**Fundamentals of Hadoop**

**What is Hadoop?**

*“The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.”*

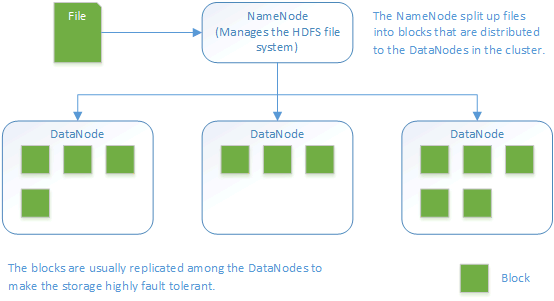
– <http://hadoop.apache.org/>

**Components of Hadoop**

Hadoop is built up by a number of components and Open Source frameworks which makes it quite flexible and modular. However before diving deeper into Hadoop it is easier to view it as two main parts – data storage (HDFS) and data processing (MapReduce):

* HDFS – Hadoop Distributed File System  
  The Hadoop Distributed File System (HDFS) was designed to run on low cost hardware and is higly fault tolerant. Files are split up into blocks that are replicated to the DataNodes. By default blocks have a size of 64MB and are replicated to 3 nodes in the cluster. However those settings can be adjusted to specific needs.

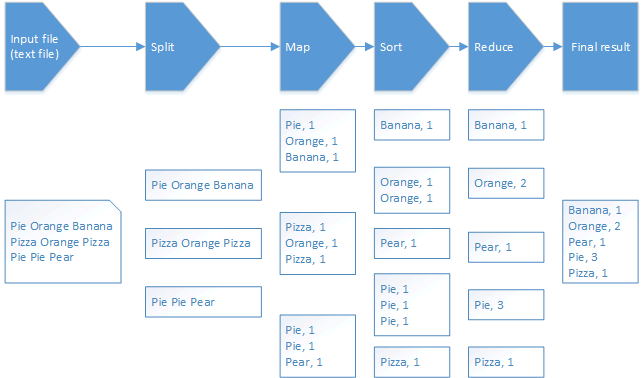
Overview of HDFS File System architecture:

[](http://www.widriksson.com/wp-content/uploads/2014/10/Hadoop-HDFS.png)

* MapReduce  
  MapReduce is a software framework written in Java that is used to create application that can process large amount of data. Although its written in Java there are other languages available to write a MapReduce application. As with HDFS it is built to be fault tolerant and to work in large-scale cluster environments. The framework have the ability to split up input data into smaller tasks (map tasks) that can be executed in parallel processes. The output from the map tasks are then reduced (reduce task) and usually saved to the file system.

Below you will see the MapReduce flow of the WordCount sample program that we will use later. WordCount takes a text file as input, divides it into smaller parts and then count each word and outputs a file with a count of all words within the file.

MapReduce flow overview (WordCount example):

[](http://www.widriksson.com/wp-content/uploads/2014/10/Hadoop-MapReduce-WordCount.png)

**Daemons/services**

|  |  |
| --- | --- |
| **Daemon/service** | **Description** |
| NameNode | Runs on a Master node. Manages the HDFS file system on the cluster. |
| Secondary NameNode | Very misleading name. It is NOT a backup for the NameNode. It make period checks/updates so in case the NameNode fails it can be restarted without the need to restart the data nodes. – <http://wiki.apache.org/hadoop/FAQ#What_is_the_purpose_of_the_secondary_name-node.3F> |
| JobTracker | Manages MapReduce jobs and distributes them to the nodes in the cluster. |
| DataNode | Runs on a slave node. Act as HDFS file storage. |
| TaskTracker | Runs MapReduce jobs which are received from the JobTracker. |

**Master and Slaves**

* Master  
  Is the node in the cluster that has the namenode and jobtracker. In this tutorial we will also configure our master node to act as both master and slave.
* Slave  
  Node in the cluster that act as a DataNode and TaskTracker.

*Note: When a node is running a job the TaskTracker will try to use local data (in its “own” DataNode”) if possible. Hence the benefit of having both the DataNode and TaskTracker on the same node since there will be no overhead network traffic. This also implies that it is important to know how data is distributed and stored in HDFS.*

**Start/stop scripts**

|  |  |
| --- | --- |
| **Script** | **Description** |
| start-dfs.sh | Starts NameNode, Secondary NameNode and DataNode(s) |
| stop-dfs.sh | Stops NameNode, Secondary NameNode and DataNode(s) |
| start-mapred.sh | Starts JobTracker and TaskTracker(s) |
| stop-mapred.sh | Stops JobTracker and TaskTracker(s) |

The above scripts should be executed from the NameNode. Through SSH connections daemons will be started on all the nodes in the cluster (all nodes defined in conf/slaves)

**Configuration files**

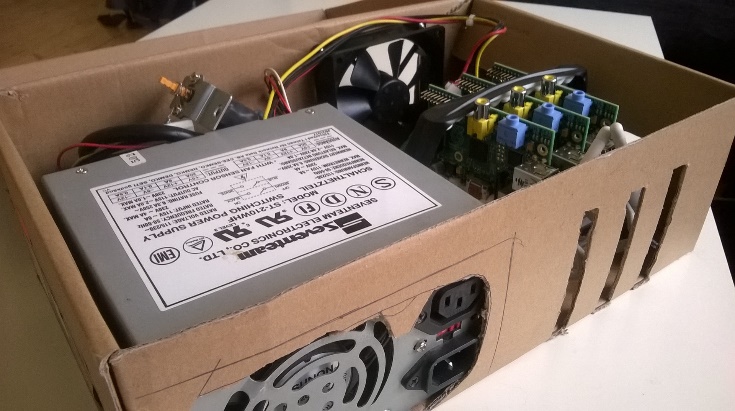
|  |  |
| --- | --- |
| **Configuration file** | **Description** |
| conf/core-site.xml | General site settings such as location of NameNode and JobTracker |
| conf/hdfs-site.xml | Settings for HDFS file system |
| conf/mapred-site.xml | Settings for MapReduce daemons and jobs |
| conf/hadoop-env.sh | Environment configuration settings. Java, SSH and others |
| conf/master | Defines master node |
| conf/slaves | Defines computing nodes in the cluster (slaves). On a slave this file has the default value of localhost |

**Web Interface (default ports)**

Status and information of Hadoop daemons can be viewed from a web browser through web each dameons web interface:

|  |  |
| --- | --- |
| **Daemon/service** | **Port** |
| NameNode | 50070 |
| Secondary NameNode | 50090 |
| JobTracker | 50030 |
| DataNode(s) | 50075 |
| TaskTracker(s) | 50060 |

**The setup**

* Three Raspberry PI’s model B  
  (Or you could do with one if you only do first part of tutorial)
* Three 8GB class 10 SD cards
* **[](http://www.widriksson.com/wp-content/uploads/2014/09/hadoop-cluster-in-a-shoebox.jpg)**An old PC Power Supply
* An old 10/100 router used as network switch
* Shoebox from my latest SPD bicycle shoes
* Raspbian Wheezy 2014-09-09
* Hadoop 1.2.1

|  |  |  |
| --- | --- | --- |
| **Name** | **IP** | **Hadoop Roles** |
| node1 | 192.168.0.110 | NameNode Secondary NameNode JobTracker DataNode TaskTracker |
| node2 | 192.168.0.111 | DataNode TaskTracker |
| node3 | 192.168.0.112 | DataNode TaskTracker |

*Ensure to adjust names and IP numbers to fit your enivronment.*

**Single Node Setup**

**Install Raspbian**

Download Raspbian from:  
<http://downloads.raspberrypi.org/raspbian_latest>

For instructions on how to write the image to an SD card and download SD card flashing program please see:  
<http://www.raspberrypi.org/documentation/installation/installing-images/README.md>

For more detailed instructions on how to setup the Pi see:  
<http://elinux.org/RPi_Hub>

Write 2014-09-09-wheezy-raspbian.img to your SD card. Insert the card to your Pi, connect keyboard, screen and network and power it up.

Go through the setup and ensure the following configuration or adjust it to your choice:

* Expand SD card
* Set password
* Choose console login
* Chose keyboard layout and locales
* Overclocking, High, 900MHz CPU, 250MHz Core, 450MHz SDRAM (If you do any voltmodding ensure you have a good power supply for the PI)
* Under advanced options:
  + Hostname: node1
  + Memory split: 16mb
  + Enable SSH Server

Restart the PI.

**Configure Network**

Install a text editor of your choice and edit as root or with sudo:  
/etc/network/interfaces

iface eth0 inet static

address 192.168.0.110

netmask 255.255.255.0

gateway: 192.168.0.1

Edit /etc/resolv.conf and ensure your namesservers (DNS) are configured properly.

Restart the PI.

**Configure Java Environment**

With the image 2014-09-09-wheezy-raspbian.img Java comes pre-installed. Verify by typing:

java -version

java version "1.8.0"

Java(TM) SE Runtime Environment (build 1.8.0-b132)

Java HotSpot(TM) Client VM (build 25.0-b70, mixed mode)

**Prepare Hadoop User Account and Group**

sudo addgroup hadoop

sudo adduser --ingroup hadoop hduser

sudo adduser hduser sudo

**Configure SSH**

Create SSH RSA pair keys with blank password in order for hadoop nodes to be able to talk with each other without prompting for password.

su hduser

mkdir ~/.ssh

ssh-keygen -t rsa -P ""

cat ~/.ssh/id\_rsa.pub > ~/.ssh/authorized\_keys

Verify that hduser can login to SSH

su hduser

ssh localhost

Go back to previous shell (pi/root).

**Install Hadoop**

**Download and install**

cd ~/

wget http://apache.mirrors.spacedump.net/hadoop/core/hadoop-1.2.1/hadoop-1.2.1.tar.gz

sudo mkdir /opt

sudo tar -xvzf hadoop-1.2.1.tar.gz -C /opt/

cd /opt

sudo mv hadoop-1.2.1 hadoop

sudo chown -R hduser:hadoop hadoop

**Configure Environment Variables**

This configuration assumes that you are using the pre-installed version of Java in 2014-09-09-wheezy-raspbian.img.

Add hadoop to environment variables by adding the following lines to the end of /etc/bash.bashrc:

export JAVA\_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")

export HADOOP\_INSTALL=/opt/hadoop

export PATH=$PATH:$HADOOP\_INSTALL/bin

Alternative you can add the configuration above to ~/.bashrc in the home directory of hduser.

Exit and reopen hduser shell to verify hadoop executable is accessible outside /opt/hadoop/bin folder:

exit

su hduser

hadoop version

hduser@node1 /home/hduser $ hadoop version

Hadoop 1.2.1

Subversion https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503152

Compiled by mattf on Mon Jul 22 15:23:09 PDT 2013

From source with checksum 6923c86528809c4e7e6f493b6b413a9a

This command was run using /opt/hadoop/hadoop-core-1.2.1.jar

**Configure Hadoop environment variables**

As root/sudo edit /opt/hadoop/conf/hadoop-env.sh, uncomment and change the following lines:

# The java implementation to use. Required.

export JAVA\_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")

# The maximum amount of heap to use, in MB. Default is 1000.

export HADOOP\_HEAPSIZE=250

# Command specific options appended to HADOOP\_OPTS when specified

export HADOOP\_DATANODE\_OPTS="-Dcom.sun.management.jmxremote $HADOOP\_DATANODE\_OPTSi **-client**"

*Note 1: If you forget to add the -client option to HADOOP\_DATANODE\_OPTS you will get the following error messge in hadoop-hduser-datanode-node1.out:*

Error occurred during initialization of VM

Server VM is only supported on ARMv7+ VFP

*Note 2: If you run SSH on a different port than 22 then you need to change the following parameter:*

# Extra ssh options. Empty by default.

# export HADOOP\_SSH\_OPTS="-o ConnectTimeout=1 -o SendEnv=HADOOP\_CONF\_DIR"

export HADOOP\_SSH\_OPTS="-p <YOUR\_PORT>"

Or you will get the error:

connect to host localhost port 22: Address family not supported by protocol

## Configure Hadoop

In /opt/hadoop/conf edit the following configuration files:

### ****core-site.xml****

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/hdfs/tmp</value>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

</property>

</configuration>

### ****mapred-site.xml****

<configuration>

  <property>

    <name>mapred.job.tracker</name>

    <value>localhost:54311</value>

  </property>

</configuration>

### ****hdfs-site.xml****

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

## Create HDFS file system

sudo mkdir -p /hdfs/tmp

sudo chown hduser:hadoop /hdfs/tmp

sudo chmod 750 /hdfs/tmp

hadoop namenode -format

## Start services

Login as hduser. Run:

/opt/hadoop/bin/start-dfs.sh

/opt/hadoop/bin/start-mapred.sh

Run the jps command to checkl that all services started as supposed to:

jps

16640 JobTracker

16832 Jps

16307 NameNode

16550 SecondaryNameNode

16761 TaskTracker

16426 DataNode

If you cannot see all of the processes above review the log files in /opt/hadoop/logs to find the source of the problem.

## Run sample test

Upload sample files to HDFS (Feel free to grab any other textfile you like than license.txt):

hadoop dfs -copyFromLocal /opt/hadoop/LICENSE.txt /license.txt

Run wordcount example:

hadoop jar /opt/hadoop/hadoop-examples-1.2.1.jar wordcount /license.txt /license-out.txt

When completed you will see some statistics about the job. If you like to see the outputfile grab the file form HDFS to local file system:

hadoop dfs -copyToLocal /license-out.txt ~/

Open the ~/license-out.txt/part-r-00000 file in any text editor to see the result. (You should have all words in the license.txt file and their number of occurrences)