Running Hadoop on Ubuntu Linux (Multi-Node Cluster)

Are you looking for the single-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
In a previous tutorial, I described [how to setup up a Hadoop single-node cluster](#) on an Ubuntu box. The main goal of this tutorial is to get a more sophisticated Hadoop installation up and running, namely building a multi-node cluster using two Ubuntu boxes.

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

![Cluster of machines running Hadoop at Yahoo!](#)

**Tutorial approach and structure**

**From two single-node clusters to a multi-node cluster** – We will build a multi-node cluster using two Ubuntu boxes in this tutorial. In my humble opinion, the best way to do this for starters is to install, configure and test a “local” Hadoop setup for each of the two Ubuntu boxes, and in a second step to “merge” these two single-node clusters into one multi-node cluster in which one Ubuntu box will become the designated master (but also act as a slave with regard to data storage and processing), and the other box will become only a slave. It’s much easier to track down any problems you might encounter due to the reduced complexity of doing a single-node cluster setup first on each machine.

![Diagram showing tutorial approach and structure](#)

Let’s get started!

**Prerequisites**
Configuring single-node clusters first

The tutorial approach outlined above means that you should read now my previous tutorial on how to setup a Hadoop single-node cluster and follow the steps described there to build a single-node Hadoop cluster on each of the two Ubuntu boxes. It is recommended that you use the “same settings” (e.g., installation locations and paths) on both machines, or otherwise you might run into problems later when we will migrate the two machines to the final multi-node cluster setup.

Just keep in mind when setting up the single-node clusters that we will later connect and “merge” the two machines, so pick reasonable network settings etc. now for a smooth transition later.

Done? Let’s continue then!

Now that you have two single-node clusters up and running, we will modify the Hadoop configuration to make one Ubuntu box the “master” (which will also act as a slave) and the other Ubuntu box a “slave”.

Note: We will call the designated master machine just the “master” from now on and the slave-only machine the “slave”. We will also give the two machines these respective hostnames in their networking setup, most notably in "/etc/hosts". If the hostnames of your machines are different (e.g. “node01”) then you must adapt the settings in this tutorial as appropriate.

Networking

This should come hardly as a surprise, but for the sake of completeness I have to point out that both machines must be able to reach each other over the network. The easiest is to put both machines in the same network with regard to hardware and software configuration, for example connect both machines via a single hub or switch and configure the network interfaces to use a common network such as 192.168.0.x/24.

To make it simple, we will assign the IP address 192.168.0.1 to the master machine and 192.168.0.2 to the slave machine. Update /etc/hosts on both machines with the following lines:

```
/etc/hosts (for master AND slave)
1 192.168.0.1     master
2 192.168.0.2     slave
```

SSH access

The hduser user on the master (aka hduser@master) must be able to connect a) to its own user account on the master — i.e. ssh master in this context and not necessarily ssh localhost — and b) to the hduser user account on the slave (aka hduser@slave) via a password-less SSH login. If you followed my single-node cluster tutorial, you just have to add the hduser@master's public SSH key (which should be in $HOME/.ssh/id_rsa.pub) to the authorized_keys file of hduser@slave (in this user's $HOME/.ssh/authorized_keys). You can do this manually or use the following SSH command:

```
Distribute the SSH public key of hduser@master

1 hduser@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave
```

This command will prompt you for the login password for user hduser on slave, then copy the public SSH key for you, creating the correct directory and fixing the permissions as necessary.

The final step is to test the SSH setup by connecting with user hduser from the master to the user account hduser on the slave. The step is also needed to save slave’s host key fingerprint to the hduser@master’s known_hosts file.

So, connecting from master to master...

```
1 hduser@master:~$ ssh master
2 The authenticity of host 'master (192.168.0.1)' can't be established.
4 Are you sure you want to continue connecting (yes/no)? yes
5 Warning: Permanently added 'master' (RSA) to the list of known hosts.
6 Linux master 2.6.20-16-386 #2 Thu Jun 7 20:16:13 UTC 2007 i686
7 ...
8 # hduser@master:~$
```

…and from master to slave.
Hadoop

Cluster Overview (aka the goal)

The next sections will describe how to configure one Ubuntu box as a master node and the other Ubuntu box as a slave node. The master node will also act as a slave because we only have two machines available in our cluster but still want to spread data storage and processing to multiple machines.

![Diagram of multi-node cluster]

Figure 3: How the final multi-node cluster will look like

The master node will run the “master” daemons for each layer: NameNode for the HDFS storage layer, and JobTracker for the MapReduce processing layer. Both machines will run the “slave” daemons: DataNode for the HDFS layer, and TaskTracker for MapReduce processing layer. Basically, the “master” daemons are responsible for coordination and management of the “slave” daemons while the latter will do the actual data storage and data processing work.

Masters vs. Slaves

Typically one machine in the cluster is designated as the NameNode and another machine the as JobTracker, exclusively. These are the actual “master nodes”. The rest of the machines in the cluster act as both DataNode and TaskTracker. These are the slaves or “worker nodes”.

Hadoop 1.x documentation [hadoop.apache.org/common/docs/…]

Configuration

conf/masters (master only)

Despite its name, the conf/masters file defines on which machines Hadoop will start secondary NameNodes in our multi-node cluster. In our case, this is just the master machine. The primary NameNode and the JobTracker will always be the machines on which you run the bin/start-dfs.sh and bin/start-mapred.sh scripts, respectively (the primary NameNode and the JobTracker will be started on the same machine if you run bin/start-all.sh).
Note: You can also start an Hadoop daemon manually on a machine via `bin/hadoop-daemon.sh start [namenode | secondarynamenode | datanode | jobtracker | tasktracker]`, which will not take the "conf/masters" and "conf/slaves" files into account.

Here are more details regarding the `conf/masters` file:

The secondary NameNode merges the fsimage and the edits log files periodically and keeps edits log size within a limit. It is usually run on a different machine than the primary NameNode since its memory requirements are on the same order as the primary NameNode. The secondary NameNode is started by “bin/start-dfs.sh” on the nodes specified in "conf/masters" file.

Hadoop HDFS user guide [hadoop.apache.org/common/docs/...]

Again, the machine on which `bin/start-dfs.sh` is run will become the primary NameNode.

On master, update `conf/masters` that it looks like this:

```
conf/masters (on master)
1 master
```

**conf/slaves (master only)**

The `conf/slaves` file lists the hosts, one per line, where the Hadoop slave daemons (DataNodes and TaskTrackers) will be run. We want both the master box and the slave box to act as Hadoop slaves because we want both of them to store and process data.

On master, update `conf/slaves` that it looks like this:

```
conf/slaves (on master)
1 master
2 slave
```

If you have additional slave nodes, just add them to the `conf/slaves` file, one hostname per line.

```
conf/slaves (on master)
1 master
2 slave
3 anotherslave01
4 anotherslave02
5 anotherslave03
```

Note: The `conf/slaves` file on master is used only by the scripts like `bin/start-dfs.sh` or `bin/stop-dfs.sh`. For example, if you want to add DataNodes on the fly (which is not described in this tutorial yet), you can “manually” start the DataNode daemon on a new slave machine via `bin/hadoop-daemon.sh start datanode`. Using the `conf/slaves` file on the master simply helps you to make “full” cluster restarts easier.

**conf/*-site.xml (all machines)**

You must change the configuration files `conf/core-site.xml`, `conf/mapred-site.xml` and `conf/hdfs-site.xml` on ALL machines as follows.

First, we have to change the `fs.default.name` parameter (in `conf/core-site.xml`), which specifies the NameNode (the HDFS master) host and port. In our case, this is the master machine.

```
conf/core-site.xml (ALL machines)
1 <property>
2 <name>fs.default.name</name>
3 <value>hdfs://master:54310</value>
4 <description>The name of the default file system. A URI whose
5 scheme and authority determine the FileSystem implementation. The
6 uri's scheme determines the config property (fs.SCHEME.impl) naming
7 the FileSystem implementation class. The uri's authority is used to
8 determine the host, port, etc. for a filesystem.</description>
9 </property>
```
Second, we have to change the `mapred.job.tracker` parameter (in `conf/mapred-site.xml`), which specifies the `JobTracker` (MapReduce master) host and port. Again, this is the `master` in our case.

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

Third, we change the `dfs.replication` parameter (in `conf/hdfs-site.xml`) which specifies the default block replication. It defines how many machines a single file should be replicated to before it becomes available. If you set this to a value higher than the number of available slave nodes (more precisely, the number of DataNodes), you will start seeing a lot of "(Zero targets found, forbidden1.size=1)" type errors in the log files.

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

### Additional Settings

There are some other configuration options worth studying. The following information is taken from the [Hadoop API Overview](http://hadoop.apache.org/core/docs/current/hadoop-project-dist/hadoop-common/Configuration-Keys.html).

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

- "mapred.local.dir"  
  Determines where temporary MapReduce data is written. It also may be a list of directories.

- "mapred.map.tasks"  
  As a rule of thumb, use 10x the number of slaves (i.e., number of TaskTrackers).

- "mapred.reduce.tasks"  
  As a rule of thumb, use `num_tasktrackers * num_reduce_slots_per_tasktracker * 0.99`. If `num_tasktrackers` is small (as in the case of this tutorial), use `(num_tasktrackers - 1) * num_reduce_slots_per_tasktracker`.

### Formatting the HDFS filesystem via the NameNode

Before we start our new multi-node cluster, we must format Hadoop’s distributed filesystem (HDFS) via the NameNode. You need to do this the first time you set up an Hadoop cluster.

Warning: Do not format a running cluster because this will erase all existing data in the HDFS filesystem!

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

```
$ haduser@master:/usr/local/hadoop$ bin/hadoop namenode -format
```

```
INFO dfs.Storage: Storage directory /app/hadoop/tmp/dfs/name has been successfully formatted.
```

```
$ haduser@master:/usr/local/hadoop$
```

Background: The HDFS name table is stored on the NameNode’s (here: `master`) local filesystem in the directory specified by `dfs.name.dir`. The name table is used by the NameNode to store tracking and coordination information for the DataNodes.

### Starting the multi-node cluster

---

www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/
Starting the cluster is performed in two steps.

1. We begin with starting the HDFS daemons: the NameNode daemon is started on master, and DataNode daemons are started on all slaves (here: master and slave).
2. Then we start the MapReduce daemons: the JobTracker is started on master, and TaskTracker daemons are started on all slaves (here: master and slave).

**HDFS daemons**

Run the command `bin/start-dfs.sh` on the machine you want the (primary) NameNode to run on. This will bring up HDFS with the NameNode running on the machine you ran the previous command on, and DataNodes on the machines listed in the `conf/slaves` file.

In our case, we will run `bin/start-dfs.sh` on `master`:

Start the HDFS layer

```
$ hdfs@master:/usr/local/hadoop$ bin/start-dfs.sh
2 starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-namenode-master.out
3 slave: Ubuntu 10.04
4 slave: starting datanode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-slave.out
5 master: starting datanode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-master.out
6 master: starting secondarynamenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hduser-secondarynamenode-master.out
$ hdfs@master:/usr/local/hadoop$
```

On slave, you can examine the success or failure of this command by inspecting the log file `logs/hadoop-hduser-datanode-slave.log`.

Example output:

```
1 ... INFO org.apache.hadoop.dfs.Storage: Storage directory /app/hadoop/tmp/dfs/data is not formatted.
2 ... INFO org.apache.hadoop.dfs.Storage: Formatting ...
3 ... INFO org.apache.hadoop.dfs.DataNode: Opened server at 50010
4 ... INFO org.mortbay.util.Credential: Checking Resource aliases
5 ... INFO org.mortbay.http.HttpServer: Version Jetty/5.1.4
6 ... INFO org.mortbay.util.Container: Started WebApplicationContext[/]
7 ... INFO org.mortbay.util.Container: Started WebApplicationContext[/]
8 ... INFO org.mortbay.util(Container: Started SocketListener: Started SocketListener on 0.0.0.0:50075
9 ... INFO org.mortbay.util.Container: Started SocketListener: Started SocketListener on 0.0.0.0:500075
10 ... INFO org.mortbay.util.Container: Started org.mortbay.jetty.Server@56a499
11 ... INFO org.mortbay.util.Container: Started org.mortbay.jetty.Server@56a499
12 ... INFO org.apache.hadoop.dfs.DataNode: Starting DataNode in: FSDataSet(dirpath='app/hadoop/tmp/dfs/data/current')
13 ... INFO org.apache.hadoop.dfs.DataNode: using BLOCKREPORT_INTERVAL of 353203msec
```

As you can see in slave’s output above, it will automatically format its storage directory (specified by the `dfs.data.dir` parameter) if it is not formatted already. It will also create the directory if it does not exist yet.

At this point, the following Java processes should run on master...

Java processes on master after starting HDFS daemons

```
1 hduser@master:/usr/local/hadoop$ jps
2 14799 NameNode
3 15314 Jps
4 14880 DataNode
5 14977 SecondaryNameNode
6 hduser@master:/usr/local/hadoop$
```

(the process IDs don’t matter of course)

...and the following on slave.

Java processes on slave after starting HDFS daemons

```
1 hduser@slave:/usr/local/hadoop$ jps
2 15183 DataNode
3 15616 Jps
4 hduser@slave:/usr/local/hadoop$
```
MapReduce daemons

Run the command `bin/start-mapred.sh` on the machine you want the JobTracker to run on. This will bring up the MapReduce cluster with the JobTracker running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the `conf/slaves` file.

In our case, we will run `bin/start-mapred.sh` on `master`:

Start the MapReduce layer

```
$ huser@master:/usr/local/hadoop$ bin/start-mapred.sh
2 starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-jobtracker-master.out
3 slave: Ubuntu 10.04
4 slave: starting tasktracker, logging to /usr/local/hadoop/bin/../logs/hadoop-huser-tasktracker-slave.out
5 master: starting tasktracker, logging to /usr/local/hadoop/bin/../logs/hadoop-huser-tasktracker-master.out
$ huser@master:/usr/local/hadoop
```

On `slave`, you can examine the success or failure of this command by inspecting the log file `logs/hadoop-huser-tasktracker-slave.log`. Example output:

```
1 ... INFO org.mortbay.util.Credential: Checking Resource aliases
2 ... INFO org.mortbay.http.HttpServletRequest: Version Jetty/5.1.4
3 ... INFO org.mortbay.util.Container: Started org.mortbay.jetty.servlet.WebApplicationHandler#d19bc8
4 ... INFO org.mortbay.util.Container: Started WebApplicationContext[/]
5 ... INFO org.mortbay.util.Container: Started WebApplicationContext[/static]/
6 ... INFO org.mortbay.util.Container: Started WebApplicationContext[/static,static]
7 ... INFO org.mortbay.http.SocketListener: Started SocketListener on 0.0.0.0:50060
8 ... INFO org.mortbay.util.Container: Started org.mortbay.jetty.server#e33ed
9 ... INFO org.apache.hadoop.ipc.Server: IPC Server listener on 50050: starting
10 ... INFO org.apache.hadoop.ipc.Server: IPC Server handler 0 on 50050: starting
11 ... INFO org.apache.hadoop.mapred.TaskTracker: TaskTracker up at: 50050
12 ... INFO org.apache.hadoop.mapred.TaskTracker: Starting tracker tracker_slave:50050
13 ... INFO org.apache.hadoop.ipc.Server: IPC Server handler 1 on 50050: starting
14 ...INFO org.apache.hadoop.mapred.TaskTracker: Starting thread: Map-events fetcher for all reduce tasks on tracker_slave:50050
```

At this point, the following Java processes should run on `master`:

Java processes on master after starting MapReduce daemons

```
$ huser@master:/usr/local/hadoop$ jps
2 16017 Jps
3 14799 NameNode
4 15686 TaskTracker
5 14880 DataNode
6 15596 JobTracker
7 14977 SecondaryNameNode
$ huser@master:/usr/local/hadoop
```

(the process IDs don’t matter of course)

...and the following on `slave`.

Java processes on slave after starting MapReduce daemons

```
$ huser@slave:/usr/local/hadoop$ jps
2 15183 DataNode
3 15897 TaskTracker
4 16284 Jps
$ huser@slave:/usr/local/hadoop
```

Stopping the multi-node cluster

Like starting the cluster, stopping it is done in two steps. The workflow however is the opposite of starting.
1. We begin with stopping the MapReduce daemons: the JobTracker is stopped on master, and TaskTracker daemons are stopped on all slaves (here: master and slave).

2. Then we stop the HDFS daemons: the NameNode daemon is stopped on master, and DataNode daemons are stopped on all slaves (here: master and slave).

MapReduce daemons

Run the command `bin/stop-mapred.sh` on the JobTracker machine. This will shut down the MapReduce cluster by stopping the JobTracker daemon running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the `conf/slaves` file.

In our case, we will run `bin/stop-mapred.sh` on master:

```
1 huser@master:/usr/local/hadoop$ bin/stop-mapred.sh
2 stopping JobTracker
3 slave: Ubuntu 10.04
4 master: stopping tasktracker
5 slave: stopping tasktracker
6 huser@master:/usr/local/hadoop$
```

Note: The output above might suggest that the JobTracker was running and stopped on "slave", but you can be assured that the JobTracker ran on "master".

At this point, the following Java processes should run on master...

```
Java processes on master after stopping MapReduce daemons
```

```
1 huser@master:/usr/local/hadoop$ jps
2 14799 NameNode
3 18386 Jps
4 14880 DataNode
5 14977 SecondaryNameNode
6 huser@master:/usr/local/hadoop$
```

...and the following on slave.

```
Java processes on slave after stopping MapReduce daemons
```

```
1 huser@slave:/usr/local/hadoop$ jps
2 15183 DataNode
3 18636 Jps
4 huser@slave:/usr/local/hadoop$
```

HDFS daemons

Run the command `bin/stop-dfs.sh` on the NameNode machine. This will shut down HDFS by stopping the NameNode daemon running on the machine you ran the previous command on, and DataNodes on the machines listed in the `conf/slaves` file.

In our case, we will run `bin/stop-dfs.sh` on master:

```
1 huser@master:/usr/local/hadoop$ bin/stop-dfs.sh
2 stopping namenode
3 slave: Ubuntu 10.04
4 slave: stopping datanode
5 master: stopping datanode
6 master: stopping secondarynamenode
7 huser@master:/usr/local/hadoop$
```

(again, the output above might suggest that the NameNode was running and stopped on slave, but you can be assured that the NameNode ran on master)

At this point, the only following `Java` processes should run on master...
Java processes on master after stopping HDFS daemons

```
1 hduser@master:/usr/local/hadoop$ jps
2 18670 Jps
3 hduser@master:/usr/local/hadoop$
```

…and the following on slave.

Java processes on slave after stopping HDFS daemons

```
1 hduser@slave:/usr/local/hadoop$ jps
2 18894 Jps
3 hduser@slave:/usr/local/hadoop$
```

## Running a MapReduce job

Just follow the steps described in the section Running a MapReduce job of the single-node cluster tutorial.

I recommend however that you use a larger set of input data so that Hadoop will start several Map and Reduce tasks, and in particular, on both master and slave. After all this installation and configuration work, we want to see the job processed by all machines in the cluster, don’t we?

Here’s the example input data I have used for the multi-node cluster setup described in this tutorial. I added four more Project Gutenberg etexts to the initial three documents mentioned in the single-node cluster tutorial. All etexts should be in plain text us-ascii encoding.

- The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson
- The Notebooks of Leonardo Da Vinci
- Ulysses by James Joyce
- The Art of War by 6th cent. B.C. Sunzi
- The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle
- The Devil’s Dictionary by Ambrose Bierce

Download these etexts, copy them to HDFS, run the WordCount example MapReduce job on master, and retrieve the job result from HDFS to your local filesystem.

Here’s the example output on master... after executing the MapReduce job...

```
1 hduser@master:/usr/local/hadoop$ bin/hadoop jar hadoop-examples*.jar wordcount /user/hduser/gutenberg /user/hduser/gutenberg-output
2 INFO mapred.FileInputFormat: Total input paths to process : 7
3 INFO mapred.JobClient: Running job: job_0001
4 INFO mapred.JobClient: map 0% reduce 0%
5 INFO mapred.JobClient: map 28% reduce 0%
6 INFO mapred.JobClient: map 57% reduce 0%
7 INFO mapred.JobClient: map 71% reduce 0%
8 INFO mapred.JobClient: map 100% reduce 9%
9 INFO mapred.JobClient: map 100% reduce 68%
10 INFO mapred.JobClient: map 100% reduce 100%
11 INFO mapred.JobClient: Job complete; job_0001
12 INFO mapred.JobClient: Counters: 11
13 INFO mapred.JobClient: org.apache.hadoop.examples.WordCount$Counter
14 INFO mapred.JobClient: WORDS=1173099
15 INFO mapred.JobClient: VALUES=1368295
16 INFO mapred.JobClient: Map-Reduce Framework
17 INFO mapred.JobClient: Map input records=136592
18 INFO mapred.JobClient: Map output records=1173099
19 INFO mapred.JobClient: Map input bytes=6925391
20 INFO mapred.JobClient: Map output bytes=11403568
21 INFO mapred.JobClient: Combine input records=1173099
22 INFO mapred.JobClient: Combine output records=195196
23 INFO mapred.JobClient: Reduce input groups=131275
24 INFO mapred.JobClient: Reduce input records=195196
25 INFO mapred.JobClient: Reduce output records=131275
26 hduser@master:/usr/local/hadoop$
```
...and the logging output on slave for its DataNode daemon...

```
logs/hadoop-hduser-datanode-slave.log (on slave)
```

```
1 ... INFO org.apache.hadoop.dfs.DataNode: Received block blk_5693969390309798974 from /192.168.0.1
2 ... INFO org.apache.hadoop.dfs.DataNode: Received block blk_7671491277162757352 from /192.168.0.1
3 ... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-711213365100166921 to /192.168.0.2
4 ... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-7545080504225510279 to /192.168.0.2
5 ... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-41144461842564099514 to /192.168.0.2
6 ... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-561652742730019659 to /192.168.0.2
7 ... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-3019298164878756077 to /app/hadoop/tmp/dfs/data/current/blk_-3019298164878756077
```

...and on slave for its TaskTracker daemon.

```
logs/hadoop-hduser-tasktracker-slave.log (on slave)
```

```
1 ... INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction: task_0001_m_000000_0
2 ... INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction: task_0001_m_000001_0
3 ... task_0001_m_000001_0 0.08362164% dfs://master:54310/user/hduser/gutenbierg/ulysses12.txt:0+1561677
4 ... task_0001_m_000000_0 0.07951202% dfs://master:54310/user/hduser/gutenbierg/19699.txt:0+1945731
5 ... INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction: task_0001_m_000000_0
6 ... task_0001_m_000000_0 1.0% dfs://master:54310/user/hduser/gutenbierg/19699.txt:0+1945731
7 ... Task task_0001_m_000001_0 is done.
8 ... task_0001_m_000000_0 1.0% dfs://master:54310/user/hduser/gutenbierg/19699.txt:0+1945731
9 ... Task task_0001_m_000001_0 is done.
10 ... Task task_0001_m_000000_0 is done.
11 ... Task task_0001_m_000000_0 is done.
12 ... Task task_0001_m_000000_0 is done.
13 ... Task task_0001_m_000000_0 0.6844295% dfs://master:54310/user/hduser/gutenbierg/132.txt:0+343695
14 ... Task task_0001_m_000000_0 0.095238104% reduce > copy (2 of 7 at 1.68 MB/s) >
15 ... Task task_0001_m_000000_0 1.0% dfs://master:54310/user/hduser/gutenbierg/132.txt:0+343695
16 ... Task task_0001_m_000000_0 is done.
17 ... task_0001_r_000000_0 0.14285716% reduce > copy (3 of 7 at 1.02 MB/s) >
18 <snip>
19 ... task_0001_r_000000_0 0.14285716% reduce > copy (3 of 7 at 1.02 MB/s) >
20 ... task_0001_r_000000_0 0.23809525% reduce > copy (5 of 7 at 0.32 MB/s) >
21 ... task_0001_r_000000_0 0.6859089% reduce > reduce
22 ... task_0001_r_000000_0 0.7897389% reduce > reduce
23 ... task_0001_r_000000_0 0.86781284% reduce > reduce
24 ... Task task_0001_r_000000_0 is done.
25 ... Received 'KillJobAction' for job: job_0001
26 ... task_0001_r_000000_0 done; removing files.
27 ... task_0001_m_000000_0 done; removing files.
28 ... task_0001_m_000000_0 done; removing files.
29 ... task_0001_m_000000_0 done; removing files.
```

If you want to inspect the job’s output data, you need to retrieve the job results from HDFS to your local file system (see instructions in the single-node cluster tutorial.

### Caveats

**java.io.IOException: Incompatible namespaceIDs**

If you observe the error “java.io.IOException: Incompatible namespaceIDs” in the logs of a DataNode (logs/hadoop-hduser-datanode..log), chances are you are affected by issue HDFS-107 (formerly known as HADOOP-1212).

The full error looked like this on my machines:

```
1 ... ERROR org.apache.hadoop.dfs.DataNode: java.io.IOException: Incompatible namespaceIDs in /app/hadoop/tmp/dfs/data: namenode namespaceID = 308967713; datanode namespaceID = 1123274563
2 ... at org.apache.hadoop.dfs.DataStorage.doTransition(DataStorage.java:281)
3 ... at org.apache.hadoop.dfs.DataStorage.recoverTransitionRead(DataStorage.java:121)
```

There are basically two solutions to fix this error as I will describe below.

**Solution 1: Start from scratch**

This step fixes the problem at the cost of erasing all existing data in the cluster’s HDFS file system.

1. Stop the full cluster, i.e. both MapReduce and HDFS layers.
2. Delete the data directory on the problematic DataNode: the directory is specified by `dfs.data.dir` in `conf/hdfs-site.xml`; if you followed this tutorial, the relevant directory is `/app/hadoop/tmp/dfs/data`.
3. Reformat the NameNode. **WARNING: all HDFS data is lost during this process!**
4. Restart the cluster.

When deleting all the HDFS data and starting from scratch does not sound like a good idea (it might be ok during the initial setup/testing), you might give the second approach a try.

**Solution 2: Manually update the namespaceID of problematic DataNodes**

Big thanks to Jared Stehler for the following suggestion. This workaround is “minimally invasive” as you only have to edit a single file on the problematic DataNodes:

1. Stop the problematic DataNode(s).
2. Edit the value of `namespaceID` in `${dfs.data.dir}/current/VERSION` to match the corresponding value of the current NameNode in `${dfs.name.dir}/current/VERSION`.
3. Restart the fixed DataNode(s).

If you followed the instructions in my tutorials, the full paths of the relevant files are:

- **NameNode**: `/app/hadoop/tmp/dfs/name/current/VERSION`
- **DataNode**: `/app/hadoop/tmp/dfs/data/current/VERSION` (background: `dfs.data.dir` is by default set to `${hadoop.tmp.dir}/dfs/data`, and we set `hadoop.tmp.dir` in this tutorial to `/app/hadoop/tmp`).

If you wonder how the contents of `VERSION` look like, here’s one of mine:

```
contents of current/VERSION
1 namespaceID=393514426
2 storageID=D5-1706792599-10.10.1.50010-1204306713481
3 ctime=1215607609074
4 storageType=DATA_NODE
5 layoutVersion=-13
```

**Where to go from here**

If you’re feeling comfortable, you can continue your Hadoop experience with my 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 (Single-Node Cluster)](http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/)
- [Writing An Hadoop MapReduce Program In Python](http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/)

From other people:

- [How to debug MapReduce programs](http://www.michael-noll.com/tutorials/how-to-debug-mapreduce.html)
- [Hadoop API Overview](http://www.michael-noll.com/tutorials/hadoop-api-overview.html) (Hadoop 2.x)
Change Log

Only major changes 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.

Comments

361 comments

Leave a message...

ivan • 8 months ago
Thanks for your tutorial. Here I got a problem.
If I put only one .txt file into dfs, the MapReduce will be completed without any problems.
But when I put more than one file, the job seems pending with below info:

"12/12/08 15:51:28 INFO mapred.JobClient: map 0% reduce 0%
12/12/08 15:51:42 INFO mapred.JobClient: map 50% reduce 0%
12/12/08 15:51:43 INFO mapred.JobClient: map 100% reduce 0%
12/12/08 15:51:51 INFO mapred.JobClient: map 100% reduce 16%"

In the same time, output in hadoop-hduser-tasktracker-xxx.log is "... reduce > copy (1 of 2 at 0.13MB/s)"

Navie • 11 months ago
Thank you for your tutorial. That’s the first long tutorial for Ubuntu that works perfectly for me :) 
btw, I got a warning message "$HADOOP_HOME is depreciated", could you help me with that? My hadoop folder is at /home/hduser/hadoop.

Michael G. Noll • Navie • 11 months ago
@Navie: As of this time the warning "$HADOOP_HOME is depreciated" is harmless. See a previous reply of mine for more information on how to fix it anyways if you care.

Nagarjuna Vissa • 4 months ago
How to install hadoop on 1000 nodes at a time. Is there any script to do?

Thanks in advance.

Sudheer • 7 months ago
hi
i run the jps command in master node it is showing only the following it is not showing the namenode
Jps
DataNode
SecondaryNameNode
after this again i reinstall the hadoop and now it is showing along with namenode after running the ./start-dfs.sh
when i run the mapreduce start-mapred.sh then i run the jps command then it is not showing the namenode
then again i reinstall the hadoop and then at this step it is not showing the datanode
can please help me

thanks for the help....

Hi Michael,
I have followed the tutorial and created a cluster with two nodes. But when I try to run word count program. Its stopping near reduce phase. It does not proceed further. I waited for 15-20 mins and the finally quit the job. I tried many times but there no change every time its getting stuck at reduce phase. Can you suggest me what I need to do.

Regards
Sundeep

Hi Michael,
Your answer to SiD on safe mode issue.
@sid: The NameNode is in Safe Mode — and this is not “a problem”. This is a normal step during the startup of the NameNode.

I followed the same with 3 nodes(1 Master, 2 Slaves). When I ran bin/start-dfs.sh and bin/start-mapred.sh all processes came up. But when i tried to run a example program, got the same error reported by @SID.

And waited for several time but no luck.

For reference I tried to execute below one-
bin/hadoop jar hadoop-examples-1.0.4.jar pi 10 100
Any suggession please going wrong here. @@When tried with single node cluster it worked fine withought any issue.

Thanks
SB.

Hi Micheal. I created a Blog entry with details how to configure 3 Virtual machines on a single laptop to create the cluster. This will help the readers who do not have more than one machine.

http://hadoopmagic.wordpress.c...
Deepak Jha • 2 months ago

Thanks for your Tutorial.

I have installed a two cluster node using two virtual instances of Ubuntu-13.04 and one as a master node and other as a slave node.

The Various hadoop Daemons are running properly on both the Virtual Os but when I am running wordcount jar file then it fails and a log file is created in the output directory stating that:

Job JOBID="job_201306071449_0005" JOBNAME="word count" USER="root" SUBMIT_TIME="1370601324238" JOBCONF="hdfs://master:54310/app/hadoop/tmp/mapred/staging/root/.staging/job_201306071449_0005/job.xml"
Job JOBID="job_201306071449_0005" JOB_PRIORITY="NORMAL".
Job JOBID="job_201306071449_0005" FINISH_TIME="1370601396277" JOB_STATUS="FAILED" FINISHED_MAPS="0" FINISHED_REDUCES="0" FAIL_REASON="Job initialization failed: java.net.UnknownHostException: Hadoop_Slave_01 is not a valid Inet address"

Michael G. Noll • a month ago

@Deepak Jha:

You have set one of your hostnames to an illegal name:

> java.net.UnknownHostException: Hadoop_Slave_01 is not a valid Inet address

Please read the error logs carefully -- you could have spotted this problem easily by yourself.

Nitin Jain • 2 months ago

Thanks Michael. Just setup a multi-node hadoop cluster with 6 machines!

Siddiqui Zeeshan • 2 months ago

2013-05-31 00:37:01,970 ERROR org.apache.hadoop.mapred.TaskTracker: Can not start task tracker because java.lang.IllegalArgumentException: Does not contain a valid host:port authority: local

I am getting this error my files are as follows

hduser@ubuntu1:/usr/local/hadoop/conf$ cat core-site.xml

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

<property>
  <name>fs.default.name</name>
  <value>hdfs://ubuntu1:9000</value>
  <description>The name of the default file system. A URI whose scheme and authority determine the FileSystem implementation. The
```

Michael G. Noll • Siddiqui Zeeshan • 2 months ago

@Siddiqui Zeeshan: Your XML snippets have been mangled by Disqus’ comment system. I tried to salvage as much of your original comment and stored it at https://gist.github.com/miguno...
From what I can tell your mapred-site.xml file is missing the `<configuration>` tag.

Thanks a lot

Siddiqui Zeeshan - Michael G. Noll - 2 months ago

You are right sir and a genius. I figured it also but by that time I had posted

Cicci Derpina Ebbaa - 3 months ago

I officially love you. Sorry for the openness :D

Da Purl - 4 months ago

Hi Micheal,

First of all thank you for the wonderful write up. Wondering if anyone has attempted to build a cluster on Amazon EC2 - using the Free tier. The ones we get for free are micro instances of 613mb memory. I just installed successfully a single node cluster following your tutorial. However, I am not sure if a cluster is possible. I am aware that I can use Whirr and install the basic cluster, but I do want to attempt to install a cluster from scratch by myself. Am not sure if its possible though.

K

Bharath - 5 months ago

First I like to thank you for publishing such a wonderful guide for hadoop installation. I have a 5 node cluster. I tried individually setting up 5 single node clusters it was working fine. Now I am trying to make a 5 node cluster. When I try to copy files from local directory to hdfs, I was getting this error.

```
copyFromLocal: org.apache.hadoop.hdfs.server.namenode.SafeModeException: Cannot create file/user/root/HadoopEbooks/EncyclopaediaBritannica.txt. Name node is in safe mode.

Then I used this command to off the safe mode.
/usr/local/hadoop/bin/hadoop dfsadmin -safemode leave

Then when I tried to copy
```

```
    /usr/local/hadoop/bin/hadoop dfs -copyFromLocal /root/Desktop/HadoopEbooks /user/root/HadoopEbooks
```

its showing this exception:

```
13/03/09 18:07:49 WARN dfs.DFSClient: DataStreamer Exception: org.apache.hadoop.ipc.RemoteException: java.io.IOException: File /user/root/HadoopEbooks/The Devil's Dictionary.txt could only be replicated to 0 nodes, instead of 1

at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getAdditionalBlock(FSNamesystem.java:1558)

at org.apache.hadoop.hdfs.server.namenode.NameNode.addBlock(NameNode.java:696)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)

<snipp>
13/03/09 18:07:49 WARN dfs.DFSClient: Error Recovery for block null bad datanode[0] nodes == null
13/03/09 18:07:49 WARN dfs.DFSClient: Could not get block locations. Source file */user/root/HadoopEbooks/The Devil's Dictionary.txt* - Aborting...

copyFromLocal: java.io.IOException: File */user/root/HadoopEbooks/The Devil's Dictionary.txt* could only be replicated to 0 nodes, instead of 1
```

Guruvarma Avvaru - Bharath - 3 months ago

I have set-up a single node cluster. When I try to `copyFromLocal`, I get the same error, what you mentioned above. Could you please help to resolve this.

www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/
Tried the following to solve the error
1) Format the namenode
2) Updated the namespaceID in Datanode version

Trying to decrease the 'Non DFS Used' space. I am unable to identify/view the files which occupy this space. My configuration details given below.

Please let me know how to identify/view the non DFS used files.

| Configured Capacity: 4497805312 (4.19 GB) |
| Present Capacity: 147365888 (140.54 MB) |
| DFS Remaining: 147333120 (140.51 MB) |
| DFS Used: 32768 (32 KB) |
| DFS Used%: 0.02% |

---

Aman Jot  •  5 months ago

Hello sir

I'm trying to implement multinode cluster on hadoop. I'm facing a problem when I give command to copy files to hdfs, it gives me the following output:

```plaintext
13/03/06 00:24:07 INFO ipc.Client: Retrying connect to server: master/192.168.159.138:54310. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1 SE
```

Please do tell what could be the problem.

Regards

---

Javier Escobar  •  5 months ago

Hi, Thank you for the tutorial.

I have a question: I have 4 virtualized ubuntu boxes, each one running on a different core (with a multicore processor). A single one is the master, and the others are the slaves. I downloaded 1GB+ of data from project Gutenberg to analyze them with the wordcount example.

When I start the cluster, with start-dfs and start-mapred commands, the system monitor of the host server (in which the ubuntu boxes are virtualized) shows that 4 cores of the processor increments its activity (1 per ubuntu box). Nevertheless, when I start the job, only the core of the master box significantly increments its activity. This is a normal behavior? Why in the starting process 4 cores increments its activity, but in the job execution it seems that only the master box is working?

Thank you very much in advance.

---

Javier Ricardo Escobar Avila  •  4 months ago

[SOLVED]

OMG!! After 2 months I solved my issue. The network configuration seemed to be wrong. My master box, in its hosts file, had the entry "127.0.1.1 'hostname' " . So, the servers that hadoop starts only listened to 1 machine (the master machine), the other machines could not connect to the master. Changing the entry with "192.168.XX.XX 'hostname' " the cluster started to work.

So, the moral is: "I MUST CAREFULLY READ THE LOGS" ;P

---

Nitesh Chauhan  •  5 months ago

This is the best post, it worked perfectly for me ;}
Thanks for the tutorials, they helped me a lot. I would like to ask about any recommendations for use of a hadoop cluster by multiple people. I would like to help people avoid stepping on each other’s toes, like accidentally deleting other’s files. So I don’t think it’s advisable to give out the hduser credentials.

Reply

Noel → Chris L. • 6 months ago

Yes... I would also like to configure hadoop to run by multiple-users and let them access ONLY their own files. (just like the /tmp files in Unix). Is there any such tutorial that I could follow? As the one one in http://titan.softnet.tuc.gr:8082/User:xenia/Page_Title/Hadoop_Cluster_Setup_Tutorial is not really working right.

Hope to hear from someone and many thanks in advanced....

Reply

Chris L. → Chris L. • 6 months ago

Thanks for the quick reply. It sounds like a tutorial on this topic would help many people!

If I understand DFS permissions correctly, anyone with network access to the cluster can submit a job, but jobs are only accepted for users with a directory in DFS (e.g., /user/chris).

The use of an edge node looks useful. For me an edge node is a computer with Hadoop binaries and suitable configuration. I think this will provide easy access (easy bcos there’s nothing to install nor configure), without granting people direct access to the cluster.

But I’ve hit many noob problems already now that I have DFS permissions enabled. The first was not using a wide-open hadoop tmp directory (apparently every newly submitted job gets copied there first). The second was not using a proper setting for the mapreduce job tracker staging area (apparently every user needs a directory there too). I suspect I’ll hit more.

Hope you’ll consider the next tutorial idea :)

Reply

Michael G. Noll → Chris L. • 6 months ago

Chris, you are right that handing out hduser credentials is ill-advised. It’s like giving the root password of a Linux server to everyone with a user account on the box. Really unpleasant things are bound to happen!

Your question would actually take a pretty long answer. But let me at least give you some basic pointers:

- create separate user/group accounts for your researchers (i.e. do not use hduser)
- enable the HDFS trash feature (to recover accidentally deleted files)
- enable HDFS permissions via “dfs.permissions.enabled”
- enable HDFS space quotas to prevent people from filling up the cluster’s storage space, thereby impacting other users; keep in mind that HDFS space quotas come with some gotchas and as I have learned the hard way they also expose you to some rather hidden bugs in the HDFS codebase, depending on the patch level / version of Hadoop you are using (like HDFS space quotas running out of sync with the actual sizes of the files in HDFS)
- use a remote metastore for Hive in multi-user environments (though Hive’s permission support is still lacking)
- in general, enable similar data permissions for related tools.
- look at what your distribution ships with; for instance, if you run Cloudera CDH3/4 you can also leverage additional tools such as the Hue UI, which provides role-based access to data/data tools
- set up monitoring for your cluster, notably to catch broken jobs (as a rule of thumb any jobs running for 24+ hours is broken) that block computing resources
- enable the preemption feature of the FairScheduler if your Hadoop cluster is using that scheduler (and if your version of FairScheduler supports preemption, not all do); preemption can take away resources from running jobs of user A and grant those resources to user B

Hope this helps!
is not really working right.

Hope to hear from someone and many thanks in advanced.....

Walter • 7 months ago
Hi, the tutorial is excellent, you helped me getting started with this complex matter.

A small detail, could it be that in the "Workaround 1: Start from scratch" instead of looking for the directory in the file called "conf/hdfs-site.xml", it is the directory in the file "conf/core-site.xml"? Because in the hdfs-site.xml, as you explained, there are the amount of replicas for a block. Thanks again!

Panshul Gupta • 7 months ago
Awesome Tutorial,

As u said:

Adding nodes on the fly are not included in the tutorial YET...

any suggestions by when will you be adding a tutorial for adding nodes on the fly?

or will you be adding it at all?

it is explained in the Apache guide but it is much easier and clear when you explain it in a much better way ;)

Regards,

Raf • 7 months ago
Hi,
Can someone tell me what's the difference between running Hadoop the way mention & creating a MapReduceJob from AWS console?

I am a bit confused.

Thanks

Raf

Hegazi Abdel Hakim Abdel Rahma • 7 months ago
Solved my many problems ... Thanks so much

Merry Xmas :-)
Avinash G A • 7 months ago
this is a lovely article. good job mike.....

souri • 8 months ago
Nicely explained, also refer this YouTube video: Creating Hadoop 2-Node Cluster.

Paddu • 8 months ago
Hi Tinniam V Ganesh, Actually the same issue i got when i configured my two nodes. it is because of firewall issue. go to your control panel and turn off firewall settings.

jaoki • 8 months ago
great article! It solved my day :)

kishore • 8 months ago
Thank you for such a thorough tutorial.

This tutorial helps much more.....

however can please let me know how can access these information using eclipse environment --any tutorial link of ours let me know i will follow the same.

arihant • 8 months ago
Sir, can we connect the two nodes directly with a lan cable without a hub.will there be any change in /etc/hosts file if node are connected only with each other(i.e. no hub).

Tinniam V Ganesh • 8 months ago
Hi,
Your posts on Single node & multi-node are really great.. However, I am getting the following error - master: ssh: connect to host master port 22: No route to host. I followed all your steps. Unable to figure out the reason. Please help.

Thanks
Ganesh

stholy • 9 months ago
Hi,
I dont know whether its a silly question, yet I am still asking:

Can we a have a cluster where the master and slave have different user accounts which have the hadoop installation??

eg: on master: hduser
on slave: hduser_s

How would we configure this cluster ???

Anju Singh • 9 months ago
hello sir,

i m running hadoop cluster with 3 datanode, but it is showing me only one in its web interface.
plz sir help me.

Kanishk Verma  •  9 months ago
Hi,
Nice Tutorial.
Keep up the good work!
Cheers,
Kanishk

Karthikeya  •  9 months ago
Hi Machael,

This page has helped me a lot in setting up the cluster and understanding the hadoop environment.

Thanks a lot!!

Anju Singh  •  9 months ago
thank u sir,
it helped me a lot.

ashish  •  9 months ago

Hi Michael,

I configured as explained above then was finding the error in my case couldn't found where I was putting wrong configuration . But as I added following line It worked...

mapred.task.tracker
conf/slaves
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.

Please comment!

Regards

ashish  •  10 months ago

Trying above @ two machines - 1 master (red hat) 1 slave (red hat)

at org.apache.hadoop.net.NetUtils.createServerAddr(NetUtils.java:162)
at org.apache.hadoop.mapred.JobTracker.getAddress(JobTracker.java:2568)
at org.apache.hadoop.mapred.JobTracker.(JobTracker.java:2200)
at org.apache.hadoop.mapred.JobTracker.(JobTracker.java:2192)
at org.apache.hadoop.mapred.JobTracker.(JobTracker.java:2186)
Please help?

@ashish:

java.lang.IllegalArgumentException: Does not contain a valid host:port authority: local

Seems like you have a configuration problem, see line above.
About Me

I am a researcher and software engineer based in Switzerland, Europe. I work for the .COM and .NET DNS registry operator Verisign as the technical lead of its large-scale computing infrastructure based on the Apache Hadoop stack and as a research affiliate at Verisign Labs. Read more »

Contact

michael@michael-noll.com

Follow Me

@miguno
Blog RSS
GitHub

Recent Posts

- Using Avro in MapReduce Jobs with Hadoop, Pig, Hive
- Understanding the Internal Message Buffers of Storm
- Installing and Running Graphite via RPM and Supervisord
- Multi-Node Storm Cluster Tutorial Published
- Reading and Writing Avro Files from the Command Line

Latest Tweets

- Status updating...

Copyright © 2004-2013 Michael G. Noll. All rights reserved. Privacy Policy. Powered by Octopress.