Lab 01: A Gentle Introduction to Hadoop

CSC14118 Introduction to Big Data 20KHMT1

The Girls

2023-02-17

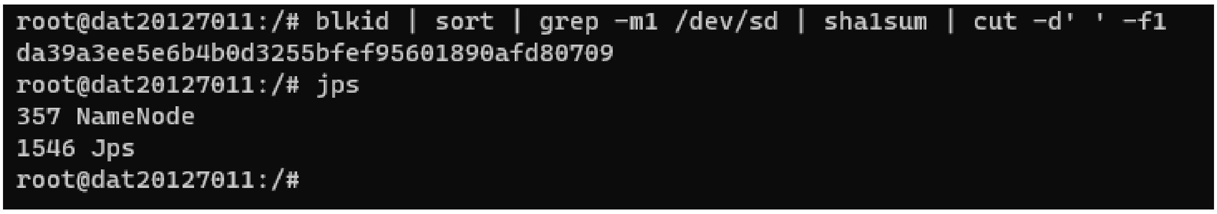
Contents

1 Lab 01: A Gentle Introduction to Hadoop

1.1 Setting up Single-node Hadoop Cluster

Verify hadoop installation for each member of the group

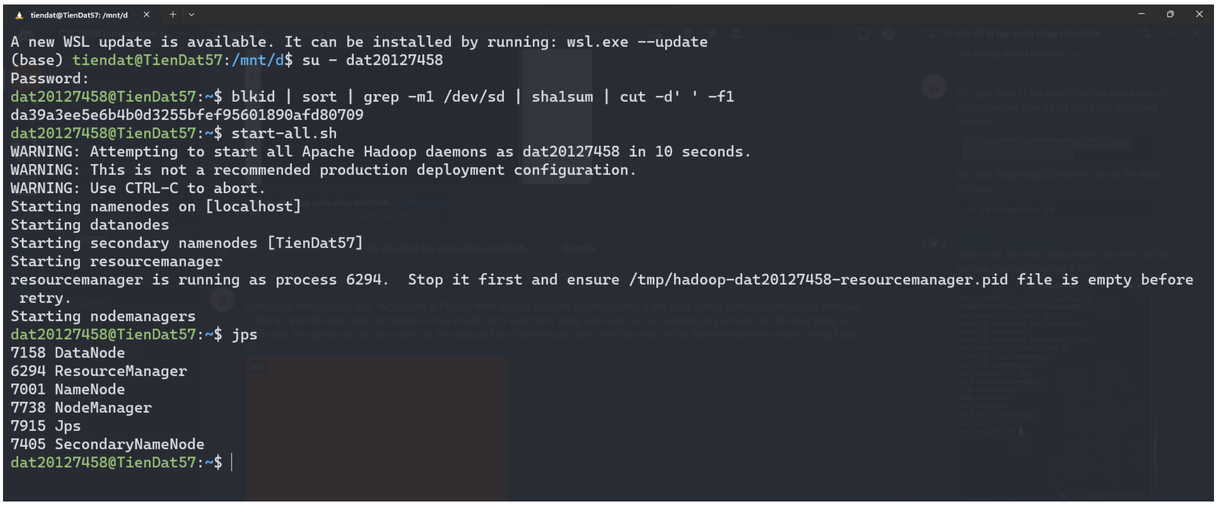
> 20127011 **Le Tan Dat**



```
root@dat20127011:/# blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@dat20127011:/# jps
357 NameNode
1546 Jps
root@dat20127011:/#
```

Figure 1.1: 01-20127458

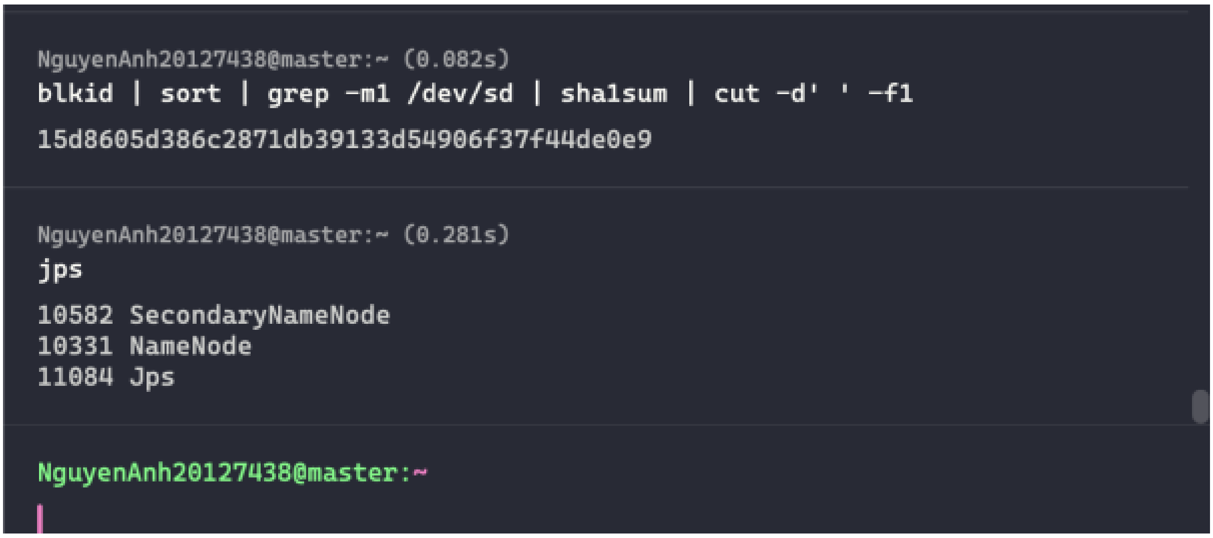
> 20127458 **Dang Tien Dat**



```
A new WSL update is available. It can be installed by running: wsl.exe --update
(base) tiendat@TienDat57:/mnt/d$ su - dat20127458
Password:
dat20127458@TienDat57:~$ blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
dat20127458@TienDat57:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as dat20127458 in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [TienDat57]
Starting resourcemanager
resourcemanager is running as process 6294.  Stop it first and ensure /tmp/hadoop-dat20127458-resourcemanager.pid file is empty before
 retry.
Starting nodemanagers
dat20127458@TienDat57:~$ jps
7158 DataNode
6294 ResourceManager
7001 NameNode
7738 NodeManager
7915 Jps
7405 SecondaryNameNode
dat20127458@TienDat57:~$ 
```

Figure 1.2: 01-20127458

> `20127438` **Le Nguyen Nguyen Anh**



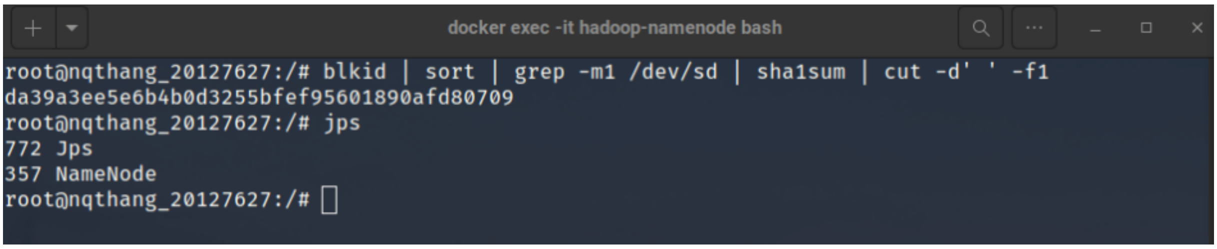
```
NguyenAnh20127438@master:~ (0.082s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@master:~ (0.281s)
jps
10582 SecondaryNameNode
10331 NameNode
11084 Jps

NguyenAnh20127438@master:~
```

Figure 1.3: 01-20127438

> `20127627` **Nguyen Quoc Thang**



```
root@nqthang_20127627:/# blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@nqthang_20127627:/# jps
772 Jps
357 NameNode
root@nqthang_20127627:/# 
```

Figure 1.4: 01-20127627

1.2 Introduction to MapReduce

This a section we will answer the following questions:

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

   Answer: In a MapReduce job, the input keys-value (represent the input data that needs to be processed) are processed by a map function to produce intermediate key-value pairs (the value represent the data that is associated with each key). These intermediate key-value pairs are then sorted by key and passed on to the reduce function, which groups the values associated with each intermediate key and produces the final output key-value pairs.

2. **How does MapReduce deal with node failures?**

   Answer: MapReduce handles the fault node in a fault tolerant manner. When a node fails during execution, the tasks running on the node are automatically rescheduled to run on other nodes in the cluster. There are 2 mechanisms in MapReduce to handle node errors including: Speculative Execution and Task Tracking as follow.

   - **Speculative Execution**: MapReduce can launch duplicate copies of a task on different nodes to ensure that at least one copy of the task completes successfully. If one of the nodes fails or is slow to complete its task, the duplicate copy can take over and complete the work.
   - **Task Tracking**: MapReduce tracks the progress of each task and can detect when a task is taking too long to complete. If a task is taking too long, MapReduce can launch a duplicate copy of the task on a different node. If the duplicate copy completes successfully, the original task is killed.

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

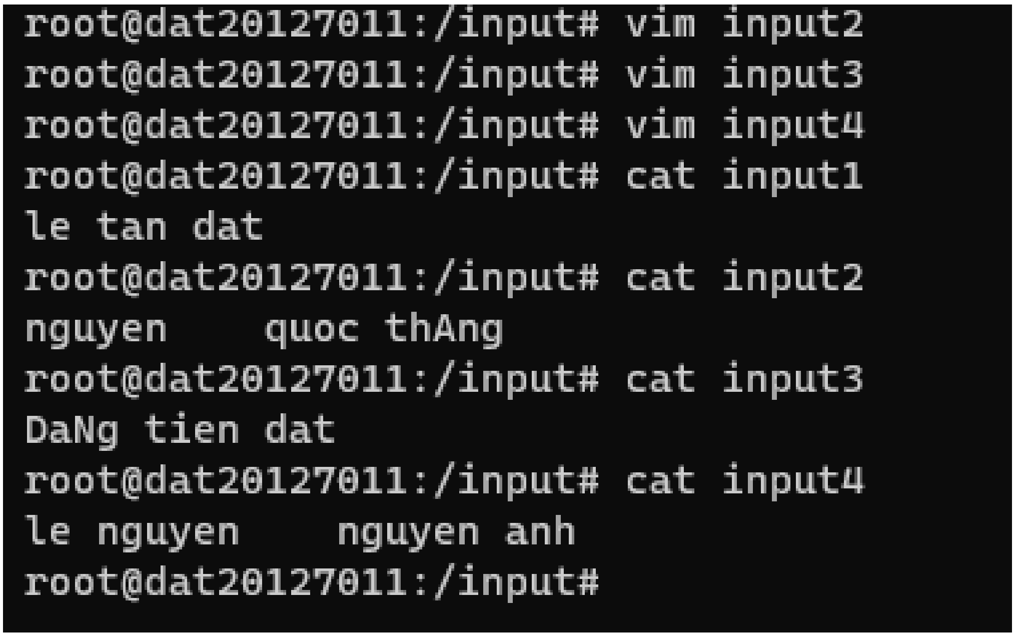
   Answer:

   - **Meaning of locality**: Locality in Hadoop refers to the ability to process data on the same node or machine where the data is stored, in order to avoid data transmission between nodes in the network.
   - **Implication of locality**: Locality is an important feature of Hadoop to optimize data processing performance by minimizing the time it takes to transmit data over the network.
   - **Used for**: Locality is used in Hadoop to optimize data processing performance by ensuring that data processing tasks are performed on the same node where the data is stored. This helps to minimize the time it takes to transmit data over the network and improve data processing performance in Hadoop.

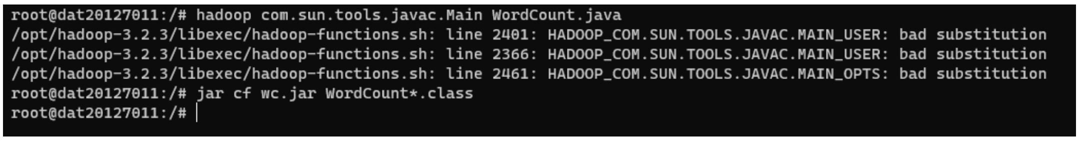
4. **Which problem is addressed by introducing a combiner function to the MapReduce model?**

   Answer: The introduction of a combiner function to the MapReduce model addresses the problem of excessive data shuffling and network traffic during the Reduce phase. The combiner function is used to reduce the amount of data that needs to be transferred between the Map and Reduce tasks in a MapReduce job. The combiner function is executed on the output of the Map task on each node before the data is transferred to the Reduce task. The combiner function is optional and is only used if it reduces the amount of data that needs to be transferred between the Map and Reduce tasks.

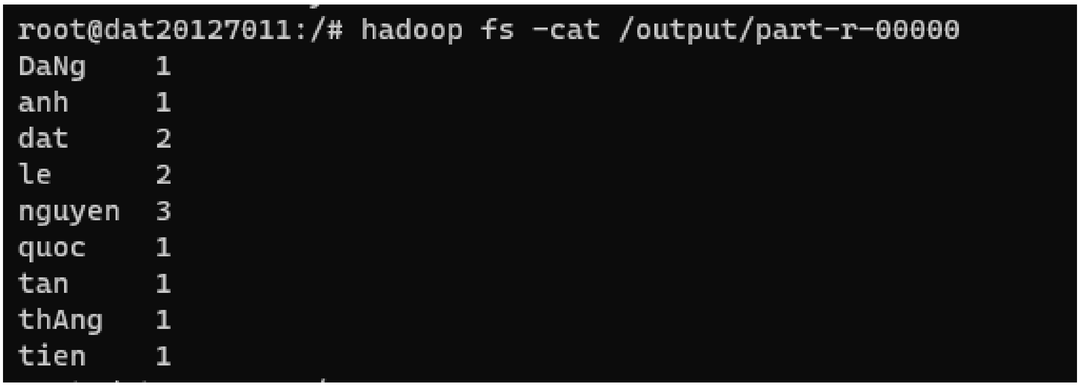
1.3 Running a warm-up problem: Word Count

- **Step 1**: Create input1 file
  - 

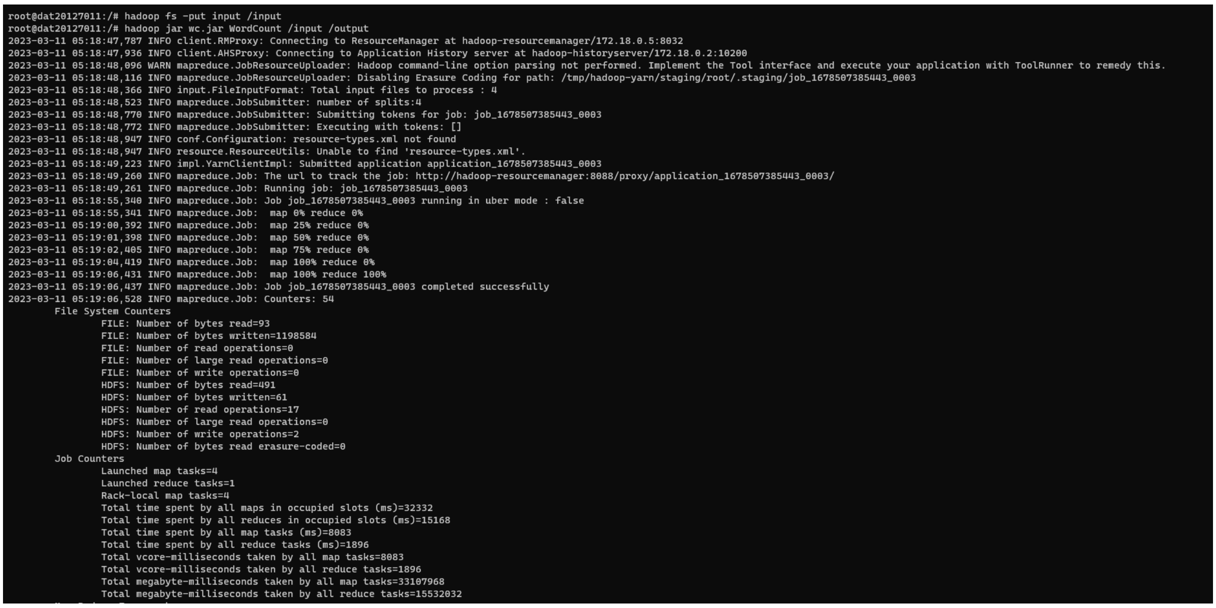
```
root@dat20127011:/input# vim input2
root@dat20127011:/input# vim input3
root@dat20127011:/input# vim input4
root@dat20127011:/input# cat input1
le tan dat
root@dat20127011:/input# cat input2
nguyen    quoc thAng
root@dat20127011:/input# cat input3
DaNg tien dat
root@dat20127011:/input# cat input4
le nguyen    nguyen anh
root@dat20127011:/input#
```

- **Step 2**: Compile WordCount.java and create a jar
  - 

```
root@dat20127011:/# hadoop com.sun.tools.javac.Main WordCount.java
/opt/hadoop-3.2.3/libexec/hadoop-functions.sh: line 2401: HADOOP_COM.SUN.TOOLS.JAVAC.MAIN_USER: bad substitution
/opt/hadoop-3.2.3/libexec/hadoop-functions.sh: line 2366: HADOOP_COM.SUN.TOOLS.JAVAC.MAIN_USER: bad substitution
/opt/hadoop-3.2.3/libexec/hadoop-functions.sh: line 2461: HADOOP_COM.SUN.TOOLS.JAVAC.MAIN_OPTS: bad substitution
root@dat20127011:/# jar cf wc.jar WordCount*.class
root@dat20127011:/# |
```

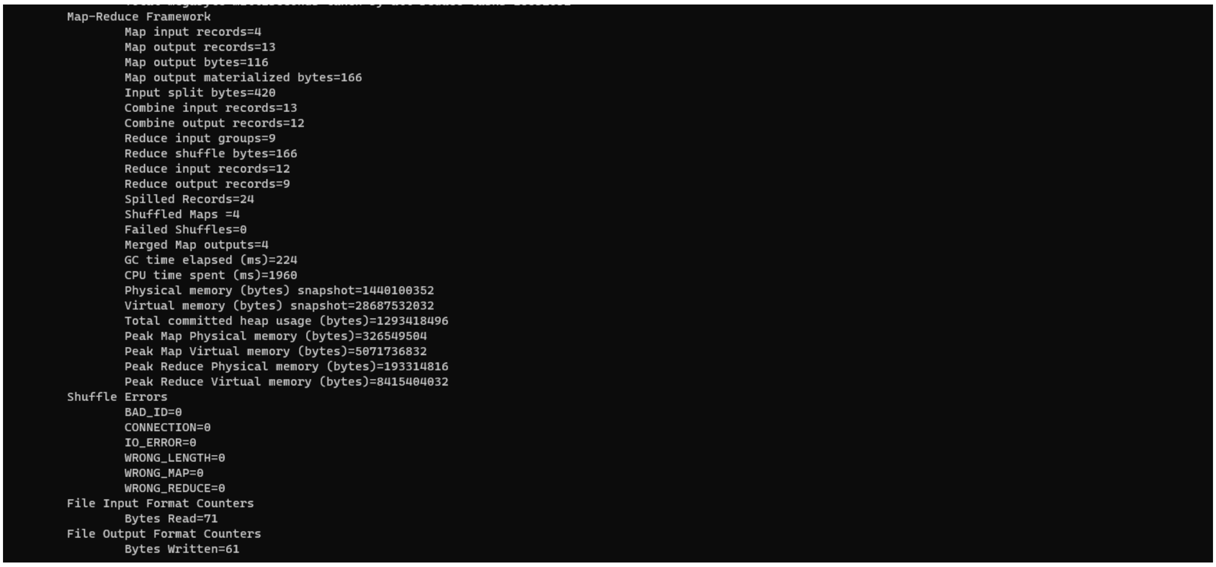
- **Step 3**: Run the application
  -
  -
- **Step 4**: Output file
  - 

```
root@dat20127011:/# hadoop fs -cat /output/part-r-00000
DaNg    1
anh     1
dat     2
le      2
nguyen  3
quoc    1
tan     1
thAng   1
tien    1
```



```
root@dat20127011:/# hadoop fs -put input /input
root@dat20127011:/# hadoop jar wc.jar WordCount /input /output
2023-03-11 05:18:47,787 INFO client.RMProxy: Connecting to ResourceManager at hadoop-resourcemanager/172.18.0.5:8032
2023-03-11 05:18:47,936 INFO client.AHSProxy: Connecting to Application History server at hadoop-historyserver/172.18.0.2:10200
2023-03-11 05:18:48,096 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2023-03-11 05:18:48,116 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1678507385443_0003
2023-03-11 05:18:48,366 INFO input.FileInputFormat: Total input files to process : 4
2023-03-11 05:18:48,523 INFO mapreduce.JobSubmitter: number of splits:4
2023-03-11 05:18:48,770 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1678507385443_0003
2023-03-11 05:18:48,772 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-03-11 05:18:48,947 INFO conf.Configuration: resource-types.xml not found
2023-03-11 05:18:48,947 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-03-11 05:18:49,223 INFO impl.YarnClientImpl: Submitted application application_1678507385443_0003
2023-03-11 05:18:49,260 INFO mapreduce.Job: The url to track the job: http://hadoop-resourcemanager:8088/proxy/application_1678507385443_0003/
2023-03-11 05:18:49,261 INFO mapreduce.Job: Running job: job_1678507385443_0003
2023-03-11 05:18:55,340 INFO mapreduce.Job: Job job_1678507385443_0003 running in uber mode : false
2023-03-11 05:18:55,341 INFO mapreduce.Job:  map 0% reduce 0%
2023-03-11 05:19:00,392 INFO mapreduce.Job:  map 25% reduce 0%
2023-03-11 05:19:01,398 INFO mapreduce.Job:  map 50% reduce 0%
2023-03-11 05:19:02,405 INFO mapreduce.Job:  map 75% reduce 0%
2023-03-11 05:19:04,419 INFO mapreduce.Job:  map 100% reduce 0%
2023-03-11 05:19:06,431 INFO mapreduce.Job:  map 100% reduce 100%
2023-03-11 05:19:06,437 INFO mapreduce.Job: Job job_1678507385443_0003 completed successfully
2023-03-11 05:19:06,528 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=93
                FILE: Number of bytes written=1198584
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=491
                HDFS: Number of bytes written=61
                HDFS: Number of read operations=17
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=4
                Launched reduce tasks=1
                Rack-local map tasks=4
                Total time spent by all maps in occupied slots (ms)=32332
                Total time spent by all reduces in occupied slots (ms)=15168
                Total time spent by all map tasks (ms)=8083
                Total time spent by all reduce tasks (ms)=1896
                Total vcore-milliseconds taken by all map tasks=8083
                Total vcore-milliseconds taken by all reduce tasks=1896
                Total megabyte-milliseconds taken by all map tasks=33107968
                Total megabyte-milliseconds taken by all reduce tasks=15532032
```

Figure 1.5: 03-step3.1



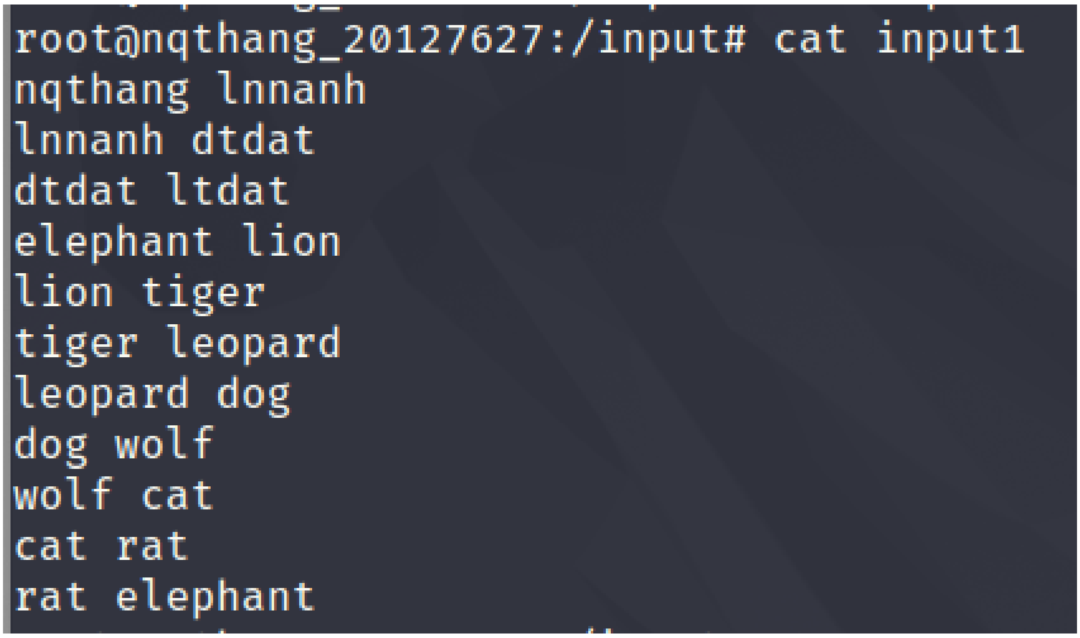
```
        Map-Reduce Framework
                Map input records=4
                Map output records=13
                Map output bytes=116
                Map output materialized bytes=166
                Input split bytes=420
                Combine input records=13
                Combine output records=12
                Reduce input groups=9
                Reduce shuffle bytes=166
                Reduce input records=12
                Reduce output records=9
                Spilled Records=24
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=224
                CPU time spent (ms)=1960
                Physical memory (bytes) snapshot=1440100352
                Virtual memory (bytes) snapshot=28687532032
                Total committed heap usage (bytes)=1293418496
                Peak Map Physical memory (bytes)=326549504
                Peak Map Virtual memory (bytes)=5071736832
                Peak Reduce Physical memory (bytes)=193314816
                Peak Reduce Virtual memory (bytes)=8415404032
        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=71
        File Output Format Counters
                Bytes Written=61
```

Figure 1.6: 03-step3.2

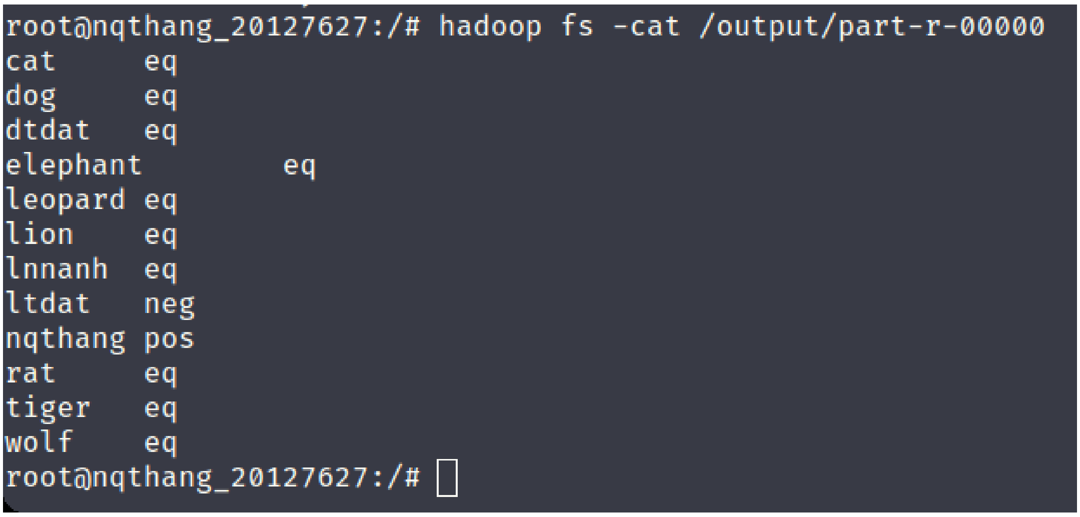
1.4 Bonus

1.4.1 Extended Word Count: Unhealthy relationships

After create file `Unhealthy_relationships.java` and `input.txt`

- **File `input.txt`**:
  - 

```
root@nqthang_20127627:/input# cat input1
nqthang lnnanh
lnnanh dtdat
dtdat ltdat
elephant lion
lion tiger
tiger leopard
leopard dog
dog wolf
wolf cat
cat rat
rat elephant
```

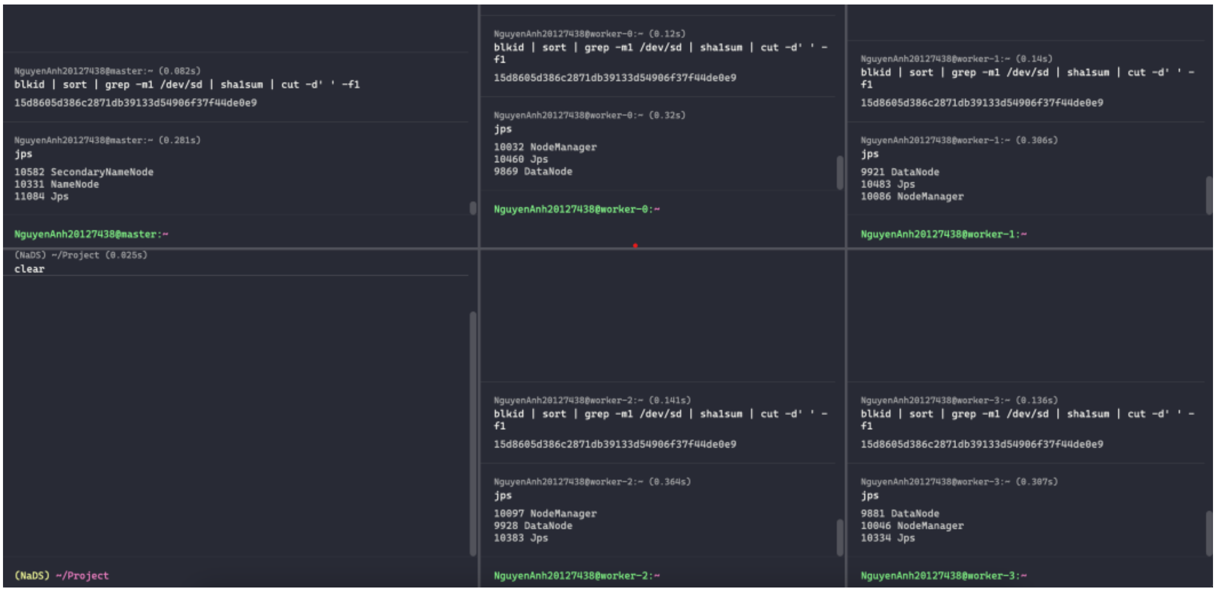
- **After run `Unhealthy_relationships.java`**, we have the result:
  - 

```
root@nqthang_20127627:/# hadoop fs -cat /output/part-r-00000
cat     eq
dog     eq
dtdat   eq
elephant        eq
leopard eq
lion    eq
lnnanh  eq
ltdat   neg
nqthang pos
rat     eq
tiger   eq
wolf    eq
root@nqthang_20127627:/# 
```

1.4.2 Setting up Fully Distributed Mode

1.4.2.1 Hadoop Cluster Setup in Non-Secure Mode

- **This section includes the machine id image of each machine:**



```
NguyenAnh20127438@master:~ (0.082s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@master:~ (0.281s)
jps
10582 SecondaryNameNode
10331 NameNode
11084 Jps

NguyenAnh20127438@master:~
```

```
(NaDS) ~/Project (0.025s)
clear

(NaDS) ~/Project
```

```
NguyenAnh20127438@worker-0:~ (0.12s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -
f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@worker-0:~ (0.32s)
jps
10032 NodeManager
10460 Jps
9869 DataNode

NguyenAnh20127438@worker-0:~
```

```
NguyenAnh20127438@worker-1:~ (0.14s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -
f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@worker-1:~ (0.306s)
jps
9921 DataNode
10483 Jps
10086 NodeManager

NguyenAnh20127438@worker-1:~
```

```
NguyenAnh20127438@worker-2:~ (0.141s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -
f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@worker-2:~ (0.364s)
jps
10097 NodeManager
9928 DataNode
10383 Jps

NguyenAnh20127438@worker-2:~
```

```
NguyenAnh20127438@worker-3:~ (0.136s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -
f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@worker-3:~ (0.307s)
jps
9881 DataNode
10046 NodeManager
10334 Jps

NguyenAnh20127438@worker-3:~
```

Figure 1.7: Machine 20127438

    -
    -

1.4.2.2 Research about Security in Hadoop Set-up

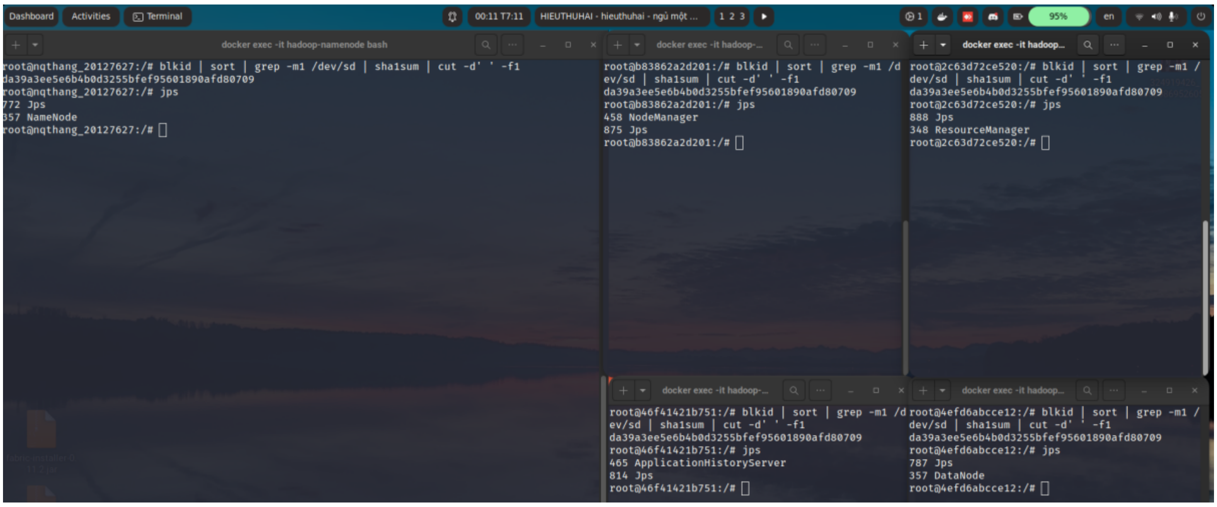
This a section we will answer the following questions:

1. **Is your Hadoop secured? Give a short explanation if your answer is yes. Otherwise, give some examples of risks to your system.**

   Answer: Encryption protects the data stored in Hadoop by making it unreadable to anyone who does not have the decryption key.

   Some examples of risks to a Hadoop system include:

   - Unauthorized access: If a Hadoop system is not properly secured, it can be accessed by unauthorized users who can steal, modify, or delete data.


Dashboard
Activities
Terminal
00:11 T7:11
HIEUTHUHAI - hieuthuhai - ngủ một ...
1 2 3
1
95%
en
docker exec -it hadoop-namenode bash
docker exec -it hadoop-...
docker exec -it hadoop...
docker exec -it hadoop-...
docker exec -it hadoop...

```
root@nqthang_20127627:/# blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@nqthang_20127627:/# jps
772 Jps
357 NameNode
root@nqthang_20127627:/# 
```

```
root@b83862a2d201:/# blkid | sort | grep -m1 /d
ev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@b83862a2d201:/# jps
458 NodeManager
875 Jps
root@b83862a2d201:/# 
```

```
root@2c63d72ce520:/# blkid | sort | grep -m1 /
dev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@2c63d72ce520:/# jps
888 Jps
348 ResourceManager
root@2c63d72ce520:/# 
```

```
root@46f41421b751:/# blkid | sort | grep -m1 /d
ev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@46f41421b751:/# jps
465 ApplicationHistoryServer
814 Jps
root@46f41421b751:/# 
```

```
root@4efd6abcce12:/# blkid | sort | grep -m1 /
dev/sd | sha1sum | cut -d' ' -f1
da39a3ee5e6b4b0d3255bfef95601890afd80709
root@4efd6abcce12:/# jps
787 Jps
357 DataNode
root@4efd6abcce12:/# 
```

Figure 1.8: Machine 20127627

- Data breaches: Hadoop systems that store sensitive information such as financial or personal data are at risk of data breaches. If data is not properly secured, it can be stolen by attackers and used for malicious purposes.
- Malware and viruses: Hadoop systems can be vulnerable to malware and viruses that can infect the system and compromise its security.
- Insider threats: Insiders with access to Hadoop systems can intentionally or unintentionally cause harm to the system by stealing or modifying data, or by introducing malware or viruses.

To prevent these risks, Hadoop administrators should implement security measures such as access controls, encryption, and security monitoring.

2. **From your perspective, which method is better when securing your HDFS: authentication, authorization, or encryption? Give an explanation about your choices.**

   Answer: I think the authentication is better than authorization and encryption. Because the authentication is the first step to access the Hadoop. If I don't have the authentication, I can't access the Hadoop. So, I think the authentication is better than authorization and encryption. From my perspective, it is not possible to say that one method is better than the other when securing HDFS, as all three methods play important and complementary roles in overall Hadoop security.

   Authentication is the process of verifying the identity of a user or application attempting to access the HDFS. Authentication ensures that only authorized users can access the system, and helps prevent unauthorized access and data breaches. Authorization ensures that users can only access the data and resources that they are authorized to use, and helps prevent unauthorized access and data breaches. Without proper authorization, users may be able to access data or resources that they should not have access to, leading to security vulnerabilities.

Encryption is the process of converting data into a format that is unreadable to unauthorized users. Encryption ensures that data is protected, even if it is accessed by unauthorized users. Encryption is especially important for sensitive data that needs to be protected at rest or in transit.

To secure HDFS, it is important to use all three methods - authentication, authorization, and encryption - in conjunction with each other. Proper authentication and authorization ensure that only authorized users can access data, while encryption ensures that the data is protected, even if it is accessed by unauthorized users.

Insert table example:

> Thang - 20127627 Docker Virtual Machine

Server IP Address	Ports Open
172.18.0.6 Namenode	**TCP**: 9870, 9000
172.18.0.4 Nodemanager	**TCP**: *
172.18.0.2 Resourcemanager	**TCP**: 8088, 8030, 8031, 8032
172.18.0.3 Historyserver	**TCP**: 8188
172.18.0.5 Datanode	**TCP**: *

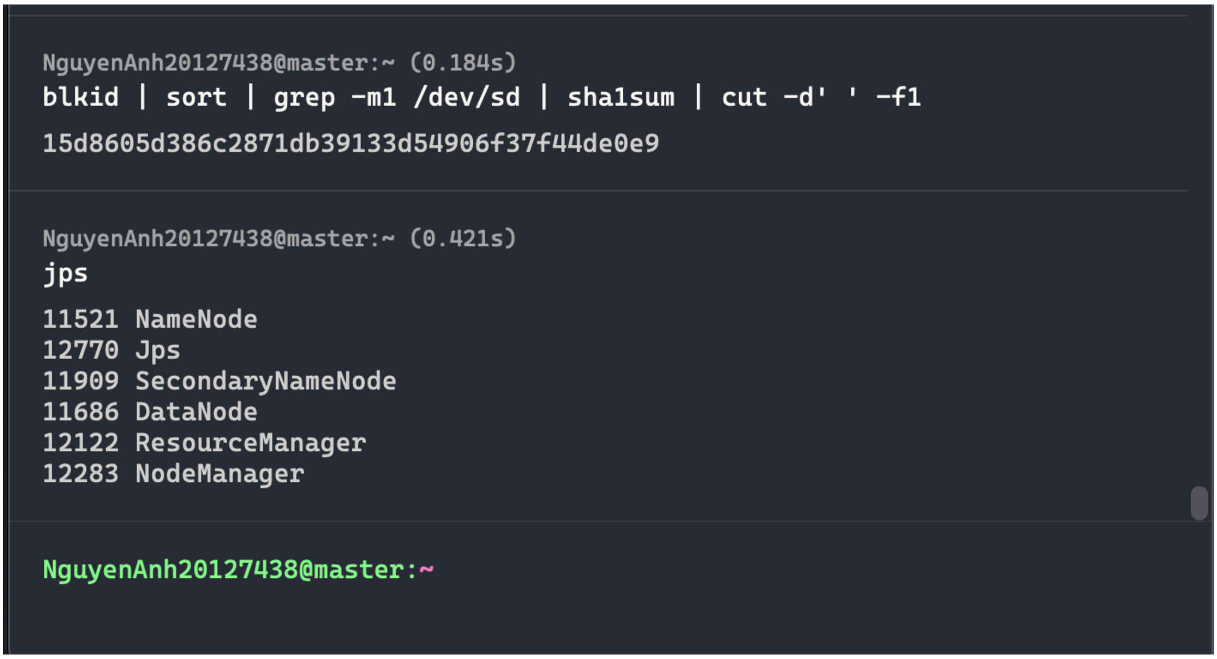
> Nguyen Anh - 20127438 Virtual Machine

- 1 Master `35.198.220.25`
  - [x] Running nodes
    - NameNode
    - SecondaryNameNode
    - DataNode
    - ResourceManager
    - NodeManager
  - [x] Ports
    - 9870

    * 9000
    * 8088
    * 8030
    * 8031
    * 8032
    * 8188
    * 50000-50100
    * 8030-8033
    * 8040
    * 8042

- 4 Worker `34.101.47.69`, `34.101.233.75`, `34.128.115.73`, `34.101.208.65`
  - [x] Running nodes
    * DataNode
    * NodeManager
  - [x] Ports
    * open all ports

Screenshot example:



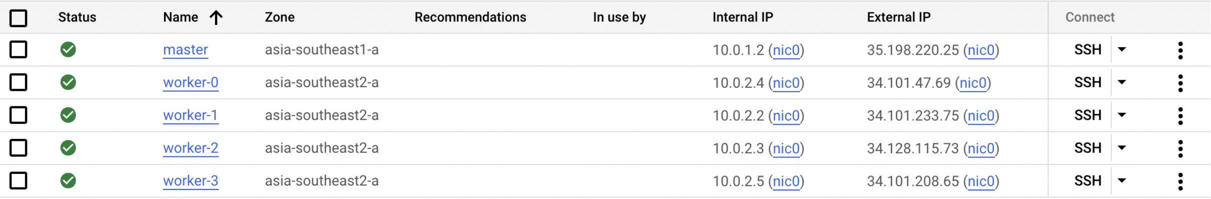
```
NguyenAnh20127438@master:~ (0.184s)
blkid | sort | grep -m1 /dev/sd | sha1sum | cut -d' ' -f1
15d8605d386c2871db39133d54906f37f44de0e9

NguyenAnh20127438@master:~ (0.421s)
jps
11521 NameNode
12770 Jps
11909 SecondaryNameNode
11686 DataNode
12122 ResourceManager
12283 NodeManager

NguyenAnh20127438@master:~
```

Figure 1.9: Proof of change your shell prompt's name

Screenshot example:



□	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect	
□	✓	master	asia-southeast1-a			10.0.1.2 (nic0)	35.198.220.25 (nic0)	SSH ▾	⋮
□	✓	worker-0	asia-southeast2-a			10.0.2.4 (nic0)	34.101.47.69 (nic0)	SSH ▾	⋮
□	✓	worker-1	asia-southeast2-a			10.0.2.2 (nic0)	34.101.233.75 (nic0)	SSH ▾	⋮
□	✓	worker-2	asia-southeast2-a			10.0.2.3 (nic0)	34.128.115.73 (nic0)	SSH ▾	⋮
□	✓	worker-3	asia-southeast2-a			10.0.2.5 (nic0)	34.101.208.65 (nic0)	SSH ▾	⋮

Figure 1.10: ImgPlaceholder

Reference examples:

Some text in which I cite an author.[1]

More text. Another citation.[2]

What is this? Yet *another* citation?[3]

1.5 References

- Three Cloudera version of WordCount problem:
  - https://docs.cloudera.com/documentation/other/tutorial/CDH5/topics-/ht_wordcount1.html
  - https://docs.cloudera.com/documentation/other/tutorial/CDH5/topics/ht_wordcount2.html
  - https://docs.cloudera.com/documentation/other/tutorial/CDH5/topics/ht_wordcount3.html
- Book: MapReduce Design Patterns [Donald Miner, Adam Shook, 2012]
- All of StackOverflow link related.
- Set up Hadoop Cluster
  - https://www.linode.com/docs/guides/how-to-install-and-set-up-hadoop-cluster/
  - https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html
- Slide of course.

[1] So Chris Krycho, “Not Exactly a Millennium,” chriskrycho.com, July 2015, http://v4.chriskrycho.com/2015/not-exactly-a-millennium.html (accessed July 25, 2015)

[2] Contra Krycho, 15, who has everything *quite* wrong.

[3] ibid