

Lab 01 Report: A Gentle Introduction to Hadoop

Teacher in charge: Nguyễn Ngọc Thảo.

Lab Instructors	Email
Đỗ Trọng Lễ	dtle@selab.hcmus.edu.vn
Bùi Huỳnh Trung Nam	huynhtrungnam2001@gmail.com

Group Name : Left4Dead

No.	Student ID	Student Name
1	21127329	Châu Tấn Kiệt
2	21127170	Nguyễn Thế Thiện
3	21127642	Trịnh Minh Long

Abstract

This lab aims to familiarize students with setting up a Hadoop cluster. Following this, students will explore Hadoop by developing a MapReduce program in Java, informed by study of the original paper on the MapReduce concept. Lastly, there are optional bonus assignments available for additional points.

Lab Progress

No.	Task	Expected output	Progress
1	Setting up Single-node Hadoop Cluster	Students can install a Hadoop cluster/instance on their own device.	100%
2	Introduction to MapReduce	Students can research new concepts to master how to express scientific concepts and understanding.	100%
3	Running a warm-up problem: Word Count	Students can verify their Hadoop cluster/instance is set up correctly and get used to run a MapReduce code in Hadoop.	100%
4	Bonus	4.1 Word Length Count	100%
		4.2 Setting up Fully Distributed Mode	0%

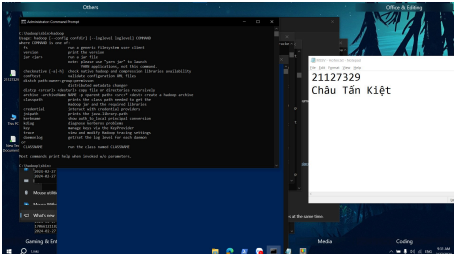
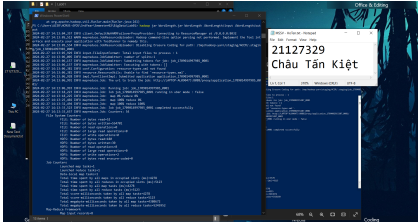
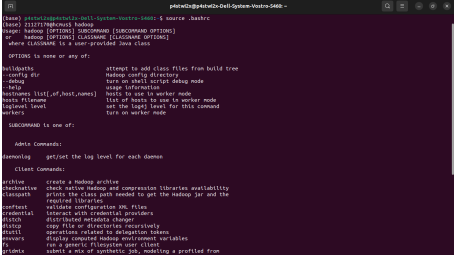
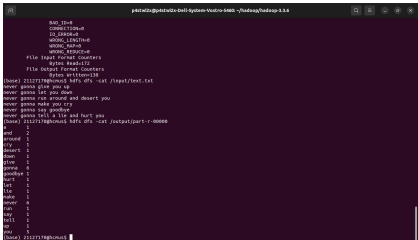
Reflection on our team:

Installing and configuring Hadoop turned out to be more problematic on some devices, despite having followed the guides closely such as:

- Namenodes/datanodes not starting: SSH exited code 1 (directory not permitted access or in inconsistent state) -> Overcome: custom configuration for namenode/datanode directories.
- RCMD configuration not matching: RCMD/socket permission denied -> Overcome: differences between RSH and SSH for additional configurations.[1]
- Bug series (especially in the case of multiple users): Connection permission denied (publickey/password); SSH exited code 255; port 9000 not opening (HDFS UI); namenode/datanodes recognized by JPS but not by yarn -> Overcome: not installing Hadoop into root directory (/usr/local/hadoop/), in order not to mess up SSH and file permission as much, especially for those who are not familiar with manual network configuration in Linux distributions.
- Hadoop crashing in usage due to low RAM -> Overcome: temporary virtual RAM, HDFS properties customization.[2]
What we have learnt:
- Limiting error possibility through log files helps a lot in searching for (online) solutions, thus greatly reduces time and risk testing incorrect solutions.

1. Setting up Single-node Hadoop Cluster

In this exercise, each member has installed a single node Hadoop cluster by following the tutorial from Apache Hadoop’s official documentation. When following the tutorial, the student needs to take screenshots of the installation and verify if Hadoop is installed correctly. The evidence is shown as below. The details of the images are available in the path /docs/images

No.	Student ID	Student Name	Hadoop command	Testing
1	21127329	Châu Tấn Kiệt		
2	21127170	Nguyễn Thế Thiện		

No.	Student ID	Student Name	Hadoop command	Testing
3	21127642	Trinh Minh Long	<pre> tel_211276420 > <dragonform-~workspace/Big_data/Bigdata(git>bin))> ls tel_211276420 > <dragonform-~workspace/Big_data/Bigdata(git>bin))> ls Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS] hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS] where CLASSNAME is a user-provided Java class (OPTIONS is none or any of): -config dir Hadoop config directory -debug turn on shell script debug mode -help usage information -libbdir attempt to add class files from build tree -hostnames List[,of,host,names] hosts to use in worker mode -hosts file list of hosts to use in worker mode -loglevel level set the log() level for this command -workers turn on worker mode SUBCOMMAND is one of: Admin Commands: daemonlog get/set the log level for each daemon Client Commands: archive create a Hadoop archive checknative check native Hadoop and compression libraries availability classpath prints the class path needed to get the Hadoop jar and the required libraries coexist validate configuration XML files credential interact with credential providers distcp distributed metadata changer distcp copy file or directory, recursively distcp operations related to delegation tokens envvars display computed Hadoop environment variables fs run a generic filesystem user client genfsimage submit a mix of synthetic job, modeling a profiled from production load jar run a jar file. NOTE: please use 'spark jar' to launch spark applications, not this command. jar prints the java.library.path jdi Diagnose JavaBeans problems kerberos show authn, local principal conversion key manage keys via the keyprovider kubernetes scale a kubernetes trace kubernetes convert log into a kubernetes trace </pre>	<pre> tel_211276420 > <dragonform-~workspace/Big_data/Bigdata(git>bin))> ls tel_211276420 > <dragonform-~workspace/Big_data/Bigdata(git>bin))> ls Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS] hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS] where CLASSNAME is a user-provided Java class (OPTIONS is none or any of): -config dir Hadoop config directory -debug turn on shell script debug mode -help usage information -libbdir attempt to add class files from build tree -hostnames List[,of,host,names] hosts to use in worker mode -hosts file list of hosts to use in worker mode -loglevel level set the log() level for this command -workers turn on worker mode SUBCOMMAND is one of: Admin Commands: daemonlog get/set the log level for each daemon Client Commands: archive create a Hadoop archive checknative check native Hadoop and compression libraries availability classpath prints the class path needed to get the Hadoop jar and the required libraries coexist validate configuration XML files credential interact with credential providers distcp distributed metadata changer distcp copy file or directory, recursively distcp operations related to delegation tokens envvars display computed Hadoop environment variables fs run a generic filesystem user client genfsimage submit a mix of synthetic job, modeling a profiled from production load jar run a jar file. NOTE: please use 'spark jar' to launch spark applications, not this command. jar prints the java.library.path jdi Diagnose JavaBeans problems kerberos show authn, local principal conversion key manage keys via the keyprovider kubernetes scale a kubernetes trace kubernetes convert log into a kubernetes trace </pre>

2. Paper Reading

We read and analyzed the original paper of MapReduce by Dean and Ghemawat and then answer the given questions:

1. How do the input keys-values, the intermediate keys-values, and the output keys-values relate?

The input key-value pairs are made by dividing the input data into smaller blocks and mapping those blocks with corresponding key-value pairs, while output key-value pairs are from the Reduce section of MapReduce. Hence the input keys and values are drawn from a different domain than the output keys and values.

Furthermore, the intermediate keys and values are from the same domain as the output keys and values, since the they are both outputs of functions which the inputs are in the type of key-value pairs.

2. How does MapReduce deal with node failures?

- Worker Failure
The Master sets the status of its currently executing Reduce tasks to idle
If a worker node fails, the master reschedules the tasks handled by the worker.
- Master Failure
The whole MapReduce job gets restarted through a different master

3. What is the meaning and implication of locality? What does it use?

- The meaning of locality is the input data (managed by GFS) is stored on the local disks of the machines that make up the cluster.
- The implication is that it saves network bandwidth.
- This process is used by the local disks

4. Which problem is addressed by introducing a combiner function to the MapReduce model?

- One disadvantage of MapReduce is its inefficiency and redundancy for some data processing tasks.

- MapReduce can generate a lot of intermediate data, which needs to be shuffled, sorted, and transferred between nodes.
- Combiner function minimizes the number of key/value pairs that will be shuffled across the network and provided as input to the Reducer

3. Running a warm-up problem: Word Count

In this section, we follow the tutorial to get the Example WordCount v1.0 [3]. Then we compiled the code to a JAR file, then run them in the installed Hadoop cluster/instance

We run the example in 2 operating systems, Windows and Linux with the command:

`hadoop jar <.jar file> WordCount <input directory in HDFS> <output directory in HDFS>`

Here's the example for input and output. The details of the images are available in the path `/docs/images :`

- Step 1: Create a file named “WordCount.java” in your hadoop folder
- Step 2: Copy code from the tutorial into the file
- Step 3: Compile the java file into jar file with the command
`jar cf <jar file name>.jar <word count file name>.java`

```
tml_21127642@ > <dragonfarm1 ~/hadoop/hadoop-3.3.6> nvim WordCount.java
tml_21127642@ > <dragonfarm1 ~/hadoop/hadoop-3.3.6> jar cf wc.jar WordCount*.class
tml_21127642@ > <dragonfarm1 ~/hadoop/hadoop-3.3.6> ls
LICENSE-binary  README.txt      WordCount.java  lib/            sbin/
LICENSE.txt     'WordCount$IntSumReducer.class'  bin/            libexec/        share/
NOTICE-binary   'WordCount$TokenizerMapper.class'  etc/            licenses-binary/ wc.jar
NOTICE.txt      WordCount.class  include/         logs/
tml_21127642@ > <dragonfarm1 ~/hadoop/hadoop-3.3.6> hdfs dfs -mkdir /input
tml_21127642@ > <dragonfarm1 ~/hadoop/hadoop-3.3.6> cd
tml_21127642@ > <dragonfarm1 ~> nvim input3.txt
```

Compile Java file

- Step 4: Create a input folder for our input file
- Step 5: Create a input file anywhere you want

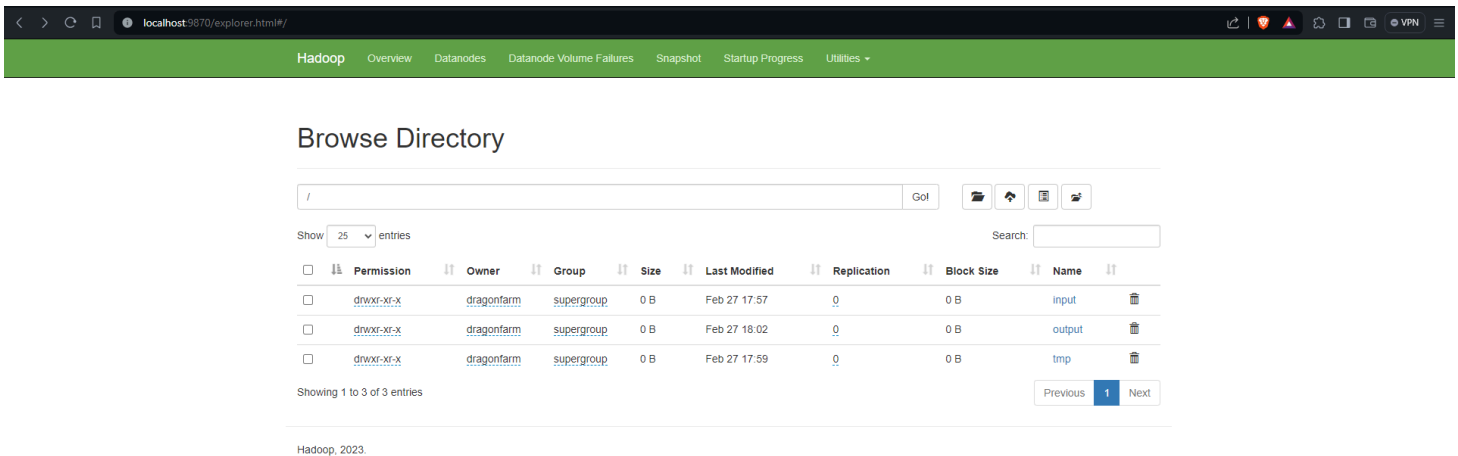
```
nvim input3.txt ~
Hello Hadoop Goodbye Hadoop
~
~
```

The content of the input file

```
tml_21127642@ > <dragonfarm1 ~/hadoop/hadoop-3.3.6> cd
tml_21127642@ > <dragonfarm1 ~> nvim input3.txt
tml_21127642@ > <dragonfarm1 ~> hdfs dfs -put input3.txt /input
tml_21127642@ > <dragonfarm1 ~> hdfs dfs -cat /input/input3.txt
Hello Hadoop Goodbye Hadoop
tml_21127642@ > <dragonfarm1 ~>
```

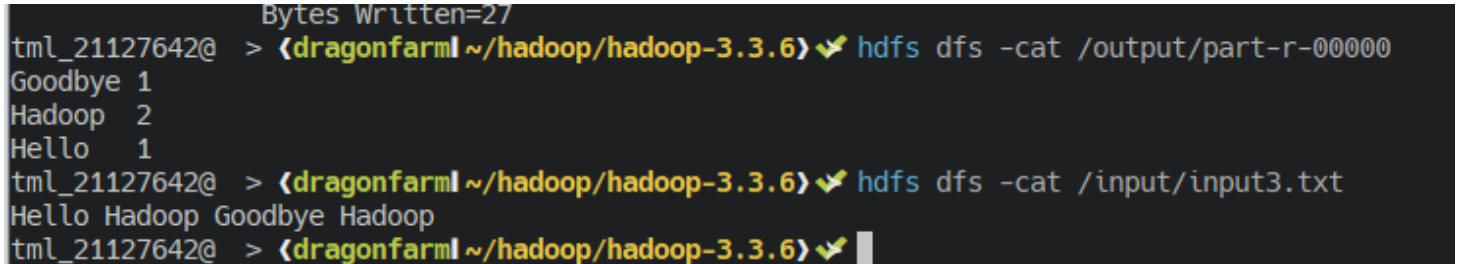
Putting input file inside the input folder

- Step 6: Put the input3.txt file into the input folder with the command
hdfs dfs -put <name of your input file>.txt <your input folder name>
- Step 7: Go into the hadoop folder and run the application with the command
hadoop jar <.jar file> WordCount <input directory in HDFS> <output directory in HDFS>
- Step 8: Check the website for input and output folder.



Website now should have both input folder and output folder

- Step 9: Open the terminal
- Step 10: Enter command `hdfs dfs -cat /output/part-r-00000` to check the result.



Result

4. Bonus

4.1. Word Length Count

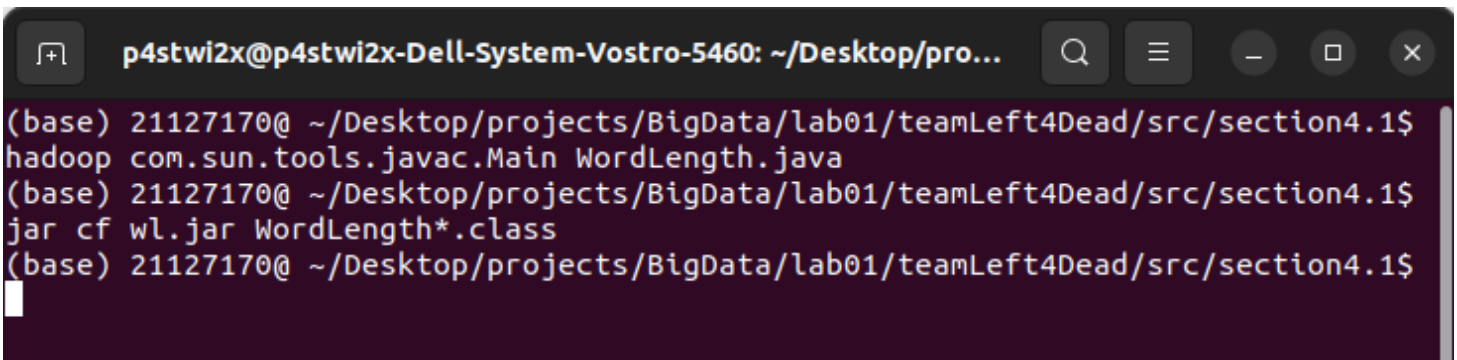
In this section we will be categorize words using a simple program, the code is available at `teamLeft4Dead/src/section4.1/WordLength.java`.

- Step 1: we start Hadoop as usual.

```
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as p4stwi2x in 10 seconds
.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
jpsStarting namenodes on [localhost]

Starting datanodes
Starting secondary namenodes [p4stwi2x-Dell-System-Vostro-5460]
Starting resourcemanager
Starting nodemanagers
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
jps
27907 Jps
27619 ResourceManager
27180 DataNode
27741 NodeManager
27053 NameNode
27389 SecondaryNameNode
```

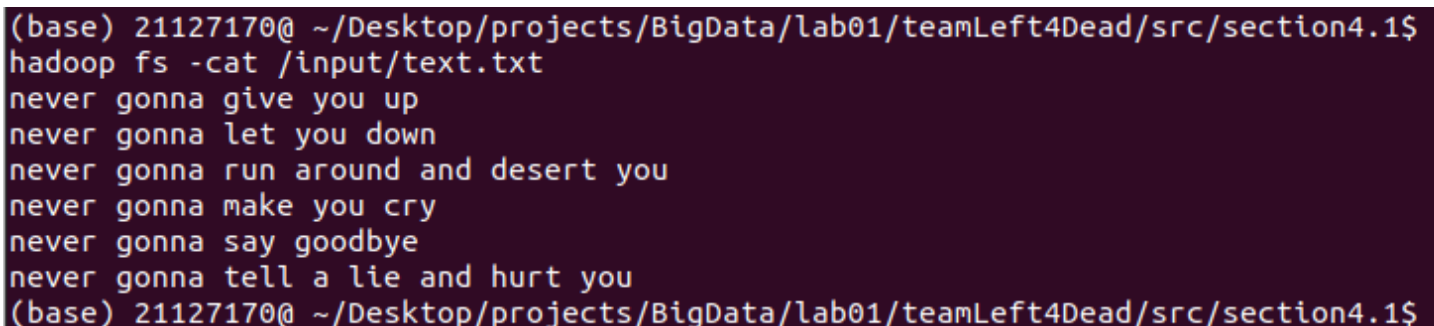
- Step 2: we compile the `WordLength.java`



A terminal window titled "p4stwi2x@p4stwi2x-Dell-System-Vostro-5460: ~/Desktop/pro..." with standard window controls. The terminal shows the following commands and output:

```
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
hadoop com.sun.tools.javac.Main WordLength.java
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
jar cf wl.jar WordLength*.class
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
```

- Step 3: we run the program.
This is the input.



A terminal window showing the output of the `hadoop fs -cat` command. The output displays the contents of a file named `/input/text.txt`, which contains the lyrics of the song "Never Gonna Give You Up".

```
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
hadoop fs -cat /input/text.txt
never gonna give you up
never gonna let you down
never gonna run around and desert you
never gonna make you cry
never gonna say goodbye
never gonna tell a lie and hurt you
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
```

We turn off safe mode as admin so we could run the program.


```
p4stwi2x@p4stwi2x-Dell-System-Vostro-5460: ~/Desktop/pro...
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
hadoop dfsadmin -safemode leave
WARNING: Use of this script to execute dfsadmin is deprecated.
WARNING: Attempting to execute replacement "hdfs dfsadmin" instead.

Safe mode is OFF
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
hadoop jar wl.jar WordLength /input /output
2024-03-03 20:00:41,302 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecti
ng to ResourceManager at /0.0.0.0:8032
2024-03-03 20:00:41,805 WARN mapreduce.JobResourceUploader: Hadoop command-line
option parsing not performed. Implement the Tool interface and execute your appl
ication with ToolRunner to remedy this.
2024-03-03 20:00:41,857 INFO mapreduce.JobResourceUploader: Disabling Erasure Co
```

This is the output.

```
p4stwi2x@p4stwi2x-Dell-System-Vostro-5460: ~/Desktop/pro...
Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=172
File Output Format Counters
    Bytes Written=26
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
hadoop fs -ls /output
Found 2 items
-rw-r--r--   1 p4stwi2x supergroup          0 2024-03-03 20:01 /output/_SUCCESS
-rw-r--r--   1 p4stwi2x supergroup        26 2024-03-03 20:01 /output/part-r-00
000
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
hadoop fs -cat /output/part-r-00000
Medium  15
Small   18
Tiny     1
(base) 21127170@ ~/Desktop/projects/BigData/lab01/teamLeft4Dead/src/section4.1$
```

5. References

[1] StackOverflow, 24 Jan 2018, "Permission Denied error while running start-dfs.sh", solution by "int32", <https://stackoverflow.com/a/48415037>, last visited: 1 Mar 2024

[2] SavvyNik, 25 Sep 2020, "How to Create, Resize, or Extend a Linux Swap File | (Ubuntu)", https://www.youtube.com/watch?v=HSbBl31ohjE&ab_channel=SavvyNik, last visited: 2 Mar 2024

[3] Apache Hadoop, "MapReduce Tutorial", <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>, last visited: 2 Mar 2024