

Assignment 4

Hadoop Map Reduce Cluster Setup

Atharva Vaidya
121942024

Contents

Introduction	2
What is Mapreduce and How it Works?	2
What is the problem that MapReduce is trying to solve?	3
Installing Hadoop	4
Configuring Linux Environment for Hadoop Installation	4
Java Installation	4
Install SSH	4
Downloading Hadoop	5
Configuration of Hadoop	5
hadoop-env.sh	7
core-site.xml	8
hdfs-site.xml	9
mapred-site.xml	10
yarn-site.xml	11
Format Namenode	12
Start Master Node and Slave Node	12
Start YARN	12

Introduction

What is Mapreduce and How it Works?

MapReduce is the processing engine of the Apache Hadoop that was directly derived from the Google MapReduce. The MapReduce application is written basically in Java. It conveniently computes huge amounts of data by the applications of mapping and reducing steps in order to come up with the solution for the required problem. The mapping step takes a set of data in order to convert it into another set of data by breaking the individual elements into key/value pairs called tuples. The second step of reducing takes the output derived from the mapping process and combines the data tuples into a smaller set of tuples.

MapReduce is a hugely parallel processing framework that can be easily scaled over massive amounts of commodity hardware to meet the increased need for processing larger amounts of data. Once you get the mapping and reducing tasks right all it needs a change in the configuration in order to make it work on a larger set of data. This kind of extreme scalability from a single node to hundreds and even thousands of nodes is what makes MapReduce a top favorite among Big Data professionals worldwide.

- Enables parallel processing required to perform Big Data jobs
- Applicable to a wide variety of business data processing applications
- A cost-effective solution for centralized processing frameworks
- Can be integrated with SQL to facilitate parallel processing capability

The entire MapReduce process is a massively parallel processing setup where the computation is moved to the place of the data instead of moving the data to the place of the computation. This kind of approach helps to speed the process, reduce network congestion and improves the efficiency of the overall process.

The entire computation process is broken down into the mapping, shuffling and reducing stages.

Mapping Stage This is the first step of the MapReduce and it includes the process of reading the information from the Hadoop Distributed File System (HDFS). The data could be in the form of a directory or a file. The input data file is fed into the mapper function one line at a time. The mapper then processes the data and reduces it into smaller blocks of data.

Reducing Stage The reducer phase can consist of multiple processes. In the shuffling process, the data is transferred from the mapper to the reducer. Without the successful shuffling of the data, there would be no input to the reducer phase. But the shuffling process can start even before the mapping process has completed. Next, the data is sorting in order to lower the time taken to reduce the data. The sorting actually helps the reducing process by providing a cue when the next key in the sorted input data is distinct from the previous key. The reduce task needs a specific key-value pair in order to

call the reduce function that takes the key-value as its input. The output from the reducer can be directly deployed to be stored in the HDFS. Some of the terminologies in the MapReduce process are:

- **MasterNode** – Place where JobTracker runs and which accepts job requests from clients
- **SlaveNode** – It is the place where the mapping and reducing programs are run
- **JobTracker** – it is the entity that schedules the jobs and tracks the jobs assigned using Task Tracker
- **TaskTracker** – It is the entity that actually tracks the tasks and provides the report status to the JobTracker
- **Job** – A MapReduce job is the execution of the Mapper and Reducer program across a dataset
- **Task** – the execution of the Mapper and Reducer program on a specific data section
- **TaskAttempt** – A particular task execution attempt on a SlaveNode

What is the problem that MapReduce is trying to solve?

MapReduce directly came from the Google MapReduce which was a technology for parsing large amounts of web pages in order to deliver the results that has the keyword which the user has searched in the Google search box.

Installing Hadoop

Configuring Linux Environment for Hadoop Installation

Hadoop can be installed in 3 different modes:

- Standalone mode
- Pseudo-Distributed mode
- Fully-Distributed mode

Standalone mode is the default mode of operation of Hadoop and it runs on a single node (a node is your machine). HDFS and YARN doesn't run on standalone mode.

Pseudo-Distributed mode stands between the standalone mode and fully distributed mode on a production level cluster. It is used to simulate the actual cluster. It simulated 2 node — a master and a slave by running JVM process. it gives you a fully-fledged test environment. HDFS is used for storage using some portion of your disk space and YARN needs to run to manage resources on this Hadoop installation.

Full Distributed runs on cluster of machines. Lots of configuration parameter had to be setup for production system.

Java Installation

Hadoop is built using Java and Java is required to run MapReduce code. You can check whether you have Java installed on your machine. The Java version should be higher than Java 7 i.e Java 1.7. for the latest hadoop installs.

Install SSH

Master node communicate with slave node very frequently over SSH protocol. in Pseudo-Distributed mode, only one node exist (your machine) and master slave interaction is simulated by JVM. Since communication is very frequent, ssh should be password less. Authentication needs to be done using Public key. Above command may not work in your machine if ssh is not installed on your machine, use this command to install ssh

```
sudo apt-get install openssh-server
```

Downloading Hadoop

Go to the directory you have downloaded the compressed Hadoop file and unzip using terminal

```
tar -xzvf hadoop-2.7.3.tar.gz
sudo mv hadoop-2.7.3 /usr/local/hadoop
```

Now go to the Hadoop distribution directory using terminal

```
cd /usr/local/hadoop
```

Inside the Hadoop folder

- etc — has the configuration files for Hadoop environment.
- bin — include various commands useful like Hadoop cmdlet.
- share — has the jars that is required when you write MapReduce job. It has Hadoop libraries

Hadoop command in the bin folder is used to run jobs in Hadoop.

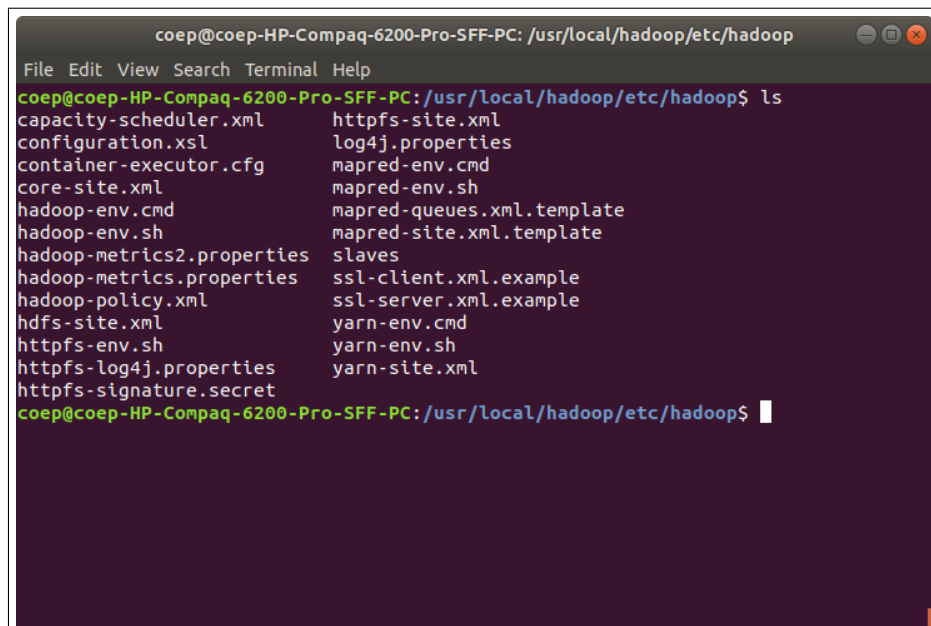
```
bin/hadoop
```

jar command is used to run the MapReduce jobs on Hadoop cluster

```
bin/hadoop jar
```

Configuration of Hadoop

Most of the files are in etc/hadoop.

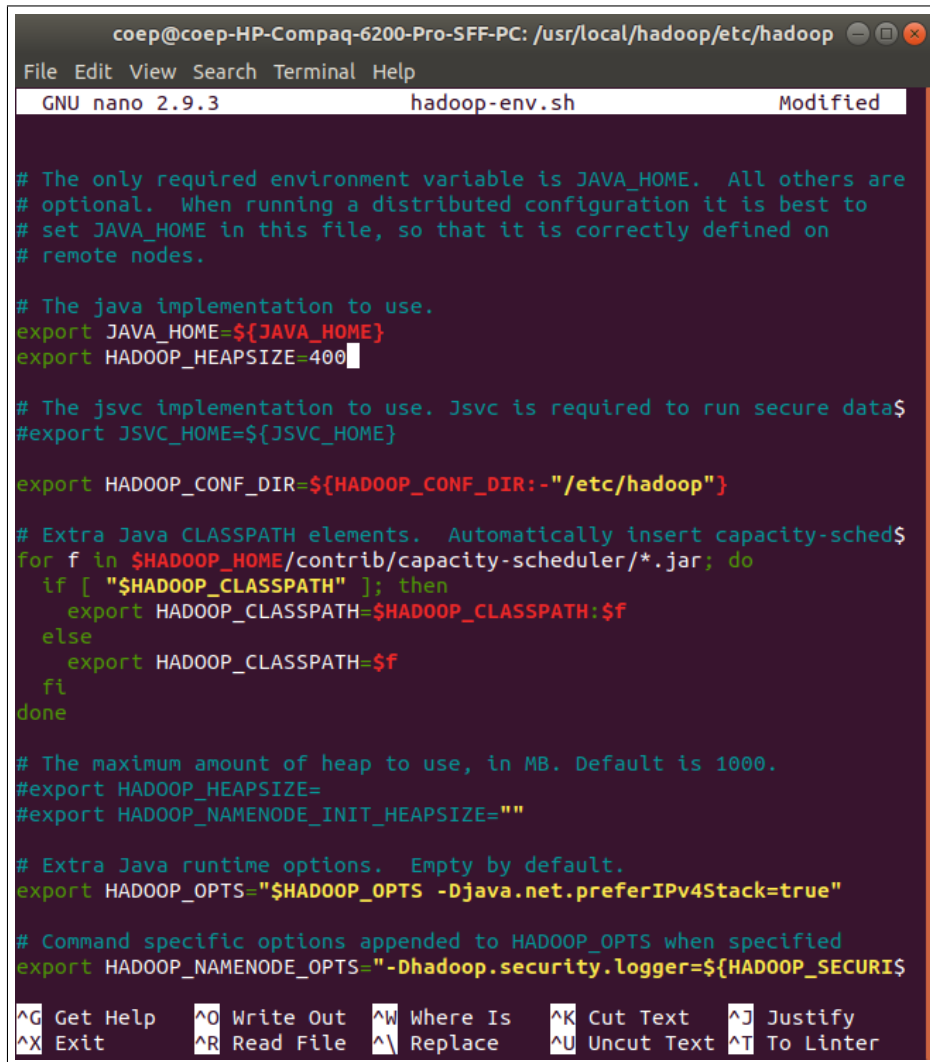
A terminal window titled 'coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop' displays the output of the 'ls' command. The files are listed in two columns. The first column contains: capacity-scheduler.xml, configuration.xml, container-executor.cfg, core-site.xml, hadoop-env.cmd, hadoop-env.sh, hadoop-metrics2.properties, hadoop-metrics.properties, hadoop-policy.xml, hdfs-site.xml, httpfs-env.sh, httpfs-log4j.properties, and httpfs-signature.secret. The second column contains: httpfs-site.xml, log4j.properties, mapred-env.cmd, mapred-env.sh, mapred-queues.xml.template, mapred-site.xml.template, slaves, ssl-client.xml.example, ssl-server.xml.example, yarn-env.cmd, yarn-env.sh, and yarn-site.xml. The prompt 'coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop\$' is visible at the bottom.

```
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop$ ls
capacity-scheduler.xml  httpfs-site.xml
configuration.xml       log4j.properties
container-executor.cfg  mapred-env.cmd
core-site.xml           mapred-env.sh
hadoop-env.cmd          mapred-queues.xml.template
hadoop-env.sh           mapred-site.xml.template
hadoop-metrics2.properties  slaves
hadoop-metrics.properties  ssl-client.xml.example
hadoop-policy.xml         ssl-server.xml.example
hdfs-site.xml            yarn-env.cmd
httpfs-env.sh            yarn-env.sh
httpfs-log4j.properties  yarn-site.xml
httpfs-signature.secret
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop$
```

Figure 1:

hadoop-env.sh

This file contains some environment variable settings used by Hadoop. It can be configured to affect some aspects of Hadoop daemon behavior, such as where log files are stored, the maximum amount of heap used etc.



```
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 2.9.3 hadoop-env.sh Modified

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}
export HADOOP_HEAPSIZE=400

# The jsvc implementation to use. Jsvc is required to run secure data$
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements. Automatically insert capacity-sched$
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
    if [ "$HADOOP_CLASSPATH" ]; then
        export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
    else
        export HADOOP_CLASSPATH=$f
    fi
done

# The maximum amount of heap to use, in MB. Default is 1000.
#export HADOOP_HEAPSIZE=
#export HADOOP_NAMENODE_INIT_HEAPSIZE=""

# Extra Java runtime options. Empty by default.
export HADOOP_OPTS="$HADOOP_OPTS -Djava.net.preferIPv4Stack=true"

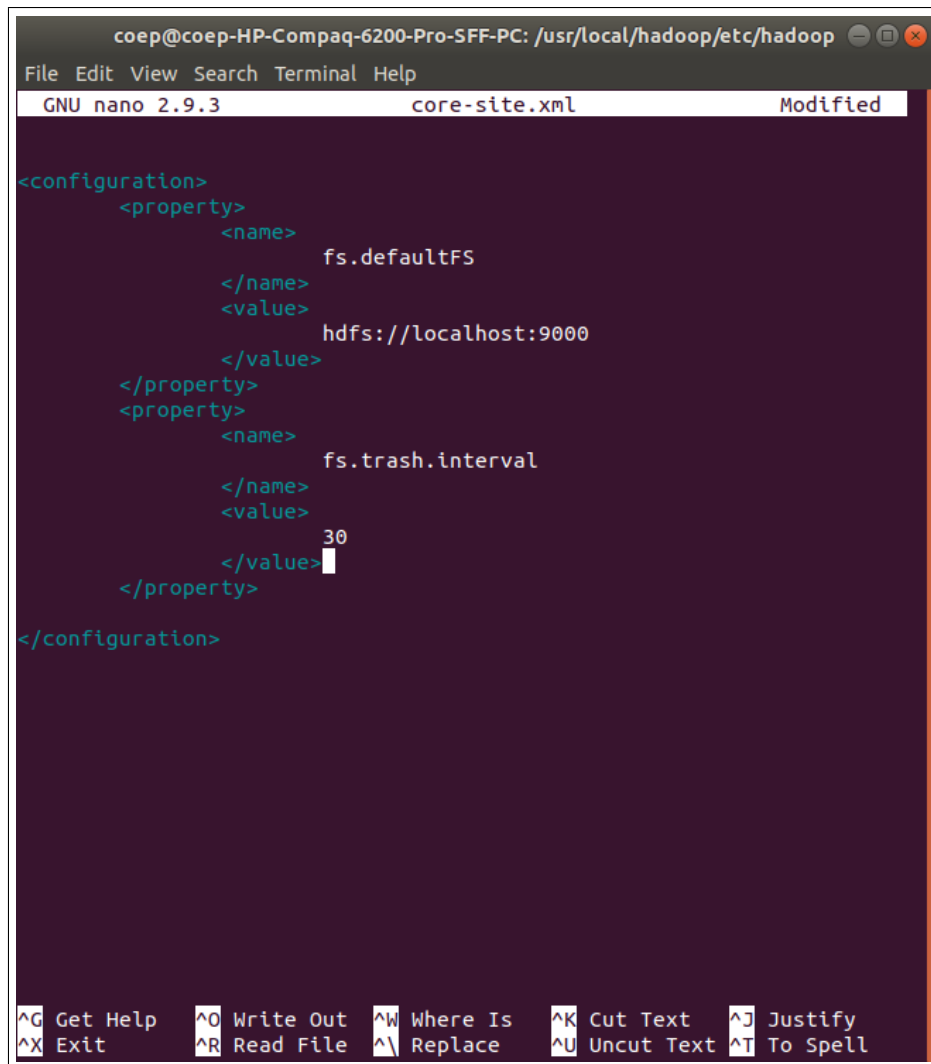
# Command specific options appended to HADOOP_OPTS when specified
export HADOOP_NAMENODE_OPTS="-Dhadoop.security.logger=${HADOOP_SECURITY_LOGGER}"

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter
```

Figure 2: Contents of hadoop-env.sh

core-site.xml

The core-site.xml file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.



```
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 2.9.3 core-site.xml Modified

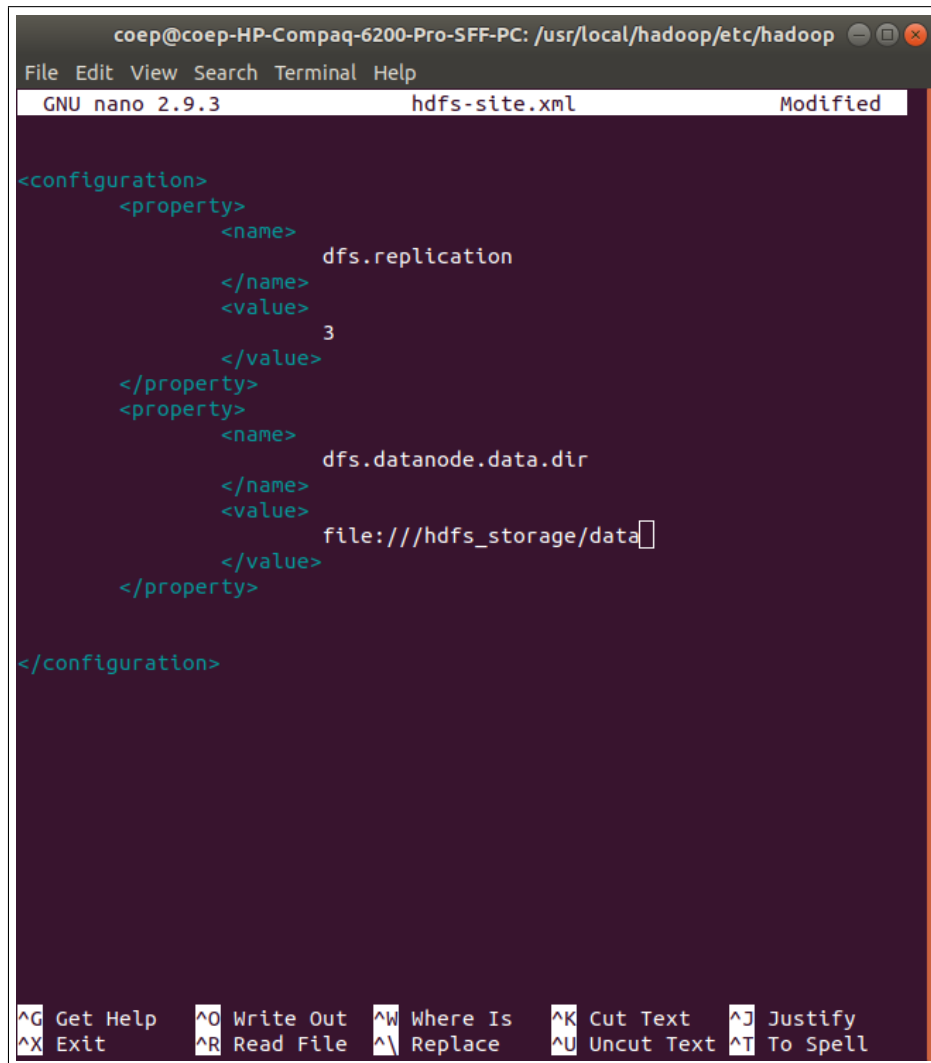
<configuration>
  <property>
    <name>
      fs.defaultFS
    </name>
    <value>
      hdfs://localhost:9000
    </value>
  </property>
  <property>
    <name>
      fs.trash.interval
    </name>
    <value>
      30
    </value>
  </property>
</configuration>

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^_ Replace    ^U Uncut Text ^T To Spell
```

Figure 3: Contents of core-site.xml

hdfs-site.xml

The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes.



```
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 2.9.3 hdfs-site.xml Modified

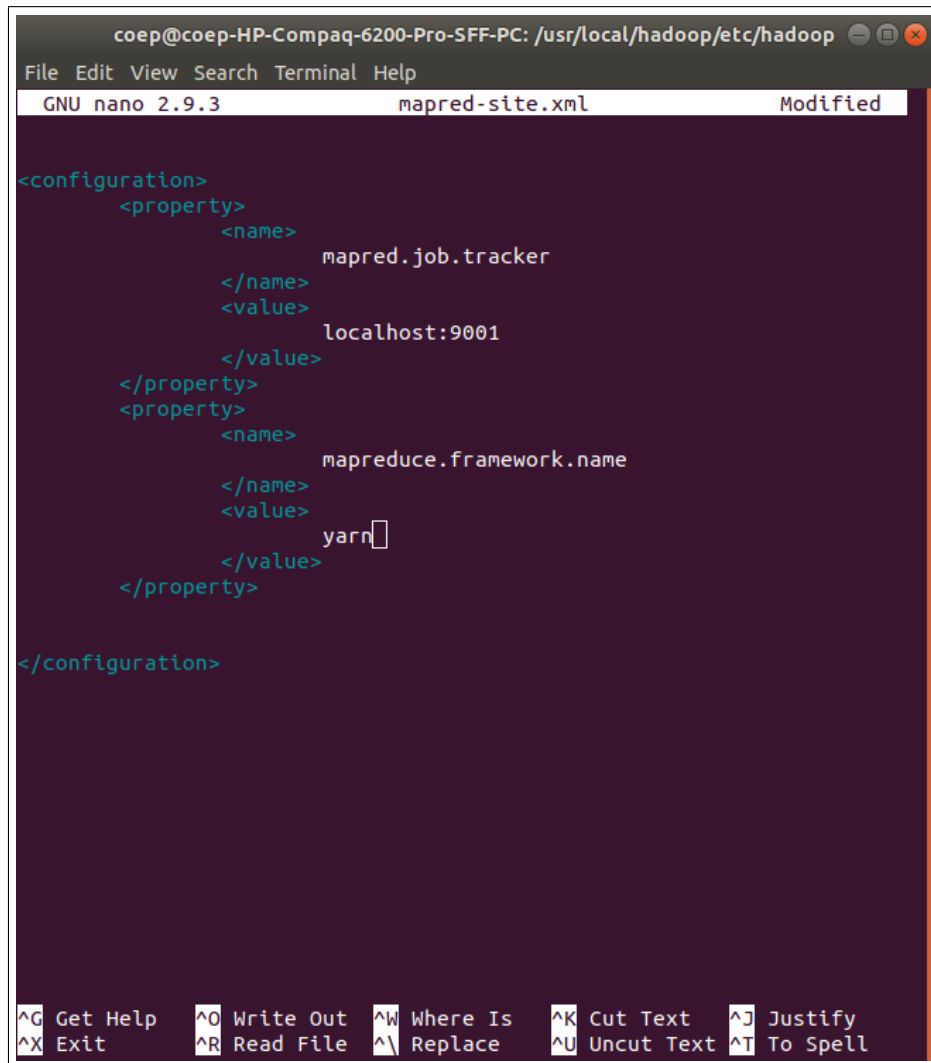
<configuration>
  <property>
    <name>
      dfs.replication
    </name>
    <value>
      3
    </value>
  </property>
  <property>
    <name>
      dfs.datanode.data.dir
    </name>
    <value>
      file:///hdfs_storage/data
    </value>
  </property>
</configuration>

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell
```

Figure 4: Contents of hdfs-site.xml

mapred-site.xml

The mapred-site.xml file contains the configuration settings for MapReduce daemons; the job tracker and the task-trackers.



The image shows a terminal window with the title bar "coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop". The window is running the GNU nano 2.9.3 text editor, editing the file mapred-site.xml. The file content is as follows:

```
<configuration>
  <property>
    <name>
      mapred.job.tracker
    </name>
    <value>
      localhost:9001
    </value>
  </property>
  <property>
    <name>
      mapreduce.framework.name
    </name>
    <value>
      yarn
    </value>
  </property>
</configuration>
```

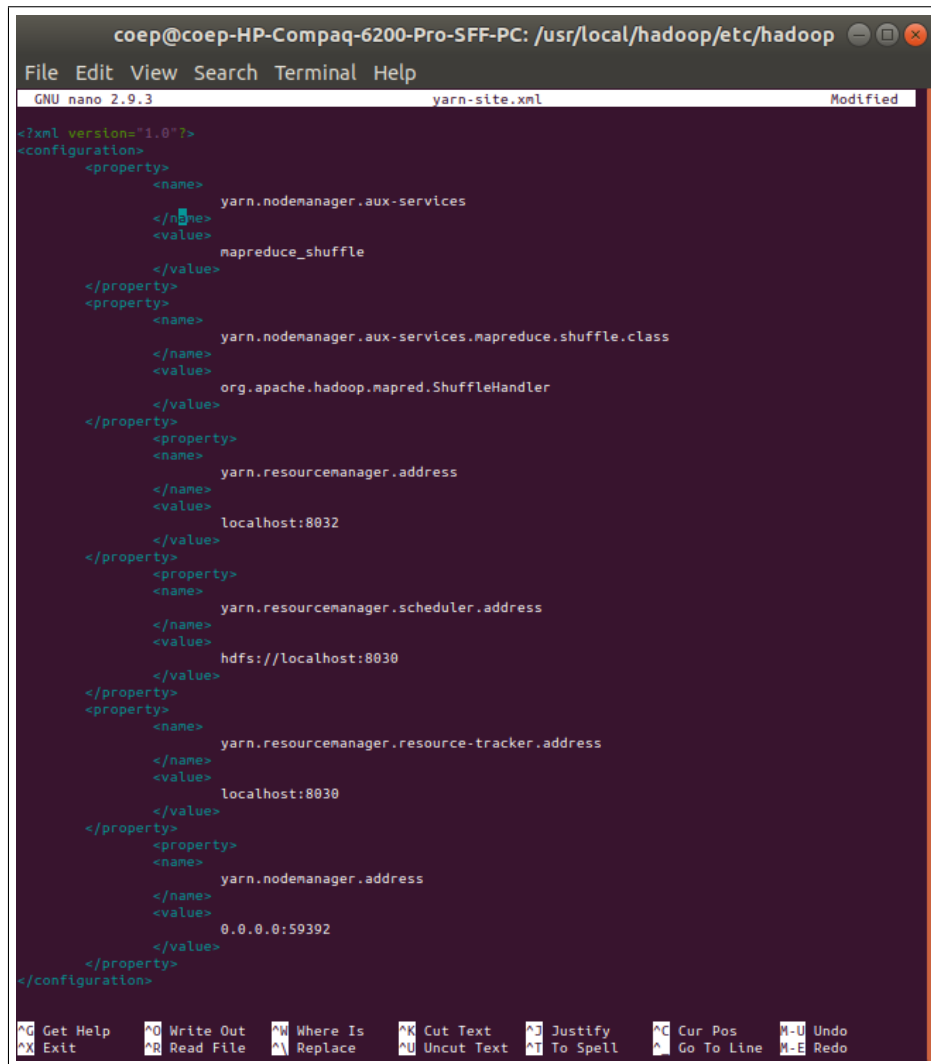
The bottom of the terminal shows the nano editor's command shortcuts:

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify
^X Exit	^R Read File	^_ Replace	^U Uncut Text	^T To Spell

Figure 5: Contents of mapred-site.xml

yarn-site.xml

YARN is a resource negotiator.



```
coep@coep-HP-Compaq-6200-Pro-SFF-PC: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 2.9.3 yarn-site.xml Modified
<?xml version="1.0"?>
<configuration>
  <property>
    <name>
      yarn.nodemanager.aux-services
    </name>
    <value>
      mapreduce_shuffle
    </value>
  </property>
  <property>
    <name>
      yarn.nodemanager.aux-services.mapreduce.shuffle.class
    </name>
    <value>
      org.apache.hadoop.mapred.ShuffleHandler
    </value>
  </property>
  <property>
    <name>
      yarn.resourcemanager.address
    </name>
    <value>
      localhost:8032
    </value>
  </property>
  <property>
    <name>
      yarn.resourcemanager.scheduler.address
    </name>
    <value>
      hdfs://localhost:8030
    </value>
  </property>
  <property>
    <name>
      yarn.resourcemanager.resource-tracker.address
    </name>
    <value>
      localhost:8030
    </value>
  </property>
  <property>
    <name>
      yarn.nodemanager.address
    </name>
    <value>
      0.0.0.0:59392
    </value>
  </property>
</configuration>
```

Figure 6: Contents of yarn-site.xml

Format Namenode

Move to bin directory and format the existing namenode using the command. NameNode is the master node in any cluster and keeps track of all the other nodes in the cluster where the processes run.

```
hdfs namenode -format
```

Start Master Node and Slave Node

Once the above step is successful, move to usr/local/hadoop/sbin directory and fire following commands

```
./start-dfs.sh
```

It can be checked if the NameNode is running on the browser under local-host:50070

jps command gives the list of running process in java. This is used to check if HDFS is running.

Start YARN

YARN is a resource negotiator. We can start it by running the command on the same sbin directory.

```
start-yarn.sh
```