about installing Hive on our Hadoop cluster.

Getting ready

Before we install Hive, we need to make sure Hadoop has been properly installed. Please refer to the previous sections about the configuration of a Hadoop cluster.

Download Hive from a mirror site with a command similar to the following on the administrator machine:

```
wget http://apache.osuosl.org/hive/stable/hive-0.9.0.tar.gz -P ~/repo
```

How to do it...

Use the following steps to install Hive:

1. Log in to the master node from the Hadoop administrator machine as `hduser` with the following command:

   ```
   ssh hduser@master
   ```

2. Copy the archive to `/usr/local` with the following command:

   ```
   sudo wget ftp://hadoop.admin/repo/hive-0.9.0.tar.gz /usr/local
   ```

3. Decompress the Hive archive with the following command:

   ```
   cd /usr/local
   tar xvf hive-0.9.0.tar.gz
   ```

4. Create a symbolic link with the following command:

   ```
   ln -s /usr/local/hive-0.9.0 /usr/local/hive
   ```

5. Use your favorite text editor to open the `~/.bashrc` file and add the following lines to this file:

```
export HIVE_HOME=/usr/local/hive
export PATH=$HIVE_HOME/bin:$PATH
```

6. Start Hive with the following command:

```
hive
```

See also

- The *Installing Pig* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- Get more documentation about Hive from `https://cwiki.apache.org/confluence/display/Hive/Home`

Installing Pig

Similar to Hive, **Pig** provides a handy tool for manipulating Hadoop data. In this recipe, we are going to discuss the installation of Apache Pig.

Getting ready

Before we install Pig, we need to make sure Hadoop has been properly installed. Please refer to the previous sections about the configuration of a Hadoop cluster.

Download the Pig archive file from a mirror site with the following command on the administrator machine:

```
wget http://www.motorlogy.com/apache/pig/stable/pig-0.10.1.tar.gz
~/repo
```

How to do it...

Use the following steps to configure Pig:

1. Log in to the master node from the Hadoop administrator machine as `hduser` with the following command:

```
ssh hduser@master
```

2. Copy the archive to `/usr/local` with the following command:

```
sudo wget ftp://hadoop.admin/repo/pig-0.10.1.tar.gz /usr/local
```

3. Decompress the Pig archive file with the following command:

```
cd /usr/local
sudo tar xvf pig-0.10.1.tar.gz
```

4. Create a symbolic link to the Pig directory using the following command:

```
sudo ln -s /usr/local/pig-0.10.1 /usr/local/pig
```

5. Open the `~/.bashrc` file with your favorite text editor and add the following lines into the file:

```
export PIG_HOME=/usr/local/pig
export PATH=$PIG_HOME/bin:$PATH
```

6. Run Pig in local mode with the following command:

```
pig -x local
```

7. Run Pig in MapReduce mode with the following command:

```
pig
```

8. Alternatively, we can use the following command:

```
pig -x mapreduce
```

Pig that runs in MapReduce mode will utilizes the power of distributed computing provided by Hadoop.

See also

- The *Installing Hive* recipe in *Chapter 3*, *Configuring a Hadoop Cluster*
- More documentation about Pig can be obtained from: `http://pig.apache.org/docs/r0.10.0/`

Installing Mahout

Apache **Mahout** is a machine learning library that scales machine learning algorithms on Big Data. It is implemented on top of the Hadoop Big Data stack. It already implements a wide range of machine learning algorithms. In this recipe, we will outline steps to configure Apache Mahout.

Getting ready

Before we install Mahout, we need to make sure Hadoop has been properly installed.

Download Mahout from the mirror site with the following command on the master node:

```
wget http://www.eng.lsu.edu/mirrors/apache/mahout/0.7/mahout-
distribution-0.7.tar.gz -P ~/repo
```

How to do it...

Use the following recipe to install Mahout:

1. Log in to the master node from the Hadoop administrator machine as `hduser` with the following command:

   ```
   ssh hduser@master
   ```

2. Copy the archive to `/usr/local` with the following command:

   ```
   sudo wget ftp://hadoop.admin/repo/mahout-distribution-0.7.tar.gz /
   usr/local
   ```

3. Decompress the Mahout archive with the following commands:

   ```
   cd /usr/local
   sudo tar xvf mahout-distribution-0.7.tar.gz
   ```

4. Create a symbolic link to the Mahout directory with the following command:

   ```
   sudo ln -s /usr/local/mahout-distribution-0.7 /usr/local/mahout
   ```

5. Open the `~/.bashrc` file with your favorite text editor and add the following lines to the file:

   ```
   export MAHOUT_HOME=/usr/local/pig
   export PATH=$MAHOUT_HOME/bin:$PATH
   ```

6. Load the configuration with the following command:

   ```
   . ~/.bashrc
   ```

7. Install **Maven** with the following command:

   ```
   sudo yum install maven
   ```

8. Compile and install Mahout core with the following commands:

   ```
   cd $MAHOUT_HOME
   sudo mvn compile
   sudo mvn install
   ```

The `install` command will run all the tests by default; we can ignore the tests to speed up the installation process with command `sudo mvn -DskipTests install`.

9. Compile the Mahout examples with the following commands:

```
cd examples
sudo mvn compile
```

Use the following steps to verify Mahout configuration:

1. Download sample data with the following command:

```
wget http://archive.ics.uci.edu/ml/databases/synthetic_control/
synthetic_control.data -P ~/
```

2. Start the Hadoop cluster with commands:

```
start-dfs.sh
start-mapred.sh
```

3. Put the downloaded data into HDFS with the following commands:

```
hadoop fs -mkdir testdata
hadoop fs -put ~/synthetic_control.data testdata
```

4. Run the `kmeans` clustering with the following command:

```
mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job
```

See also

- More documentation about Mahout can be obtained from `https://cwiki.apache.org/confluence/display/MAHOUT/Mahout+Wiki`.

Where to buy this book

You can buy Hadoop Operations and Cluster Management Cookbook from the Packt Publishing website: `http://packtpub.com/hadoop-operations-and-cluster-management-cookbook/book`.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information:
www.packtpub.com/hadoop-operations-and-cluster-management-cookbook/book