SCHOOL OF SCIENCE & ENGINEERING

Installation and configuration system/tool for Hadoop

Capstone Design

April 2015
Abdelaziz El Ammari
Supervised by: Dr. Nasser Assem
Supervisor's signature:

Dr Nasser Assem
ACKNOWLEDGMENT

The idea of this capstone project would not have been realized without the continuous help and encouragement of many people:

I would like first to thank my supervisor Dr. Nasser Assem for his valuable recommendations and pieces of advice and great supervising methodology.

In addition, I would like to thank my parents, sister, aunt and grandmother for their psychological support during the hard times of this semester.

Many thanks go to my very close friends: Salim EL Kouhen, Ahmed El Agri, soukayna Jermouni, Lamyaa Hanani.

I would like also to thank Ms. Ouidad Achahbar and Ms. Khadija Akherfi for their assistance and support.

Special acknowledgements go to all my professors for their support, respect and encouragement. Thank you Mr. Omar Iraqui, Ms. Asmaa Mourhir, Dr. Naeem Nizar Sheikh, Dr. Redouane Abid, Dr. Violetta Cavalli Sforza and Dr. Kevin Smith.
Contents

1. Introduction.................................................................................................................. 1
   1.1 Project Purpose ....................................................................................................... 2
   1.2 Motivation .............................................................................................................. 2
   1.2 STEEPLE Analysis ............................................................................................... 2
2. Theoretical Baselines ................................................................................................. 3
   2.1 Big Data ................................................................................................................ 3
   2.1.1 Big data analysis approach .............................................................................. 4
   2.2 Distributed file system ........................................................................................ 4
   2.3 Hadoop .................................................................................................................. 5
   2.4 Hadoop Distributed File System .......................................................................... 6
   2.4.1 Name node ...................................................................................................... 6
   2.4.2 Data node ....................................................................................................... 6
   2.4.3 Data Replication ............................................................................................. 6
   2.5 MapReduce .......................................................................................................... 7
   2.5.1 Example of MapReduce(word count) ............................................................... 8
3. Methodology .............................................................................................................. 9
   3.1 Process Model ....................................................................................................... 9
4. Functional Requirements ........................................................................................... 10
5. Non-Functional requirements .................................................................................. 10
   5.2 Sequence diagrams ........................................................................................... 11
   5.3 Entity relational diagram .................................................................................... 14
   5.3.1 Table description ............................................................................................ 14
6. Installation and configuration .................................................................................. 15
7. Technology enablers and implementation results .................................................... 15
   7.1 Technology enablers ........................................................................................... 15
   7.2 Implementation result .......................................................................................... 16
8. Future work .............................................................................................................. 16
9. Conclusion ................................................................................................................. 17
10. Appendix ................................................................................................................... 17
   10.1 Prerequisites ..................................................................................................... 17
   10.2 Hadoop .............................................................................................................. 20
11. References ............................................................................................................... 27
List of figures

- Figure 2.1.1: Big Data 3Vs
- Figure 2.3.1: High level Architecture of Hadoop
- Figure 2.4.1: HDFS Architecture
- Figure 2.5.1: Map Reduce Architecture
- Figure 2.5.2: Mapreduce word count process
- Figure 3.1.1: waterfall model
- Figure 1: JDK being installed
- Figure 2: SSH server being installed
- Figure 3: SSH client being installed
- Figure 4: SSH keygen being generated
- Figure 5: Downloading Hadoop
- Figure 6: Hadoop being extracted
- Figure 7: Creation of a temporary file
- Figure 8: Configuration of Hadoop files
- Figure 10: Running Hadoop
- Figure 11: Checking JPS
- Figure 4.2.1: Sequence diagram for installing the JDK
- Figure 4.2.2: Sequence diagram for installing the SSH
- Figure 4.2.3: Sequence diagram for installing the Hadoop
- Figure 4.2.4: Sequence diagram for running the Hadoop
- Figure 4.4.1: Entity relational diagram
ABSTRACT

The idea behind my capstone project is to develop a Linux based application to help Hadoop users to install and configure the framework without having to deal with the overhead of modifying many files and calling a lot of command lines in the terminal. This application concerns the configuration of Hadoop for a single node cluster. It is based on Python programming language for the coding. Concerning the interface part, I used the software “Quickly”, which is dedicated to create Linux applications. To be able to develop this application, I needed to analyze the different ways to use system calls directly from my python application. Finally, I adapted the steps of configuration of Hadoop to work for every 64bit machine using my previous experience during the research I have performed for single node cluster. This Application will help Big Data users and developers especially Hadoop users to get it configured in less time than normally needed by other users. In fact, a novice user of Hadoop would need several days to get it installed and configured. This application would help them perform the task in 5 to 10 minutes for a single node cluster.
1. Introduction

Big Data is definitely one of the most important terms nowadays, and will still be in the future within the development of technology. As this term means a large collection of data sets, one would immediately think about the different possible ways to manage and process this Big Data. After making some research about it, the results have shown some issues related to the handling of such large and complex data sets. Indeed, many data processing software have been proven to have some limitations and to be unable to handle Big Data including standard database management systems, relational database management systems and object relational database management systems. [8]

Big Data is currently a subject to many studies because of its huge significance. Indeed, very important areas are collecting huge amounts of data such as:

- Astronomical research centers.
- DNA research centers.
- NASA Center for Climate Simulation.
- Governments, from which the United States Federal Government is occupying the first position with its six out of ten most powerful worldwide supercomputers.
- Walmart, the American multinational retail corporation.
- Facebook, the most popular social network.
- Ebay.com, the multinational e-commerce company.
- Amazon.com, the international electronic commerce company.

The above companies and research centers, in addition to many others, play a significant role in our daily lives, which makes Big Data necessary to be able to find a way for the data produced to be structured and stored safely and effectively. Therefore, this capstone will deliver an installation and configuration tool for Hadoop. Hadoop is a very helpful technology enabler to parallel processing platforms that uses the MapReduce algorithm.

The idea behind my capstone project is to develop a Linux based application to help Hadoop users install and configure the framework without having to deal with the overhead of modifying many files and calling a lot of command lines in the terminal.
1.1 Project Purpose

This capstone project called “Installation configuration system/tool for Hadoop” is a partial fulfillment of the Bachelor of Science in Computer Science’ requirements in Al Akhawayn University. The main purpose behind this project is to test and ensure that I am a ready engineer who can design and develop sophisticated and complex systems based on all the competences I acquired throughout my journey in the university. This project is the result of my work during this whole semester under the supervision of Dr Nasser Assem.

1.2 Motivation

My capstone project is an extension of the research project that I have performed last semester, which consists of analyzing the performance of HDFS within a cluster using a benchmark. This benchmark involves measuring the time of the execution of creating a large sample of random data. Also, the calculation of the time needed to delete it using a cluster. Before going through all this process, I had to install Hadoop in my machines. The required time for installing this framework is several days due to many errors I faced and needed to fix. The other issue was that there is a shortage of resources and documentation to be used while trying to install my application.

1.2 STEEPLE Analysis

- **Societal considerations**

Concerning the societal implications of my application, It will help Big data users and developers especially Hadoop users to get it configured in less time than normally needed by other users. In fact, a novice user of Hadoop would need several to get it installed and configured. This application would help them perform the task in 2 to 5 minutes.

- **Technical considerations**

Concerning this aspect, the application is respecting it as it does not involve some complex technologies that the user cannot deal with. However, the user needs not to have a prior technical knowledge to Linux technologies in order to interact with the system.

- **Environmental considerations**
My application does not have a direct relation with this kind of consideration. Indirectly, Big Data is used in many environmental sectors such as smart grids.

- **Ethical consideration**
  The subject of my application is very ethical, and the research materials that I relied on are all cited to respect engineering ethics. Moreover, I will make sure to contact the Hadoop center to allow me to share this application with other people in a legal manner.

- **Political & legal consideration**
  Actually, my application does not have a relationship with any political or legal sides of our society.

- **Economic**
  My application is free for every user and will be shared on Ubuntu software center.

2. **Theoretical Baselines**

2.1 **Big Data**

Big data is described as voluminous amount of structured, semi-structured and unstructured data that has the potential to be mined from different sources including social media, administrative services and research organizations [9].

These organizations may face hundreds of gigabytes or terabytes data and need to consider the processing time and data management options. Big Data are characterized by the 3Vs:

- **Volume**: The gigantic volume of data can reach up to hundreds of Petabytes.
- **Variety**: The wide type of data processed
- **Velocity**: The velocity by which data is processed varies
2.1.1 Big data analysis approach

Big data necessitates an extensive time to be loaded and processed through normal Relational Database Management Systems (DBMS) in order to be analyzed. New approaches and schemes having been developed to analyze the Big Data relying less on the quality.[8]

Big data analytics is allied to cloud computing environments because the analysis of a large set of data in real time entails a platform similar to Hadoop in order to process data across distributed clusters and MapReduce.[8]

2.2 Distributed file system

Google file system was designed in 2003 by Google foundation. It is based on a distributed file system that is fault tolerant since data is partitioned and replicated. The Core layer of using the cloud computing platform is to read an output and to store an input using the Map reduce.[8] Hadoop was generated in 2005 by Apache foundation as open source framework that uses MapReduce system from GFS. The first enterprise to deploy Hadoop was Yahoo in its file system. Hadoop file system and GFS do not implement POSIX, but they are optimized for large files up to Exabyte of data. Besides being fault tolerant, Hadoop is able to handle the growth of amount data needed to be processed.[8]
2.3 Hadoop

Margaret Rouse defines Hadoop as a free Java-based programming framework that supports the processing of large data sets in a distributed computing environment [9]. Apache Software Foundation is the main sponsor for this project.

Using Hadoop framework, it is possible to run applications connected by thousands of nodes and demanding Petabytes of data. Hadoop distributed file system enables the data transfer within the nodes allowing the system to continue the process in uninterruptable manner.[8]

Hadoop is using MapReduce as a software framework, a product from Google, that consists of breaking down an application into small chunks to be run in a node in the cluster.[8]

![High Level Architecture of Hadoop](image)

*Figure 2.3.1: High level Architecture of Hadoop*
2.4 Hadoop Distributed File System

HDFS is a distributed file system that offers a high performance access to data across clusters. It has become a key tool for managing Big Data analytics programs. The file system is developed in such a way to be fault-tolerant. Nevertheless, HDFS facilitates the data transfer and enables the system to run even if a node fails, which decreases the risk of failure.[8]

HDFS breaks down data and distributes it through the nodes among the cluster allowing for parallel processing. Moreover, the data is copied several times and each copy is placed in at least two different server racks. Hence, if the system fails to find data in node, it could be retrieved from a different rack to continue the data processing while the system is recovering from the failure. HDFS is built in a platform that consists of Master and Slave architecture within a cluster. [8]

2.4.1 Name node

An HDFS cluster comprises a single master server that manages the filesystem namespace. In addition, it regulates access to files by clients. This server is called the namenode. HDFS consists of a file system namespace that permits user data to be stored in files. Internally, a file is split into different blocks stored in a set of Datanodes. The Namenode makes filesystem namespace operations like opening, closing, renaming etc. of files and directories. It also determines the mapping of blocks to Datanodes.[10]

2.4.2 Data node

In each node of the cluster, there are a number of Datanodes. These systems manage storage attached to the nodes that they run on. The reading and writing requests from filesystem clients are performed by the Datanodes. Also, they perform block creation, deletion, and replication upon instruction from the Namenode.[10]

2.4.3 Data Replication

The design of HDFS is recognized as a reliable system to store very large files across a large cluster [10]. The files are stored in a form if sequence of blocks of the same size except the last block. The purpose behind replicating the files is fault tolerance. The block size and replication factor are configurable per file. Files in HDFS are write-once and have strictly one writer at any time [10]. The Namenode makes all decisions concerning replication of blocks.
Moreover, the Namenode receives an intervallic Heartbeat and a Blockreport from each Datanode within the cluster. A receipt of a heartbeat informs about the good health of the Datanode. A Blockreport contains a list of all blocks on that Datanode.[10]

As shown in the figure 2.4.1, The NameNode manages the file system operations whereas the DataNode is responsible for the management of data storage on each node.

2.5 MapReduce

MapReduce is a programming pattern that was designed by Google Foundation in 2004. The idea behind MapReduce is to split data into chunks as that will be processed in parallel. The output of the mapping process is direct as an input to the Reduce part so that can be gathered at the end.[8]

The Map task consists of five phases including reading, mapping, collecting, spilling and merging. The reading phase entails reading the data from the HDFS and creating the Key-value. The Mapping phase involves executing the map function in order to generate the map-output data. Concerning the collection of the data, the map-out data is put in a buffer before spilling. In the spilling phase, data is compressed and written in local disk. At the end of the process, the merging phase consists of integrating all file into an output file in each node.[8]
The Reduce task comprises four other phases: shuffling, merging, reducing, and writing. During the shuffling phase, the map-output data is transferred to the reduced node to get decompressed. The merging phase executes the assimilation of the outputs coming from different mappers to the reduce phase. The reducing phase calls a reduce function to return the final output. The last step consists of writing back the output data in the HDFS. [8]

MapReduce allows developers to use library routines to create programs that need a parallel processing framework without being perturbed by the intra-cluster communication and failure handling since it is also Fault-tolerant. [8]

---

**Figure 2.5.1: Map Reduce Architecture**

### 2.5.1 Example of MapReduce (word count)

As demonstrated in figure 2.5.2, the process of counting the occurrence of words in a text file. The framework starts by reading the input. The next step is to split the lines through different nodes. In the mapping phase, the key part representing the word is mapped to a value showing the number of times the word is repeated within the line. These actions are repeated over all stations. Then, the word is shuffled by putting the mapping of the same word in the
corresponding rack. Finally, this steps consists of summing up the values of the words. As a final step, data is set as an output file ready to be generated.

![Figure 2.5.2: MapReduce word count process](image)

3. Methodology

3.1 Process Model

Deciding about which development model to follow in this project was not difficult to make, as I knew exactly the output of my project at the end of the semester, given the constraints of time, of implementation difficulties and other constraints. I decided to use the Waterfall model for different reasons. I already have my list of requirements finalized and I am sure that I do not have to reiterate on the requirements. Finally, Waterfall model is suitable to develop my application because it allows me to have each step of the development clear and separated from others.
4. Functional Requirements

The system “Installation configuration system/tool for Hadoop” will deal with two main parts, the application that Installation of major components of Hadoop. In the second part, the application will configure the required files and check if all processes are successfully running.

5. Non-Functional requirements

The program should be:

- Containing information about each function that is going to be executed:
  - In each of the windows, I describe the actions performed at each level.

- Having a help window to assist the users:
  - This window will guide the user to better use the application, as well as the prerequisites needed to start installing the program including a 64bit machine.

- User friendly:
- The application consists of dialogs, and the user needs to click on the button next until he finishes installing Hadoop.

- Available for every user in the Ubuntu software center:
  - After finalizing my project, I will post it on the Ubuntu software center to make it available for everyone.

- Keeping track of the operation done before aborting the program:
  - When the user restart the application, the system will ask him if he wants to continue from the point where he stopped.

5.2 Sequence diagrams

As shown in figure 4.2.1, the sequence diagram demonstrates how the system interacts the online server to download the JDK version 7. It starts first by sending a request to the server. After getting a reply from the server, the application starts download the JDK.

![Sequence diagram for installing the JDK](image)

*Figure 4.2.1: Sequence diagram for installing the JDK*
The figure 4.2.2 shows how the process of installing SSH as part of prerequisites to install Hadoop. The application starts first by installing the openssh server and client components followed by generating a public key that is needed to authenticate machines.

**Figure 4.2.2: Sequence diagram for installing the SSH**

The figure 4.2.3 expresses the steps of downloading Hadoop from the server. Then, the system extracts the file inside the Hadoop-tool folder. The last step is to give ownership to the user to make changes to it including creating a temporary file inside it.
Figure 4.2.3: Sequence diagram for installing the Hadoop

As shown in figure 4.2.4, the last step before running Hadoop consists of formatting the Namenode. After running the cluster, the user needs to call the jps to check if all processes are successfully running.
Figure 4.2.4: Sequence diagram for running the Hadoop

5.3 Entity relational diagram

<table>
<thead>
<tr>
<th></th>
<th>History</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-his_id: int</td>
<td>-record_id: int</td>
</tr>
<tr>
<td></td>
<td>-time: DateTime</td>
<td>-his_id: int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-operation: varchar(20)</td>
</tr>
<tr>
<td></td>
<td>0..1</td>
<td>0..n</td>
</tr>
</tbody>
</table>

Figure 4.4.1: Entity relational diagram

5.3.1 Table description:
As shown in figure 4.4.1, the ERD is a combination of all the entities that I will be working on to satisfy all the functional and non-functional requirements of this capstone project “Installation configuration system/tool for Hadoop”. Here is a brief description of all the entities presented in the ERD.
• Table History:

This table allows the user to track the date and time when he started the operation of installing Hadoop. Using this table, the user continues installing the program from the point where he stopped before aborting it.

• Table Record:

This Table allows the user to keep track of the operations that he has made before abort the installation.

6. Installation and configuration

Before starting to work in my project, I had to find an environment that will allow me to create Linux application in an easy manner. After many investigations and extensive research, my choice was to work using “Quickly”. It is a Python based development environment that allows the creation, packaging and sharing of Linux applications on Ubuntu Software center. I used the MySQL server to manage my database from the creation of tables to the different operations and routines that I needed during the implementation.

7. Technology enablers and implementation results

7.1 Technology enablers:

To develop my application, I tried to follow a minimalistic approach. This means that I had to find the technologies that will help us build a great application tool without having to add many additional packages. This way, the maintenance of the program will be easier. Thus, I decided on using

• Quickly: a framework for creating applications for a Linux distribution using Python, PyGTK and Glade Interface Designer. It allows developers to publish their applications using Launchpad.

• Python: Python is a high-level programming language designed to allow programmers to express concepts in less lines of code than C or Java. The language provides constructs intended to enable clear programs on both a small and large scale.
Python supports multiple programming paradigms including

- Object-oriented, imperative and functional programming or procedural styles
- Whitespace indentation, rather than curly braces or keywords, to delimit blocks.
- Regular Expressions

- MySQL Server: is the second most used relational database management system.

7.2 Implementation result

Once I had a clear and detailed scenario of what is supposed to be done in the back office of my application. Also, after choosing the right development model, which is the waterfall model and the different technology enablers to be used for the implementation, the next step was the implementation of the application. This part of the development of the application was the most exhausting past, as it took me 3 months of efforts and research to finalize the implementation.

After some tremendous nights of work and design, the results I was hoping to achieve was finally visible. At the end I found out that all the functional requirements and the non-functional requirements that I drew in the beginning of the project were met within the period agreed upon.

8. Future work

This application is just the beginning and the first generation of “Installation configuration system/tool for Hadoop”. I was able to meet all the functional and non-functional requirements of the project. In the future, I am planning to make this application available for every type of machines including 32 and 64bits. This tool works only for a single node cluster. This means that I will work on making the system suitable also for multi node clusters.

Moreover, I can extend this application to include the installation of Hive or any similar framework belonging to Apache. As already mentioned, I will try to contact Hadoop managers to all me to share this application legally and get some refund from them.

In addition, I will make the interface of my application more colorfull. I tried to add images to the interface, but it does not work for the time being. I will do more reading about “Quickly” to have done during the upcoming months.
All these suggestions are ones that I am seriously considering to work on in the future, as the time during the semester does not allow to work on all these aspects. However, as I am very interested in this idea I will keep up with the good work to get all these ideas to reality.

9. Conclusion

At the end of this report, I found that developing this application was a successful experience for me since it allowed me to get different skills related to both personal and professional fields.

Concerning my application, as discussed before, respects many of the STEEPLE considerations. It plays an important role in gaining a lot of time and stress; it will ease the task of having to deal with unsolvable errors. In addition, the application takes into consideration the fact that there could be novice users to Linux and system calls. It gives direct links to download the JDK, openSSH and Hadoop. Users do not need to look for download links for every prerequisite application.

Personally, I expanded my knowledge about Hadoop, its components and the way it works. First, I mastered the long process required to install Hadoop in a Linux machine. I learned how to find the best research methodology.

Professionally, I had the chance to work with a professional supervisor that took in consideration and respected my own ideas and way of tackling problems. During the last three month, I challenged myself to reach my goal and manage the stress of work. On the other hand, I experienced new techniques of implementation and design, and I was able to move from my comfort zone in working on small applications on a more professional level of development.

10. Appendix
10.1 Prerequisites

- Installing Sun Java version 7

As shown in the figure 1, the first prerequisite for installing Hadoop is to get JDK version 7 installed, which improves the performance, scalability and administration, according to oracle website. To download and install the version 7 of JDK, I used the following command:

```
Sudo apt-get install openjdk-7-jdk
```

Depending on the machine, the system installs either a normal 32bit version or amd64 version.
Figure 1: JDK being installed

- Configuring openSSH

According to openssh website, “OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other attacks. Additionally, OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions.”

As shown in figure 2, the installation of OpenSSH server component provides a server daemon tool to facilitate security, remote control encryption and file transfer operations. The OpenSSH server component, listens continuously for client connections from any of the client tools. When a connection request occurs, it sets up the correct connection depending on the type of client tool connecting. To install the OpenSSH server applications on Ubuntu system, I used the following command:

Sudo apt-get install openssh-server
As shown in figure 3, the openSSH client component is being installed. To install the OpenSSH client applications on Ubuntu system, I used the following command:

`sudo apt-get install openssh-client`

The SSH key permits authentication between two hosts without the need of a password. SSH key authentication uses both a private key and a public key.

To generate the keys, I used the following command:

`ssh-keygen -t rsa`
10.2 Hadoop

- Installation
  - Download

As shown in figure 5, I adopted version 1.2.1 of Hadoop since it is known as a stable version of this framework. The mirror link for downloading it:

Wget https://archive.apache.org/dist/hadoop/core/Hadoop-1.2.1/Hadoop-1.2.1-bin.tar.gz
Extraction

As shown in figure 6, the Hadoop tar file is extracted inside the Hadoop-tool folder. This operation takes around 60 seconds to be completed. To extract the file, I used the following command:

```
Sudo tar –xvf ~/Hadoop-tool/Hadoop-1.2.1.tar.gz
```

Figure 6: Hadoop being extracted

Update /.bashrc

According to the Linux website, “The shell program /bin/bash (hereafter referred to as just "the shell") uses a collection of startup files to help create an environment. Each file has a specific use and may affect login and interactive environments differently. The files in the /etc directory generally provide global settings. If an equivalent file exists in your home directory it may override the global settings.”

As shown in figure 7, I added the following lines to the end of the .bashrc file to have the environment recognize the Hadoop and java files.
export HADOOP_HOME=~/.Hadoop
export JAVA_HOME=/usr/lib/jvm/openjdk-7-jdk-amd64
export PATH=$PATH:$HADOOP_HOME/bin

- Configuration
  - Hadoop-env.sh

This configuration consists of defining a new environment variable for the JDK by updating it to:

JAVA_HOME=/usr/lib/jvm/openjdk-7-jdk-amd64

- Core-site.xml

This part starts by the creation of a temporary file and giving the user the right permissions and ownership.

```
sudo mkdir -p ~/Hadoop-tool/hadoop/temp
sudo chown $USER:$USER ~/Hadoop-tool/hadoop/temp
sudo chmod 750
```

![Image of Hadoop Tool interface]

*Figure 7: Creation of a temporary file*

This part consists of defining the path of the Hadoop folder through adding the following lines inside the configuration Tag:

```
<property>
```
<name>hadoop.tmp.dir</name>
<value>~/Hadoop-tool/Hadoop-1.2.1</value>
<description>A base for other temporary directories.</description>
</property>

<property>
 <name>fs.default.name</name>
 <value>hdfs://localhost: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>

- **hdfs-site.xml**

**Inside the HDFS file we should add:**

<property>
 <name>dfs.replication</name>
 <value>1</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>

- **mapred-site.xml**

**Inside the mapred file, the following lines should be added:**

<property>
 <name>mapred.job.tracker</name>
 <value>localhost: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>

After making all these modifications, the application shows the following message as shown in the figure 8:
Figure 8: Configuration of Hadoop files

- Formatting the HDFS file System

Before starting a Hadoop cluster, formatting the namenode using the following command is necessary.

```
Sudo ~/Hadoop-tool/Hadoop-1.2.1/bin namenode –format
```

Reminder: if you format the namenode as you will lose all data in the HDFS

The figure 9 shows the output of formatting the namenode. You need to check a message saying that it has been successfully formatted.

Figure 9: Formatting the NameNode
- **Starting Single node Cluster**

  To start the cluster, we should run the following command

  ```
  sudo ~/Hadoop-tool/Hadoop-1.2.1/bin/start-all.sh
  ```

  As shown in the figure 10, the namenode, Datanode, Jobtracker and a Tasktracker started in the machine.

  ![Figure 10: Running Hadoop](image)

- **JPS**

  As shown in the figure 11, the JPS tool is used to check if the Hadoop processes are correctly running.

  To call the JPS command:

  ```
  sudo ~/hadoop-tool/hadoop-1.2.1/jps
  ```
Figure 11: Checking JPS
11. References


