operational modes of Hadoop and example configuration files for each.

2.3.1 Local (standalone) mode

The standalone mode is the default mode for Hadoop. When you first uncompress the Hadoop source package, it's ignorant of your hardware setup. Hadoop chooses to be conservative and assumes a minimal configuration. All three XML files (or hadoop-site.xml before version 0.20) are empty under this default mode:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

</configuration>
```

With empty configuration files, Hadoop will run completely on the local machine. Because there's no need to communicate with other nodes, the standalone mode doesn't use HDFS, nor will it launch any of the Hadoop daemons. Its primary use is for developing and debugging the application logic of a MapReduce program without the additional complexity of interacting with the daemons. When you ran the example MapReduce program in chapter 1, you were running it in standalone mode.

2.3.2 Pseudo-distributed mode

The pseudo-distributed mode is running Hadoop in a "cluster of one" with all daemons running on a single machine. This mode complements the standalone mode for debugging your code, allowing you to examine memory usage, HDFS input/output issues, and other daemon interactions. Listing 2.1 provides simple XML files to configure a single server in this mode.

configure a single server in this mode.

Listing 2.1 Example of the three configuration files for pseudo-distributed mode

core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

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

</configuration>
```

mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
 <name>mapred.job.tracker</name>
 <value>localhost:9001</value>
 <description>The host and port that the MapReduce job tracker runs
 at.</description>
</property>

</configuration>
```

hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
 <name>dfs.replication</name>
 <value>1</value>
 <description>The actual number of replications can be specified when the
 file is created.</description>
</property>

</configuration>
```

In `core-site.xml` and `mapred-site.xml` we specify the hostname and port of the NameNode and the JobTracker, respectively. In `hdfs-site.xml` we specify the default replication factor for HDFS, which should only be one because we're running on only one node. We must also specify the location of the Secondary NameNode in the masters file and the slave nodes in the slaves file:

```
[hadoop-user@master]$ cat masters
localhost
[hadoop-user@master]$ cat slaves
localhost
```

While all the daemons are running on the same machine, they still communicate with each other using the same SSH protocol as if they were distributed over a cluster. Section 2.2 has a more detailed discussion of setting up the SSH channels, but for single-node operation simply check to see if your machine already allows you to `ssh` back to itself.

```
[hadoop-user@master]$ ssh localhost
```

If it does, then you're good. Otherwise setting up takes two lines.

```
[hadoop-user@master]$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
[hadoop-user@master]$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

You are almost ready to start Hadoop. But first you'll need to format your HDFS by using the command

```
[hadoop-user@master]$ bin/hadoop namenode -format
```

We can now launch the daemons by use of the `start-all.sh` script. The Java `jps` command will list all daemons to verify the setup was successful.

```
[hadoop-user@master]$ bin/start-all.sh
[hadoop-user@master]$ jps
26893 Jps
26832 TaskTracker
26620 SecondaryNameNode
26333 NameNode
26484 DataNode
26703 JobTracker
```

When you've finished with Hadoop you can shut down the Hadoop daemons by the command

```
[hadoop-user@master]$ bin/stop-all.sh
```

Both standalone and pseudo-distributed modes are for development and debugging purposes. An actual Hadoop cluster runs in the third mode, the fully distributed mode.

.3 Fully distributed mode

After continually emphasizing the benefits of distributed storage and distributed computation, it's time for us to set up a full cluster. In the discussion below we'll use the following server names:

- *master*—The master node of the cluster and host of the NameNode and JobTracker daemons

- *backup*—The server that hosts the Secondary NameNode daemon
- *hadoop1, hadoop2, hadoop3, ...*—The slave boxes of the cluster running both DataNode and TaskTracker daemons

Using the preceding naming convention, listing 2.2 is a modified version of the pseudo-distributed configuration files (listing 2.1) that can be used as a skeleton for your cluster's setup.

Listing 2.2 Example configuration files for fully distributed mode

core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
 <name>fs.default.name</name>
```


1 Locate NameNode

for filesystem

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

</configuration>
```

mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
```

```
<property>
 <name>mapred.job.tracker</name>
 <value>master:9001</value>
 <description>The host and port that the MapReduce job tracker runs
 at.</description>
</property>

</configuration>
```

2 **Locate JobTracker master**

hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
 <name>dfs.replication</name>
 <value>3</value>
 <description>The actual number of replications can be specified when the
```

3 **Increase HDFS replication factor**

```
 file is created.</description>
</property>

</configuration>
```

The key differences are

- We explicitly stated the hostname for location of the NameNode (1) and JobTracker (2) daemons.
- We increased the HDFS replication factor to take advantage of distributed storage (3). Recall that data is replicated across HDFS to increase availability and reliability.

We also need to update the masters and slaves files to reflect the locations of the other daemons.

```
[hadoop-user@master]$ cat masters
backup
[hadoop-user@master]$ cat slaves
hadoop1
```

```
[hadoop-user@master]$ cat slaves
hadoop1
hadoop2
hadoop3
...
```

Once you have copied these files across all the nodes in your cluster, be sure to format HDFS to prepare it for storage:

```
[hadoop-user@master]$ bin/hadoop namenode-format
```

Now you can start the Hadoop daemons:

```
[hadoop-user@master]$ bin/start-all.sh
```

and verify the nodes are running their assigned jobs.

```
[hadoop-user@master]$ jps
30879 JobTracker
30717 NameNode
```

```
30965 Jps
[hadoop-user@backup]$ jps
2099 Jps
1679 SecondaryNameNode
[hadoop-user@hadoop1]$ jps
7101 TaskTracker
7617 Jps
6988 DataNode
```

You have a functioning cluster!

Switching between modes

A practice that I found useful when starting with Hadoop was to use symbolic links to switch between Hadoop modes instead of constantly editing the XML files. To do so, create a separate configuration folder for each of the modes and place the appropriate version of the XML files in the corresponding folder. Below is an example directory listing:

```
[hadoop@hadoop_master hadoop]$ ls -l
total 4884
drwxr-xr-x  2 hadoop-user hadoop  4096 Nov 26 17:36 bin
-rw-rw-r--  1 hadoop-user hadoop  57430 Nov 13 19:09 build.xml
drwxr-xr-x  4 hadoop-user hadoop  4096 Nov 13 19:14 c++
-rw-rw-r--  1 hadoop-user hadoop 287046 Nov 13 19:09 CHANGES.txt
lrwxrwxrwx  1 hadoop-user hadoop   12 Jan 5 16:06 conf -> conf.cluster
drwxr-xr-x  2 hadoop-user hadoop  4096 Jan 8 17:05 conf.cluster
drwxr-xr-x  2 hadoop-user hadoop  4096 Jan 2 15:07 conf.pseudo
drwxr-xr-x  2 hadoop-user hadoop  4096 Dec 1 10:10 conf.standalone
drwxr-xr-x 12 hadoop-user hadoop  4096 Nov 13 19:09 contrib
```

```
drwxrwxr-x  5 hadoop-user hadoop  4096 Jan 2 09:28 datastore

drwxr-xr-x  6 hadoop-user hadoop  4096 Nov 26 17:36 docs

...
```

You can then switch between configurations by using the Linux `ln` command (e.g., `ln -s conf.cluster conf`). This practice is also useful to temporarily pull a node out of the cluster to debug a MapReduce program in pseudo-distributed mode, but be sure that the modes have different file locations for HDFS and stop all daemons on the node before changing configurations.

Now that we've gone through all the settings to successfully get a Hadoop cluster up and running, we'll introduce the Web UI for basic monitoring of the cluster's state.