Go

- Home
- About
- Contact
- Blog
- Code
- Publications
- DMOZ100k06
- Photography

Running Hadoop On Ubuntu Linux (Multi-Node Cluster)

From Michael G. Noll

Contents

What we want to do

In this tutorial, I will describe the required steps for setting up a *multi-node* Hadoop (http://lucene.apache.org/hadoop/) cluster using the Hadoop Distributed File System (HDFS) (http://lucene.apache.org/hadoop/hdfs_design.html) on Ubuntu Linux (http://www.ubuntu.com/) .

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

Hadoop (http://lucene.apache.org/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 (http://en.wikipedia.org/wiki/Google_File_System) and of MapReduce (http://en.wikipedia.org/wiki/MapReduce) . HDFS (http://lucene.apache.org/hadoop/hdfs_design.html) is a highly fault-tolerant distributed file system and like Hadoop 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.

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

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 (http://www.ubuntu.com/) 7.10, 7.04
- Hadoop (http://lucene.apache.org/hadoop/) 0.15.2, released January 2008 (also works with 0.14.x and 0.13.x)

You can find the time of the last document update at the bottom of this page.

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.

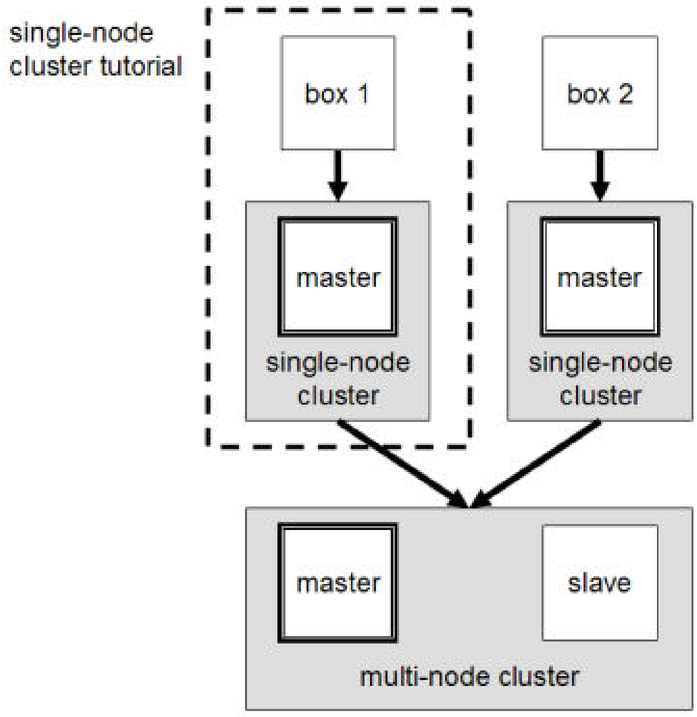

Figure 2: Tutorial approach and structure.

Prerequisites

Configuring single-node clusters first

The tutorial approach outlined above means that you should read now my previous tutorial on how to setup up 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's 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*.

We will call the designated master machine just the `master` from now on and the slave-only machine the `slave`.

Shutdown each single-node cluster with `<HADOOP_INSTALL>/bin/stop-all.sh` before continuing if you haven't done so already.

Networking

This should come as no 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)
192.168.0.1    master
192.168.0.2    slave
```

SSH access

The `hadoop` user on the `master` (aka `hadoop@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 `hadoop` user account on the `slave` (aka `hadoop@slave`) via a password-less SSH login. If you followed my single-node cluster tutorial, you just have to add the `hadoop@master`'s public SSH key (which should be in `$HOME/.ssh/id_rsa.pub`) to the `authorized_keys` file of `hadoop@slave` (in this user's `$HOME/.ssh/authorized_keys`).

The final step is to test the SSH setup by connecting with user `hadoop` from the `master` to the user account `hadoop` on the `slave`. The step is also needed to save `slave`'s host key fingerprint to the `hadoop@master`'s </tt>known_hosts</tt> file.

So, connecting from `master` to `master`...

```
hadoop@master:~$ ssh master
The authenticity of host 'master (192.168.0.1)' can't be established.
RSA key fingerprint is 3b:21:b3:c0:21:5c:7c:54:2f:1e:2d:96:79:eb:7f:95.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master' (RSA) to the list of known hosts.
Linux master 2.6.20-16-386 #2 Thu Jun 7 20:16:13 UTC 2007 i686
...
hadoop@master:~$
```

...and from `master` to `slave`.

```
hadoop@master:~$ ssh slave
The authenticity of host 'slave (192.168.0.2)' can't be established.
RSA key fingerprint is 74:d7:61:86:db:86:8f:31:90:9c:68:b0:13:88:52:72.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave' (RSA) to the list of known hosts.
Ubuntu 7.04
...
hadoop@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.

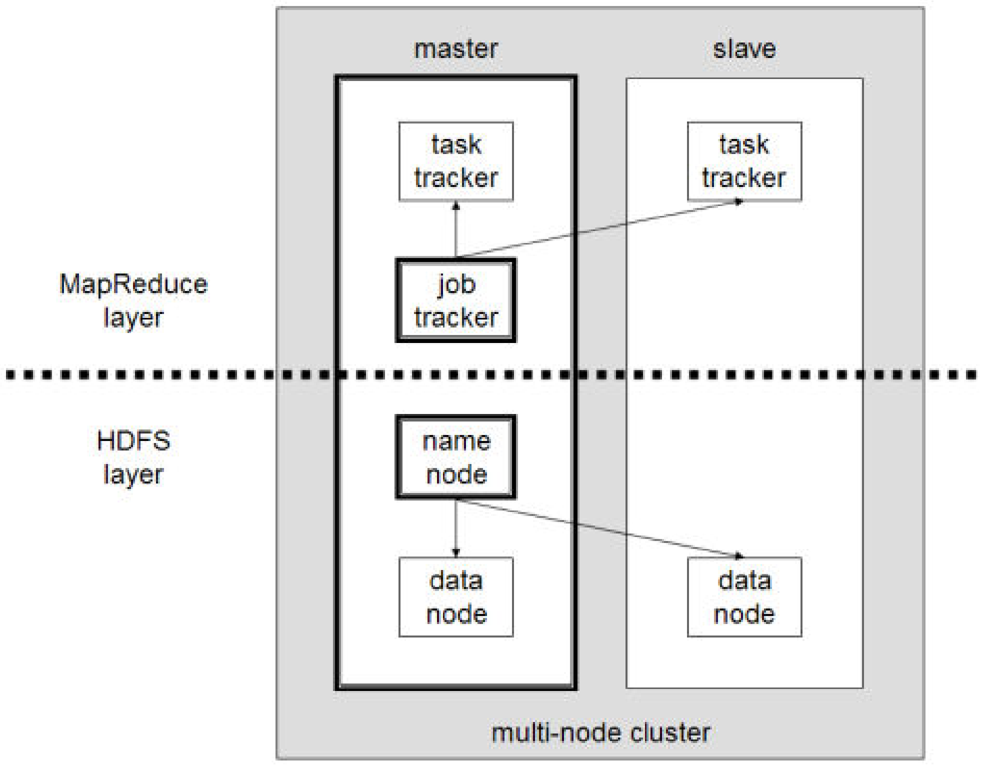

Figure 3: How the final multi-node cluster will look like.

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.

Configuration

conf/masters (`master` only)

The `conf/masters` file defines the master nodes of our multi-node cluster. In our case, this is just the `master` machine.

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

```
master
```

conf/slaves (`master` only)

This `conf/slaves` file lists the hosts, one per line, where the Hadoop slave daemons (datanodes and tasktrackers) will 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 `<HADOOP_INSTALL>/conf/slaves` that it looks like this:

```
master
slave
```

If you have additional slave nodes, just add them to the `conf/slaves` file, one per line (do this on all machines in the cluster).

```
master
slave
anotherslave01
anotherslave02
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 --config <config_path> start datanode`. Using the `conf/slaves` file on the master simply helps you to make "full" cluster restarts easier.

conf/hadoop-site.xml (all machines)

Assuming you configured `conf/hadoop-site.xml` on each machine as described in the single-node cluster tutorial, you will only have to change a few variables.

Important: You have to change `conf/hadoop-site.xml` on ALL machines as follows.

First, we have to change the fs.default.name (http://lucene.apache.org/hadoop/hadoop-default.html#fs.default.name) variable which specifies the NameNode (http://lucene.apache.org/hadoop/api/overview-summary.html) (the HDFS master) host and port. In our case, this is the `master` machine.

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

Second, we have to change the mapred.job.tracker (http://lucene.apache.org/hadoop/hadoop-default.html#mapred.job.tracker) variable which specifies the JobTracker (http://lucene.apache.org/hadoop/api/org/apache/hadoop/mapred/JobTracker.html) (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 (http://lucene.apache.org/hadoop/hadoop-default.html#dfs.replication) variable 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 slave nodes (more precisely, the number of datanodes) that you have available, you will start seeing a lot of `(Zero targets found, forbidden1.size=1)` type errors in the log files.

The default value of `dfs.replication` is `3`. However, we have only two nodes available, so we set `dfs.replication` to `2`.

```
<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://lucene.apache.org/hadoop/api/overview-summary.html) (see bottom of page).

`conf/hadoop-site.xml`

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

`conf/mapred-default.xml`
(note that there is no `conf/mapred-site.xml` file!)

`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 2x the number of slave processors (i.e., number of tasktrackers).

Formatting the namenode

Before we start our new multi-node cluster, we have to format Hadoop's distributed filesystem (HDFS) for the namenode. You need to do this the first time you set up a Hadoop cluster. Do not format a running Hadoop namenode, this will cause all your data in the HDFS filesytem to be erased.

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

```
hadoop@master:/usr/local/hadoop$ bin/hadoop namenode -format
... INFO dfs.Storage: Storage directory /usr/local/hadoop-datastore/hadoop-hadoop/dfs/name has been success
hadoop@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

Starting the cluster is done in two steps. First, the HDFS daemons are started: the namenode daemon is started on `master`, and datanode daemons are started on all slaves (here: `master` and `slave`). Second, the MapReduce daemons are started: the jobtracker is started on `master`, and tasktracker daemons are started on all slaves (here: `master` and `slave`).

HDFS daemons

Run the command `<HADOOP_INSTALL>/bin/start-dfs.sh` on the machine you want the 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`:

```
hadoop@master:/usr/local/hadoop$ bin/start-dfs.sh
starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-namenode-master.out
slave: Ubuntu 7.04
slave: starting datanode, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-datanode-slave.out
master: starting datanode, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-datanode-master.out
master: starting secondarynamenode, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-secondarynamenod
hadoop@master:/usr/local/hadoop$
```

On `slave`, you can examine the success or failure of this command by inspecting the log file `<HADOOP_INSTALL>/logs/hadoop-hadoop-datanode-slave.log`. Exemplary output:

```
... INFO org.apache.hadoop.dfs.Storage: Storage directory /usr/local/hadoop-datastore/hadoop-hadoop/dfs/dat
... INFO org.apache.hadoop.dfs.Storage: Formatting ...
... INFO org.apache.hadoop.dfs.DataNode: Opened server at 50010
... INFO org.mortbay.util.Credential: Checking Resource aliases
... INFO org.mortbay.http.HttpServer: Version Jetty/5.1.4
... INFO org.mortbay.util.Container: Started org.mortbay.jetty.servlet.WebApplicationHandler@17a8a02
... INFO org.mortbay.util.Container: Started WebApplicationContext[/,/]
... INFO org.mortbay.util.Container: Started HttpContext[/logs,/logs]
... INFO org.mortbay.util.Container: Started HttpContext[/static,/static]
... INFO org.mortbay.http.SocketListener: Started SocketListener on 0.0.0.0:50075
... INFO org.mortbay.util.Container: Started org.mortbay.jetty.Server@56a499
... INFO org.apache.hadoop.dfs.DataNode: Starting DataNode in: FSDataset{dirpath='/usr/local/hadoop-datasto
... INFO org.apache.hadoop.dfs.DataNode: using BLOCKREPORT_INTERVAL of 3538203msec
```

As you can see in `slave`'s output above, it will automatically format it's storage directory (specified by `dfs.data.dir`) 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`...

```
hadoop@master:/usr/local/hadoop$ jps
14799 NameNode
15314 Jps
14880 DataNode
14977 SecondaryNameNode
hadoop@master:/usr/local/hadoop$
```

(the process IDs don't matter of course)

...and the following on `slave`.

```
hadoop@slave:/usr/local/hadoop$ jps
15183 DataNode
15616 Jps
hadoop@slave:/usr/local/hadoop$
```

MapReduce daemons

Run the command `<HADOOP_INSTALL>/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`:

```
hadoop@master:/usr/local/hadoop$ bin/start-mapred.sh
starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-jobtracker-master.out
slave: Ubuntu 7.04
slave: starting tasktracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-tasktracker-slave.out
master: starting tasktracker, logging to /usr/local/hadoop/bin/../logs/hadoop-hadoop-tasktracker-master.out
hadoop@master:/usr/local/hadoop$
```

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

```
... INFO org.mortbay.util.Credential: Checking Resource aliases
... INFO org.mortbay.http.HttpServer: Version Jetty/5.1.4
... INFO org.mortbay.util.Container: Started org.mortbay.jetty.servlet.WebApplicationHandler@d19bc8
... INFO org.mortbay.util.Container: Started WebApplicationContext[/,/]
... INFO org.mortbay.util.Container: Started HttpContext[/logs,/logs]
... INFO org.mortbay.util.Container: Started HttpContext[/static,/static]
... INFO org.mortbay.http.SocketListener: Started SocketListener on 0.0.0.0:50060
... INFO org.mortbay.util.Container: Started org.mortbay.jetty.Server@1e63e3d
... INFO org.apache.hadoop.ipc.Server: IPC Server listener on 50050: starting
... INFO org.apache.hadoop.ipc.Server: IPC Server handler 0 on 50050: starting
... INFO org.apache.hadoop.mapred.TaskTracker: TaskTracker up at: 50050
... INFO org.apache.hadoop.mapred.TaskTracker: Starting tracker tracker_slave:50050
... INFO org.apache.hadoop.ipc.Server: IPC Server handler 1 on 50050: starting
... INFO org.apache.hadoop.mapred.TaskTracker: Starting thread: Map-events fetcher for all reduce tasks on
```

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

```
hadoop@master:/usr/local/hadoop$ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
hadoop@master:/usr/local/hadoop$
```

(the process IDs don't matter of course)

...and the following on `slave`.

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

Stopping the multi-node cluster

Like starting the cluster, stopping it is done in two steps. The workflow is the opposite of starting, however. First, 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`). Second, the HDFS daemons are stopped: the namenode daemon is stopped on `master`, and datanode daemons are stopped on all slaves (here: `master` and `slave`).

MapReduce daemons

Run the command `<HADOOP_INSTALL>/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`:

```
hadoop@master:/usr/local/hadoop$ bin/stop-mapred.sh
stopping jobtracker
slave: Ubuntu 7.04
master: stopping tasktracker
slave: stopping tasktracker
hadoop@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`)

On `slave` (strangely, and for a reason unknown to me), you will not see any information in the log file `<HADOOP_INSTALL>/logs/hadoop-hadoop-tasktracker-slave.log` that the datanode daemon has been shut down. However, you can list the running Java processes with `jps` as a workaround as you can see below.

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

```
hadoop@master:/usr/local/hadoop$ jps
14799 NameNode
18386 Jps
14880 DataNode
14977 SecondaryNameNode
hadoop@master:/usr/local/hadoop$
```

...and the following on `slave`.

```
hadoop@slave:/usr/local/hadoop$ jps
15183 DataNode
18636 Jps
hadoop@slave:/usr/local/hadoop$
```

HDFS daemons

Run the command `<HADOOP_INSTALL>/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`:

```
hadoop@master:/usr/local/hadoop$ bin/stop-dfs.sh
stopping namenode
slave: Ubuntu 7.04
slave: stopping datanode
master: stopping datanode
master: stopping secondarynamenode
hadoop@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`)

On `slave` (again strangely, and again for a reason unknown to me), you will not see any information in the log file `<HADOOP_INSTALL>/logs/hadoop-hadoop-datanode-slave.log` that the datanode daemon has been shut down. However, you can list the running Java processes with `jps` as a workaround as you can see below.

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

```
hadoop@master:/usr/local/hadoop$ jps
18670 Jps
hadoop@master:/usr/local/hadoop$
```

...and the following on `slave`.

```
hadoop@slave:/usr/local/hadoop$ jps
18894 Jps
hadoop@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 (http://www.gutenberg.org/etext/20417)
- The Notebooks of Leonardo Da Vinci (http://www.gutenberg.org/etext/5000)
- Ulysses by James Joyce (http://www.gutenberg.org/etext/4300)
- The Art of War by 6th cent. B.C. Sunzi (http://www.gutenberg.org/etext/132)
- The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle (http://www.gutenberg.org/etext/1661)
- The Devil's Dictionary by Ambrose Bierce (http://www.gutenberg.org/etext/972)
- Encyclopaedia Britannica, 11th Edition, Volume 4, Part 3 (http://www.gutenberg.org/etext/19699)

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 exemplary output on `master`...

```
hadoop@master:/usr/local/hadoop$ bin/hadoop jar hadoop-0.13.0-examples.jar wordcount  gutenberg gutenberg-ou
... INFO mapred.FileInputFormat: Total input paths to process : 7
... INFO mapred.JobClient: Running job: job_0001
... INFO mapred.JobClient:  map 0% reduce 0%
... INFO mapred.JobClient:  map 28% reduce 0%
... INFO mapred.JobClient:  map 57% reduce 0%
... INFO mapred.JobClient:  map 71% reduce 0%
... INFO mapred.JobClient:  map 100% reduce 9%
... INFO mapred.JobClient:  map 100% reduce 68%
... INFO mapred.JobClient:  map 100% reduce 100%
.... INFO mapred.JobClient: Job complete: job_0001
... INFO mapred.JobClient: Counters: 11
... INFO mapred.JobClient:   org.apache.hadoop.examples.WordCount$Counter
... INFO mapred.JobClient:     WORDS=1173099
... INFO mapred.JobClient:     VALUES=1368295
... INFO mapred.JobClient:   Map-Reduce Framework
... INFO mapred.JobClient:     Map input records=136582
... INFO mapred.JobClient:     Map output records=1173099
... INFO mapred.JobClient:     Map input bytes=6925391
... INFO mapred.JobClient:     Map output bytes=11403568
... INFO mapred.JobClient:     Combine input records=1173099
... INFO mapred.JobClient:     Combine output records=195196
... INFO mapred.JobClient:     Reduce input groups=131275
... INFO mapred.JobClient:     Reduce input records=195196
... INFO mapred.JobClient:     Reduce output records=131275
hadoop@master:/usr/local/hadoop$
```

...and on `slave` for its datanode...

```
# from <HADOOP_INSTALL>/logs/hadoop-hadoop-datanode-slave.log on slave
... INFO org.apache.hadoop.dfs.DataNode: Received block blk_5693969390309798974 from  /192.168.0.1
... INFO org.apache.hadoop.dfs.DataNode: Received block blk_7671491277162757352 from /192.168.0.1
<<<SNIPP>>>
... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-7112133651100166921 to /192.168.0.2
... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-7545080504225510279 to /192.168.0.2
... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-4114464184254609514 to /192.168.0.2
... INFO org.apache.hadoop.dfs.DataNode: Served block blk_-4561652742730019659 to /192.168.0.2
<<<SNIPP>>>
... INFO org.apache.hadoop.dfs.DataNode: Received block blk_-2075170214887808716 from /192.168.0.2 and mirro
... INFO org.apache.hadoop.dfs.DataNode: Received block blk_1422409522782401364 from /192.168.0.2 and mirror
... INFO org.apache.hadoop.dfs.DataNode: Deleting block blk_-2942401177672711226 file /home/hadoop/hadoop-da
... INFO org.apache.hadoop.dfs.DataNode: Deleting block blk_-3019298164878756077 file /home/hadoop/hadoop-da
```

...and on `slave` for its tasktracker.

```
# from <HADOOP_INSTALL>/logs/hadoop-hadoop-tasktracker-slave.log on slave
... INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction: task_0001_m_000000_0
... INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction: task_0001_m_000001_0
... task_0001_m_000001_0 0.08362164% hdfs://master:54310/user/hadoop/gutenberg/ulyss12.txt:0+1561677
... task_0001_m_000000_0 0.07951202% hdfs://master:54310/user/hadoop/gutenberg/19699.txt:0+1945731
<<<SNIPP>>>
... task_0001_m_000001_0 0.35611463% hdfs://master:54310/user/hadoop/gutenberg/ulyss12.txt:0+1561677
... Task task_0001_m_000001_0 is done.
... task_0001_m_000000_0 1.0% hdfs://master:54310/user/hadoop/gutenberg/19699.txt:0+1945731
... LaunchTaskAction: task_0001_m_000006_0
... LaunchTaskAction: task_0001_r_000000_0
... task_0001_m_000000_0 1.0% hdfs://master:54310/user/hadoop/gutenberg/19699.txt:0+1945731
... Task task_0001_m_000000_0 is done.
... task_0001_m_000006_0 0.6844295% hdfs://master:54310/user/hadoop/gutenberg/132.txt:0+343695
... task_0001_r_000000_0 0.095238104% reduce > copy (2 of 7 at 1.68 MB/s) >
... task_0001_m_000006_0 1.0% hdfs://master:54310/user/hadoop/gutenberg/132.txt:0+343695
... Task task_0001_m_000006_0 is done.
... task_0001_r_000000_0 0.14285716% reduce > copy (3 of 7 at 1.02 MB/s) >
<<<SNIPP>>>
... task_0001_r_000000_0 0.14285716% reduce > copy (3 of 7 at 1.02 MB/s) >
... task_0001_r_000000_0 0.23809525% reduce > copy (5 of 7 at 0.32 MB/s) >
... task_0001_r_000000_0 0.6859089% reduce > reduce
... task_0001_r_000000_0 0.7897389% reduce > reduce
... task_0001_r_000000_0 0.86783284% reduce > reduce
... Task task_0001_r_000000_0 is done.
... Received 'KillJobAction' for job: job_0001
... task_0001_r_000000_0 done; removing files.
... task_0001_m_000000_0 done; removing files.
... task_0001_m_000006_0 done; removing files.
... task_0001_m_000001_0 done; removing files.
```

If you want to inspect the job's output data, just retrieve the job result from HDFS to your local filesystem.

Caveats

java.io.IOException: Incompatible namespaceIDs

If you see the error `java.io.IOException: Incompatible namespaceIDs` in the logs of a datanode (`<HADOOP_INSTALL>/logs/hadoop-hadoop-datanode-<HOSTNAME>.log`), chances are you are affected by bug HADOOP-1212 (http://issues.apache.org/jira/browse/HADOOP-1212) (well, I've been affected by it at least).

The full error looked like this on my machines:

```
... ERROR org.apache.hadoop.dfs.DataNode: java.io.IOException: Incompatible namespaceIDs in /usr/local/hado
      at org.apache.hadoop.dfs.DataStorage.doTransition(DataStorage.java:281)
      at org.apache.hadoop.dfs.DataStorage.recoverTransitionRead(DataStorage.java:121)
      at org.apache.hadoop.dfs.DataNode.startDataNode(DataNode.java:230)
      at org.apache.hadoop.dfs.DataNode.<init>(DataNode.java:199)
      at org.apache.hadoop.dfs.DataNode.makeInstance(DataNode.java:1202)
      at org.apache.hadoop.dfs.DataNode.run(DataNode.java:1146)
      at org.apache.hadoop.dfs.DataNode.createDataNode(DataNode.java:1167)
      at org.apache.hadoop.dfs.DataNode.main(DataNode.java:1326)
```

The crude workaround I have found is to:

1. stop the cluster
2. delete the data directory on the problematic datanode: the directory is specified by `dfs.data.dir` in `conf/hadoop-site.xml`; if you followed this tutorial, the relevant directory is `/usr/local/hadoop-datastore/hadoop-hadoop/dfs/data`
3. reformat the namenode (NOTE: all HDFS data is lost during this process!)
4. restart the cluster

For more information regarding this issue, read the bug description (http://issues.apache.org/jira/browse/HADOOP-1212) .

What's next?

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.

Feedback

Comments, questions and constructive feedback are always welcome. Just drop me a note.

Related Links

- Running Hadoop On Ubuntu Linux (Single-Node Cluster)
- Writing An Hadoop MapReduce Program In Python
- Hadoop home page (http://lucene.apache.org/hadoop/)
- Project Description (http://wiki.apache.org/lucene-hadoop/ProjectDescription) @ Hadoop Wiki
- Getting Started with Hadoop (http://wiki.apache.org/lucene-hadoop/GettingStartedWithHadoop) @ Hadoop Wiki
- How to debug MapReduce programs (http://wiki.apache.org/lucene-hadoop/HowToDebugMapReducePrograms) @ Hadoop Wiki
- Hadoop API Overview (http://lucene.apache.org/hadoop/api/overview-summary.html)
- Nutch Hadoop Tutorial (http://wiki.apache.org/nutch/NutchHadoopTutorial) @ Nutch Wiki
- Bug HADOOP-1212: Data-nodes should be formatted when the name-node is formatted (http://issues.apache.org/jira/browse/HADOOP-1212) @ Hadoop Bugzilla
- Bug HADOOP-1374: TaskTracker falls into an infinite loop (http://issues.apache.org/jira/browse/HADOOP-1374) @ Hadoop Bugzilla
- Adding datanodes on the fly (http://www.nabble.com/adding-datanodes-on-the-fly--t4097939.html) @ Hadoop Users mailing list
- Running Hadoop MapReduce on Amazon EC2 and Amazon S3 (http://developer.amazonwebservices.com/connect/entry.jspa?externalID=873)
- Nutch and Hadoop (as user with NFS) (http://joey.mazzarelli.com/2007/07/25/nutch-and-hadoop-as-user-with-nfs/)

Tags: apache, cluster, foss, google, hadoop, howto, large-scale, linux, lucene, mapreduce, node, open source, scalability, screenshot, screenshots, multi-node, tutorial, ubuntu
Retrieved from
"http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_%28Multi-Node_Cluster%29"

Categories: Ubuntu | Hadoop | Tutorials & HOWTOs

- Article |
- Discussion |
- View source |
- History

- Log in / create account

- What links here |
- Related changes |
- Upload file |
- Special pages
- | Permanent link

This page was last modified 10:28, 9 January 2008. This page has been accessed 3,501 times.