Michael G. Noll

Applied Research. Big Data. Distributed Systems. Open Source.

- RSS

Enter your search... Search

- Blog
- Archive
- Tutorials
- Projects
- Publications

Running Hadoop on Ubuntu Linux (Single-Node Cluster)

Table of Contents

In this tutorial I will describe the required steps for setting up a *pseudo-distributed, single-node* Hadoop cluster backed by the Hadoop Distributed File System, running on Ubuntu Linux.

Are you looking for the multi-node cluster tutorial? Just head over there.

Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Hadoop’s HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications that have large data sets.

The main goal of this tutorial is to get a simple Hadoop installation up and running so that you can play around with the software and learn more about it.

This tutorial has been tested with the following software versions:

- Ubuntu Linux 10.04 LTS (deprecated: 8.10 LTS, 8.04, 7.10, 7.04)
- Hadoop 1.0.3, released May 2012

Figure 1: Cluster of machines running Hadoop at Yahoo! (Source: Yahoo!)

Prerequisites

Sun Java 6

Hadoop requires a working Java 1.5+ (aka Java 5) installation. However, using Java 1.6 (aka Java 6) is recommended for running Hadoop. For the sake of this tutorial, I will therefore describe the installation of Java 1.6.

Important Note: The apt instructions below are taken from this SuperUser.com thread. I got notified that the previous instructions that I provided no longer work. Please be aware that adding a third-party repository to your Ubuntu configuration is considered a security risk. If you do not want to proceed with the apt instructions below, feel free to install Sun JDK 6 via alternative means (e.g. by downloading the binary package from Oracle) and then continue with the next section in the tutorial.

```
# Add the Ferramosca Roberto's repository to your apt repositories
# See https://launchpad.net/~ferramroberto/
#
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:ferramroberto/java

# Update the source list
$ sudo apt-get update

# Install Sun Java 6 JDK
$ sudo apt-get install sun-java6-jdk

# Select Sun's Java as the default on your machine.
# See 'sudo update-alternatives --config java' for more information.
#
$ sudo update-java-alternatives -s java-6-sun
```

The full JDK which will be placed in `/usr/lib/jvm/java-6-sun` (well, this directory is actually a symlink on Ubuntu).

After installation, make a quick check whether Sun’s JDK is correctly set up:

```
user@ubuntu:~# java -version
java version "1.6.0_20"
Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
Java HotSpot(TM) Client VM (build 16.3-b01, mixed mode, sharing)
```

Adding a dedicated Hadoop system user

We will use a dedicated Hadoop user account for running Hadoop. While that’s not required it is recommended because it helps to separate the Hadoop installation from other software applications and user accounts running on the same machine (think: security, permissions, backups, etc).

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
```

This will add the user `hduser` and the group `hadoop` to your local machine.

Configuring SSH

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine if you want to use Hadoop on it (which is what we want to do in this short tutorial). For our single-node setup of Hadoop, we therefore need to configure SSH access to `localhost` for the `hduser` user we created in the previous section.

I assume that you have SSH up and running on your machine and configured it to allow SSH public key authentication. If not, there are several online guides available.

First, we have to generate an SSH key for the `hduser` user.

```
user@ubuntu:~$ su - hduser
hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
```

```
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
9b:82:ea:58:b4:e0:35:d7:ff:19:66:a6:ef:ae:0e:d2 hduser@ubuntu
The key's randomart image is:
[...snipp...]
hduser@ubuntu:~$
```

The second line will create an RSA key pair with an empty password. Generally, using an empty password is not recommended, but in this case it is needed to unlock the key without your interaction (you don't want to enter the passphrase every time Hadoop interacts with its nodes).

Second, you have to enable SSH access to your local machine with this newly created key.

```
hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The final step is to test the SSH setup by connecting to your local machine with the `hduser` user. The step is also needed to save your local machine's host key fingerprint to the `hduser` user's `known_hosts` file. If you have any special SSH configuration for your local machine like a non-standard SSH port, you can define host-specific SSH options in `$HOME/.ssh/config` (see `man ssh_config` for more information).

```
hduser@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686 GNU/Linux
Ubuntu 10.04 LTS
[...snipp...]
hduser@ubuntu:~$
```

If the SSH connect should fail, these general tips might help:

- Enable debugging with `ssh -vvv localhost` and investigate the error in detail.
- Check the SSH server configuration in `/etc/ssh/sshd_config`, in particular the options `PubkeyAuthentication` (which should be set to `yes`) and `AllowUsers` (if this option is active, add the `hduser` user to it). If you made any changes to the SSH server configuration file, you can force a configuration reload with `sudo /etc/init.d/ssh reload`.

Disabling IPv6

One problem with IPv6 on Ubuntu is that using `0.0.0.0` for the various networking-related Hadoop configuration options will result in Hadoop binding to the IPv6 addresses of my Ubuntu box. In my case, I realized that there's no practical point in enabling IPv6 on a box when you are not connected to any IPv6 network. Hence, I simply disabled IPv6 on my Ubuntu machine. Your mileage may vary.

To disable IPv6 on Ubuntu 10.04 LTS, open `/etc/sysctl.conf` in the editor of your choice and add the following lines to the end of the file:

/etc/sysctl.conf

```
# disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

You have to reboot your machine in order to make the changes take effect.

You can check whether IPv6 is enabled on your machine with the following command:

```
$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

A return value of 0 means IPv6 is enabled, a value of 1 means disabled (that's what we want).

Alternative

You can also disable IPv6 only for Hadoop as documented in HADOOP-3437. You can do so by adding the following line to `conf/hadoop-env.sh`:

conf/hadoop-env.sh

```
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

Hadoop

Installation

Download Hadoop from the Apache Download Mirrors and extract the contents of the Hadoop package to a location of your choice. I picked `/usr/local/hadoop`. Make sure to change the owner of all the files to the `hduser` user and `hadoop` group, for example:

```
$ cd /usr/local
$ sudo tar xzf hadoop-1.0.3.tar.gz
$ sudo mv hadoop-1.0.3 hadoop
$ sudo chown -R hduser:hadoop hadoop
```

(Just to give you the idea, YMMV – personally, I create a symlink from `hadoop-1.0.3` to `hadoop`.)

Update $HOME/.bashrc

Add the following lines to the end of the `$HOME/.bashrc` file of user `hduser`. If you use a shell other than bash, you should of course update its appropriate configuration files instead of `.bashrc`.

$HOME/.bashrc

```
# Set Hadoop-related environment variables
export HADOOP_HOME=/usr/local/hadoop

# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
export JAVA_HOME=/usr/lib/jvm/java-6-sun

# Some convenient aliases and functions for running Hadoop-related commands
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# If you have LZO compression enabled in your Hadoop cluster and
# compress job outputs with LZOP (not covered in this tutorial):
# Conveniently inspect an LZOP compressed file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Requires installed 'lzop' command.
#
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

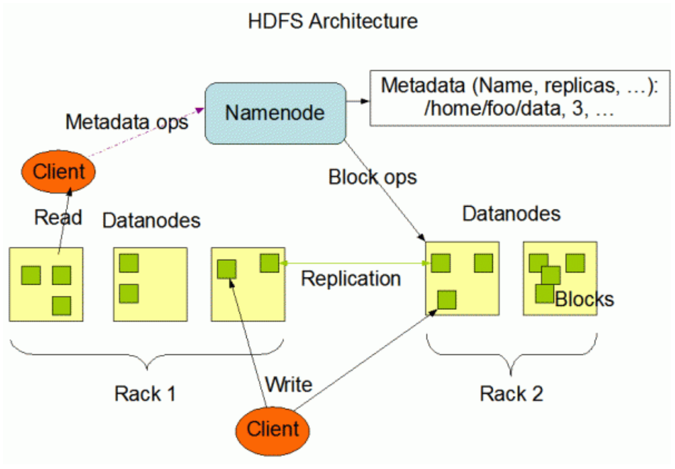
You can repeat this exercise also for other users who want to use Hadoop.

Excursus: Hadoop Distributed File System (HDFS)

Before we continue let us briefly learn a bit more about Hadoop's distributed file system.

> The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is part of the Apache Hadoop project, which is part of the Apache Lucene project.
>
> **The Hadoop Distributed File System: Architecture and Design** *hadoop.apache.org/hdfs/docs/…*

The following picture gives an overview of the most important HDFS components.



Configuration

Our goal in this tutorial is a single-node setup of Hadoop. More information of what we do in this section is available on the Hadoop Wiki.

hadoop-env.sh

The only required environment variable we have to configure for Hadoop in this tutorial is `JAVA_HOME`. Open `conf/hadoop-env.sh` in the editor of your choice (if you used the installation path in this tutorial, the full path is `/usr/local/hadoop/conf/hadoop-env.sh`) and set the `JAVA_HOME` environment variable to the Sun JDK/JRE 6 directory.

Change

conf/hadoop-env.sh

```
# The java implementation to use.  Required.
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

to

conf/hadoop-env.sh

```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

Note: If you are on a Mac with OS X 10.7 you can use the following line to set up `JAVA_HOME` in `conf/hadoop-env.sh`.

conf/hadoop-env.sh (on Mac systems)

```
# for our Mac users
export JAVA_HOME=`/usr/libexec/java_home`
```

conf/*-site.xml

In this section, we will configure the directory where Hadoop will store its data files, the network ports it listens to, etc. Our setup will use Hadoop's Distributed File System, HDFS, even though our little "cluster" only contains our single local machine.

You can leave the settings below "as is" with the exception of the `hadoop.tmp.dir` parameter – this parameter you must change to a directory of your choice. We will use the directory `/app/hadoop/tmp` in this tutorial. Hadoop's default configurations use `hadoop.tmp.dir` as the base temporary directory both for the local file system and HDFS, so don't be surprised if you see Hadoop creating the specified directory automatically on HDFS at some later point.

Now we create the directory and set the required ownerships and permissions:

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
# ...and if you want to tighten up security, chmod from 755 to 750...
$ sudo chmod 750 /app/hadoop/tmp
```

If you forget to set the required ownerships and permissions, you will see a `java.io.IOException` when you try to format the name node in the next section).

Add the following snippets between the `<configuration> ... </configuration>` tags in the respective configuration XML file.

In file `conf/core-site.xml`:

conf/core-site.xml

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

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

In file `conf/mapred-site.xml`:

conf/mapred-site.xml

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

In file `conf/hdfs-site.xml`:

conf/hdfs-site.xml

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
```

See Getting Started with Hadoop and the documentation in Hadoop's API Overview if you have any questions about Hadoop's configuration options.

Formatting the HDFS filesystem via the NameNode

The first step to starting up your Hadoop installation is formatting the Hadoop filesystem which is implemented on top of the local filesystem of your "cluster" (which includes only your local machine if you followed this tutorial). You need to do this the first time you set up a Hadoop cluster.

Do not format a running Hadoop filesystem as you will lose all the data currently in the cluster (in HDFS)!

To format the filesystem (which simply initializes the directory specified by the `dfs.name.dir` variable), run the command

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

The output will look like this:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop namenode -format
10/05/08 16:59:56 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntu/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.2
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20 -r 911707; compiled by 'chrisdo' on Fri Feb 19 08:07:34 UTC 2010
************************************************************/
10/05/08 16:59:56 INFO namenode.FSNamesystem: fsOwner=hduser,hadoop
10/05/08 16:59:56 INFO namenode.FSNamesystem: supergroup=supergroup
10/05/08 16:59:56 INFO namenode.FSNamesystem: isPermissionEnabled=true
10/05/08 16:59:56 INFO common.Storage: Image file of size 96 saved in 0 seconds.
10/05/08 16:59:57 INFO common.Storage: Storage directory .../hadoop-hduser/dfs/name has been successfully formatted.
10/05/08 16:59:57 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at ubuntu/127.0.1.1
************************************************************/
hduser@ubuntu:/usr/local/hadoop$
```

Starting your single-node cluster

Run the command:

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

This will startup a Namenode, Datanode, Jobtracker and a Tasktracker on your machine.

The output will look like this:

```
hduser@ubuntu:/usr/local/hadoop$ bin/start-all.sh
starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-namenode-ubuntu.out
localhost: starting datanode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-ubuntu.out
localhost: starting secondarynamenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-secondarynamenode-ubuntu.out
starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-jobtracker-ubuntu.out
localhost: starting tasktracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-tasktracker-ubuntu.out
hduser@ubuntu:/usr/local/hadoop$
```

A nifty tool for checking whether the expected Hadoop processes are running is `jps` (part of Sun's Java since v1.5.0). See also How to debug MapReduce programs.

```
hduser@ubuntu:/usr/local/hadoop$ jps
2287 TaskTracker
2149 JobTracker
1938 DataNode
2085 SecondaryNameNode
2349 Jps
1788 NameNode
```

You can also check with `netstat` if Hadoop is listening on the configured ports.

```
hduser@ubuntu:~$ sudo netstat -plten | grep java
tcp   0  0 0.0.0.0:50070   0.0.0.0:*  LISTEN  1001  9236  2471/java
tcp   0  0 0.0.0.0:50010   0.0.0.0:*  LISTEN  1001  9998  2628/java
tcp   0  0 0.0.0.0:48159   0.0.0.0:*  LISTEN  1001  8496  2628/java
tcp   0  0 0.0.0.0:53121   0.0.0.0:*  LISTEN  1001  9228  2857/java
tcp   0  0 127.0.0.1:54310 0.0.0.0:*  LISTEN  1001  8143  2471/java
tcp   0  0 127.0.0.1:54311 0.0.0.0:*  LISTEN  1001  9230  2857/java
tcp   0  0 0.0.0.0:59305   0.0.0.0:*  LISTEN  1001  8141  2471/java
tcp   0  0 0.0.0.0:50060   0.0.0.0:*  LISTEN  1001  9857  3005/java
tcp   0  0 0.0.0.0:49900   0.0.0.0:*  LISTEN  1001  9037  2785/java
tcp   0  0 0.0.0.0:50030   0.0.0.0:*  LISTEN  1001  9773  2857/java
hduser@ubuntu:~$
```

If there are any errors, examine the log files in the `/logs/` directory.

Stopping your single-node cluster

Run the command

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/stop-all.sh
```

to stop all the daemons running on your machine.

Example output:

```
hduser@ubuntu:/usr/local/hadoop$ bin/stop-all.sh
stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: stopping datanode
localhost: stopping secondarynamenode
hduser@ubuntu:/usr/local/hadoop$
```

Running a MapReduce job

We will now run your first Hadoop MapReduce job. We will use the WordCount example job which reads text files and counts how often words occur. The input is text files and the output is text files, each line of which contains a word and the count of how often it occurred, separated by a tab. More information of what happens behind the scenes is available at the Hadoop Wiki.

Download example input data

We will use three ebooks from Project Gutenberg for this example:

- The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson
- The Notebooks of Leonardo Da Vinci
- Ulysses by James Joyce

Download each ebook as text files in `Plain Text UTF-8` encoding and store the files in a local temporary directory of choice, for example `/tmp/gutenberg`.

```
hduser@ubuntu:~$ ls -l /tmp/gutenberg/
total 3604
-rw-r--r-- 1 hduser hadoop  674566 Feb  3 10:17 pg20417.txt
-rw-r--r-- 1 hduser hadoop 1573112 Feb  3 10:18 pg4300.txt
-rw-r--r-- 1 hduser hadoop 1423801 Feb  3 10:18 pg5000.txt
hduser@ubuntu:~$
```

Restart the Hadoop cluster

Restart your Hadoop cluster if it's not running already.

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

Copy local example data to HDFS

Before we run the actual MapReduce job, we first have to copy the files from our local file system to Hadoop's HDFS.

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -copyFromLocal /tmp/gutenberg /user/hduser/gutenberg
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /user/hduser
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2010-05-08 17:40 /user/hduser/gutenberg
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /user/hduser/gutenberg
Found 3 items
-rw-r--r--   3 hduser supergroup     674566 2011-03-10 11:38 /user/hduser/gutenberg/pg20417.txt
-rw-r--r--   3 hduser supergroup    1573112 2011-03-10 11:38 /user/hduser/gutenberg/pg4300.txt
-rw-r--r--   3 hduser supergroup    1423801 2011-03-10 11:38 /user/hduser/gutenberg/pg5000.txt
hduser@ubuntu:/usr/local/hadoop$
```

Run the MapReduce job

Now, we actually run the WordCount example job.

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop*examples*.jar wordcount /user/hduser/gutenberg /user/hduser/gutenberg-output
```

This command will read all the files in the HDFS directory `/user/hduser/gutenberg`, process it, and store the result in the HDFS directory `/user/hduser/gutenberg-output`.

Note: Some people run the command above and get the following error message:

```
Exception in thread "main" java.io.IOException: Error opening job jar: hadoop*examples*.jar
at org.apache.hadoop.util.RunJar.main (RunJar.java: 90)
Caused by: java.util.zip.ZipException: error in opening zip file
```

In this case, re-run the command with the full name of the Hadoop Examples JAR file, for example:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop-examples-1.0.3.jar wordcount /user/hduser/gutenberg /user/hduser/gutenberg-output
```

Example output of the previous command in the console:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop*examples*.jar wordcount /user/hduser/gutenberg /user/hduser/gutenberg-output
10/05/08 17:43:00 INFO input.FileInputFormat: Total input paths to process : 3
10/05/08 17:43:01 INFO mapred.JobClient: Running job: job_201005081732_0001
10/05/08 17:43:02 INFO mapred.JobClient:  map 0% reduce 0%
10/05/08 17:43:14 INFO mapred.JobClient:  map 66% reduce 0%
10/05/08 17:43:17 INFO mapred.JobClient:  map 100% reduce 0%
10/05/08 17:43:26 INFO mapred.JobClient:  map 100% reduce 100%
10/05/08 17:43:28 INFO mapred.JobClient: Job complete: job_201005081732_0001
10/05/08 17:43:28 INFO mapred.JobClient: Counters: 17
10/05/08 17:43:28 INFO mapred.JobClient:   Job Counters
10/05/08 17:43:28 INFO mapred.JobClient:     Launched reduce tasks=1
10/05/08 17:43:28 INFO mapred.JobClient:     Launched map tasks=3
10/05/08 17:43:28 INFO mapred.JobClient:     Data-local map tasks=3
10/05/08 17:43:28 INFO mapred.JobClient:   FileSystemCounters
10/05/08 17:43:28 INFO mapred.JobClient:     FILE_BYTES_READ=2214026
10/05/08 17:43:28 INFO mapred.JobClient:     HDFS_BYTES_READ=3639512
10/05/08 17:43:28 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=3687918
10/05/08 17:43:28 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=880330
10/05/08 17:43:28 INFO mapred.JobClient:   Map-Reduce Framework
10/05/08 17:43:28 INFO mapred.JobClient:     Reduce input groups=82290
10/05/08 17:43:28 INFO mapred.JobClient:     Combine output records=102286
10/05/08 17:43:28 INFO mapred.JobClient:     Map input records=77934
10/05/08 17:43:28 INFO mapred.JobClient:     Reduce shuffle bytes=1473796
10/05/08 17:43:28 INFO mapred.JobClient:     Reduce output records=82290
10/05/08 17:43:28 INFO mapred.JobClient:     Spilled Records=255874
10/05/08 17:43:28 INFO mapred.JobClient:     Map output bytes=6076267
10/05/08 17:43:28 INFO mapred.JobClient:     Combine input records=629187
10/05/08 17:43:28 INFO mapred.JobClient:     Map output records=629187
10/05/08 17:43:28 INFO mapred.JobClient:     Reduce input records=102286
```

Check if the result is successfully stored in HDFS directory `/user/hduser/gutenberg-output`:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /user/hduser
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2010-05-08 17:40 /user/hduser/gutenberg
drwxr-xr-x   - hduser supergroup          0 2010-05-08 17:43 /user/hduser/gutenberg-output
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /user/hduser/gutenberg-output
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2010-05-08 17:43 /user/hduser/gutenberg-output/_logs
-rw-r--r--   1 hduser supergroup     880802 2010-05-08 17:43 /user/hduser/gutenberg-output/part-r-00000
hduser@ubuntu:/usr/local/hadoop$
```

If you want to modify some Hadoop settings on the fly like increasing the number of Reduce tasks, you can use the `"-D"` option:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop*examples*.jar wordcount -D mapred.reduce.tasks=16 /user/hduser/gutenberg /user/hduser/gutenberg-output
```

> An important note about `mapred.map.tasks`: Hadoop does not honor mapred.map.tasks beyond considering it a hint. But it accepts the user specified `mapred.reduce.tasks` and doesn't manipulate that. You cannot force `mapred.map.tasks` but you can specify `mapred.reduce.tasks`.

Retrieve the job result from HDFS

To inspect the file, you can copy it from HDFS to the local file system. Alternatively, you can use the command

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -cat /user/hduser/gutenberg-output/part-r-00000
```

to read the file directly from HDFS without copying it to the local file system. In this tutorial, we will copy the results to the local file system though.

```
hduser@ubuntu:/usr/local/hadoop$ mkdir /tmp/gutenberg-output
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -getmerge /user/hduser/gutenberg-output /tmp/gutenberg-output
hduser@ubuntu:/usr/local/hadoop$ head /tmp/gutenberg-output/gutenberg-output
"(Lo)cra"       1
"1490   1
"1498," 1
"35"    1
"40,"   1
"A      2
"AS-IS".        1
"A_     1
"Absoluti       1
"Alack! 1
hduser@ubuntu:/usr/local/hadoop$
```

Note that in this specific output the quote signs (") enclosing the words in the `head` output above have not been inserted by Hadoop. They are the result of the word tokenizer used in the WordCount example, and in this case they matched the beginning of a quote in the ebook texts. Just inspect the `part-00000` file further to see it for yourself.

> The command `fs -getmerge` will simply concatenate any files it finds in the directory you specify. This means that the merged file might (and most likely will) **not be sorted**.

Hadoop Web Interfaces

Hadoop comes with several web interfaces which are by default (see `conf/hadoop-default.xml`) available at these locations:

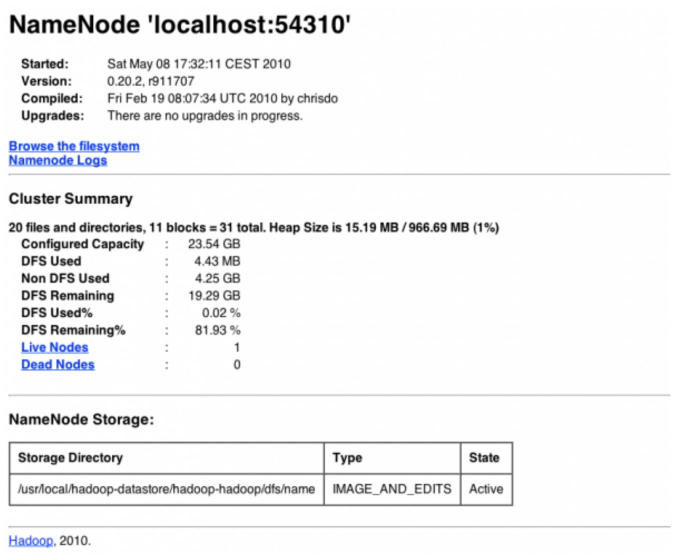
- http://localhost:50070/ – web UI of the NameNode daemon
- http://localhost:50030/ – web UI of the JobTracker daemon
- http://localhost:50060/ – web UI of the TaskTracker daemon

These web interfaces provide concise information about what’s happening in your Hadoop cluster. You might want to give them a try.

NameNode Web Interface (HDFS layer)

The name node web UI shows you a cluster summary including information about total/remaining capacity, live and dead nodes. Additionally, it allows you to browse the HDFS namespace and view the contents of its files in the web browser. It also gives access to the local machine’s Hadoop log files.

By default, it’s available at http://localhost:50070/.



NameNode 'localhost:54310'

Started:	Sat May 08 17:32:11 CEST 2010
Version:	0.20.2, r911707
Compiled:	Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

20 files and directories, 11 blocks = 31 total. Heap Size is 15.19 MB / 966.69 MB (1%)

Configured Capacity	:	23.54 GB
DFS Used	:	4.43 MB
Non DFS Used	:	4.25 GB
DFS Remaining	:	19.29 GB
DFS Used%	:	0.02 %
DFS Remaining%	:	81.93 %
Live Nodes	:	1
Dead Nodes	:	0

NameNode Storage:

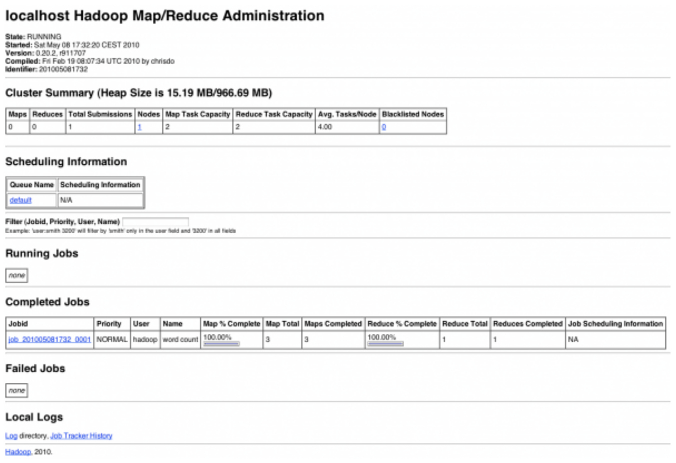
Storage Directory	Type	State
/usr/local/hadoop-datastore/hadoop-hadoop/dfs/name	IMAGE_AND_EDITS	Active

Hadoop, 2010.

JobTracker Web Interface (MapReduce layer)

The JobTracker web UI provides information about general job statistics of the Hadoop cluster, running/completed/failed jobs and a job history log file. It also gives access to the ‘‘local machine’s’’ Hadoop log files (the machine on which the web UI is running on).

By default, it’s available at http://localhost:50030/.



localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Sat May 08 17:32:20 CEST 2010
Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Identifier: 201005081732

Cluster Summary (Heap Size is 15.19 MB/966.69 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	1	1	2	2	4.00	0

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_201005081732_0001	NORMAL	hadoop	word count	100.00%	3	3	100.00%	1	1	NA

Failed Jobs

none

Local Logs

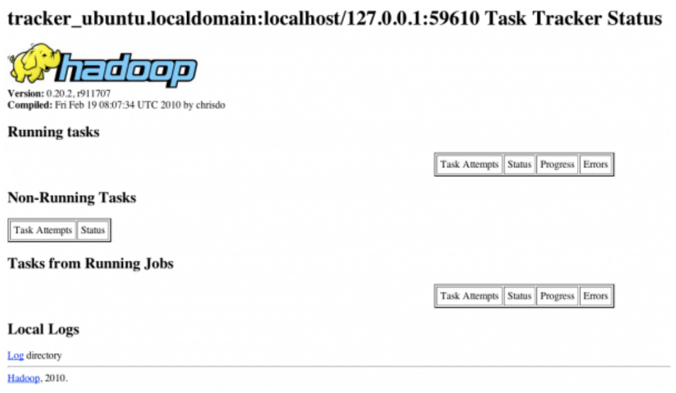
Log directory, Job Tracker History

Hadoop, 2010.

TaskTracker Web Interface (MapReduce layer)

The task tracker web UI shows you running and non-running tasks. It also gives access to the ‘‘local machine’s’’ Hadoop log files.

By default, it’s available at http://localhost:50060/.


tracker_ubuntu.localdomain:localhost/127.0.0.1:59610 Task Tracker Status

Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo

Running tasks

Task Attempts	Status	Progress	Errors

Non-Running Tasks

Task Attempts	Status

Tasks from Running Jobs

Task Attempts	Status	Progress	Errors

Local Logs

Log directory

Hadoop, 2010.

What’s next?

If you’re feeling comfortable, you can continue your Hadoop experience with my follow-up tutorial Running Hadoop On Ubuntu Linux (Multi-Node Cluster) where I describe how to build a Hadoop ‘‘multi-node’’ cluster with two Ubuntu boxes (this will increase your current cluster size by 100%, heh).

In addition, I wrote a tutorial on how to code a simple MapReduce job in the Python programming language which can serve as the basis for writing your own MapReduce programs.

Related Links

From yours truly:

- Running Hadoop On Ubuntu Linux (Multi-Node Cluster)
- Writing An Hadoop MapReduce Program In Python

From other people:

- How to debug MapReduce programs
- Hadoop API Overview (for Hadoop 2.x)

Change Log

Only important changes to this article are listed here:

- 2011-07-17: Renamed the Hadoop user from `hadoop` to `hduser` based on readers’ feedback. This should make the distinction between the local Hadoop user (now `hduser`), the local Hadoop group (`hadoop`), and the Hadoop CLI tool (`hadoop`) more clear.

Tweet 16

Comments

825 Comments Michael G. Noll 1 Login

Recommend 38 Share Sort by Best

Join the discussion…

Frederic Stahl • 2 years ago
Hi,

I'm sorry to be a pain. Some time ago I tried this tutorial and I coudn't get it to work. I have now made a second attempt, and this time I belief I have followed it word by word with a fresh XUbuntu installation. I have indeed succeeded!!!

Well, then I tried it again a couple of hours later. I formatted the namenode again using namenode -format. Then I tried starting the cluster again. What always happens is that either the namenode or the datanode are not starting. Please find below the commandline output, I can't see anything odd happening.

Thanks in advance for the help!

Frederic

hduser@Else:/usr/local/hadoop/hadoop/bin$ ./start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/hadoop/libexec/../logs/hadoop-hduser-namenode-Else.out

see more

48 • Reply • Share ›

Stephen Baker → Frederic Stahl • a year ago
I found you sometimes just have to delete the contents of the local files in your area reserved for the HDFS (in this case.../app/hadoop...and there is a dfs folder in there somewhere that is the culprit). You'll have to have root/sudo access to delete it (easy enough). I found that when my namenode wouldn't format, deleting the dfs folder locally fixed it and allowed a format and all of the services starting. It's not optimal and I have not touched the Hadoop infrastructure side in a little over a year, though.

Also, the deprecated message can be fixed by using Hadoop Prefix instead of Hadoop Home in your environment (.bashrc in Ubuntu).

• Reply • Share ›

Prathibha P G • 3 years ago
Sir,
When I run ./start-all.sh jobtracker and tasktracker is not started.Also there is nothing written inside the log files.

Kindly help me
21 • Reply • Share ›

Prathibha.P.G • 3 years ago
Sir,
I have a problem.When I run ./start-all.sh,I have secondarynode only.When I use jps, there is no jobtracker,tasktracker,namenode etc.There is nothing written inside the log file. Kindly help me .I am waiting for your reply
12 • Reply • Share ›

Melvin Lobo → Prathibha.P.G • 3 years ago
run stop-all.sh

Then run only namenode first and check.

bin/hadoop-daemon.sh start namenode
6 • Reply • Share ›

@dasfaha → Prathibha.P.G • 3 years ago
Reformat the node and check that you are not getting any error. When I had that problem it was because I didn't notice the errors when I tried to format the node, and those errors were due to me specifying the hdfs directory incorrectly in this section:

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

About Me

I am a researcher and software engineer based in Switzerland, Europe. In my day job I am the developer evangelist of Confluent, i.e. the US startup founded in 2014 by the creators of Apache Kafka who developed Kafka while at LinkedIn. At Confluent we are focusing on building a stream data platform to help other companies get easy access to enterprise data as real-time streams. Read more »

Contact

michael@michael-noll.com

Follow Me

Twitter
Blog RSS
GitHub
Slideshare

Recent Posts

- Integrating Kafka and Spark Streaming: Code Examples and State of the Game
- Apache Storm 0.9 training deck and tutorial
- Apache Kafka 0.8 training deck and tutorial
- Integrating Kafka and Storm: Code Examples and State of the Game
- Wirbelsturm: 1-Click Deployments of Storm and Kafka clusters with Vagrant and Puppet