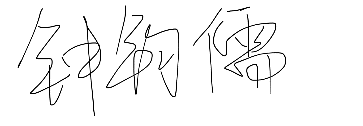
**IERG 4300 – Homework #0**

I declare that the assignment here submitted is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website <http://www.cuhk.edu.hk/policy/academichonesty/>.

**Student Name:** Junru Zhong 鍾鈞儒 **Student ID:** 1155130306 **Signature**: 

1. **Configurate a Pseudo-Distributed Hadoop Cluster**

I configurated the cluster on AWS with a t3.large instance in HK. Here is the screenshot of the HDFS web interface, which can be find on http://address.to.ec2:50070.

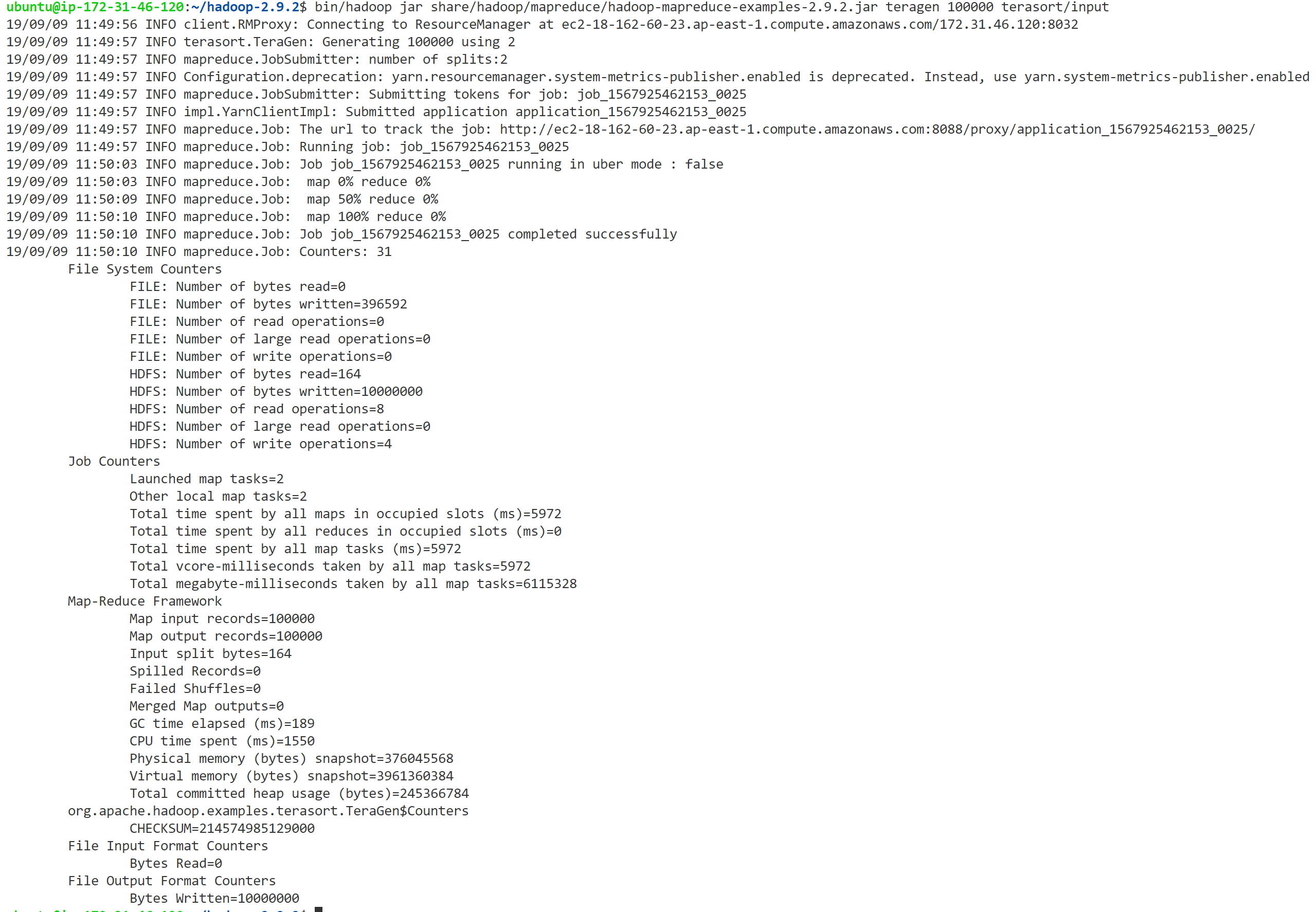
图片包含 屏幕截图

描述已自动生成

Then, I ran the example “TeraSort” programme. The programme can be run by the following command:

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar teragen 100000 terasort/input

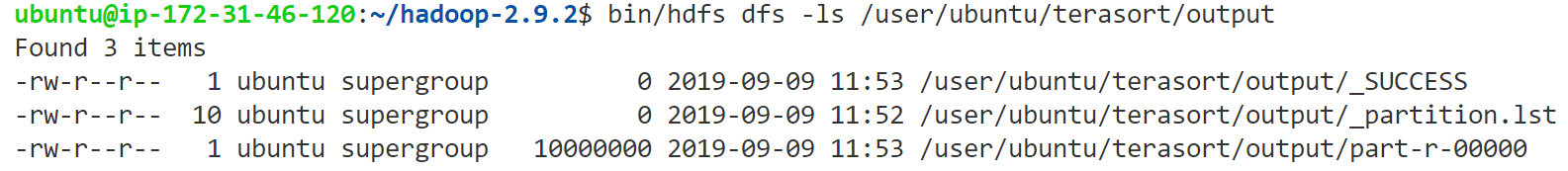
And the output of it looks like,



With no error, continue to TeraSort by the following command.

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar terasort terasort/input terasort/output

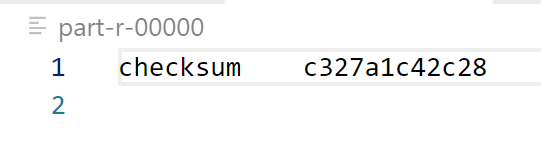
After finishing this task, the output folder was added with the output file like the following. However, the output file is binary, and I cannot open it to see the inside.



Finally, the TeraValidate can be run by the following command.

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar teravalidate terasort/output terasort/check

Then it outputted a checksum in the output file.



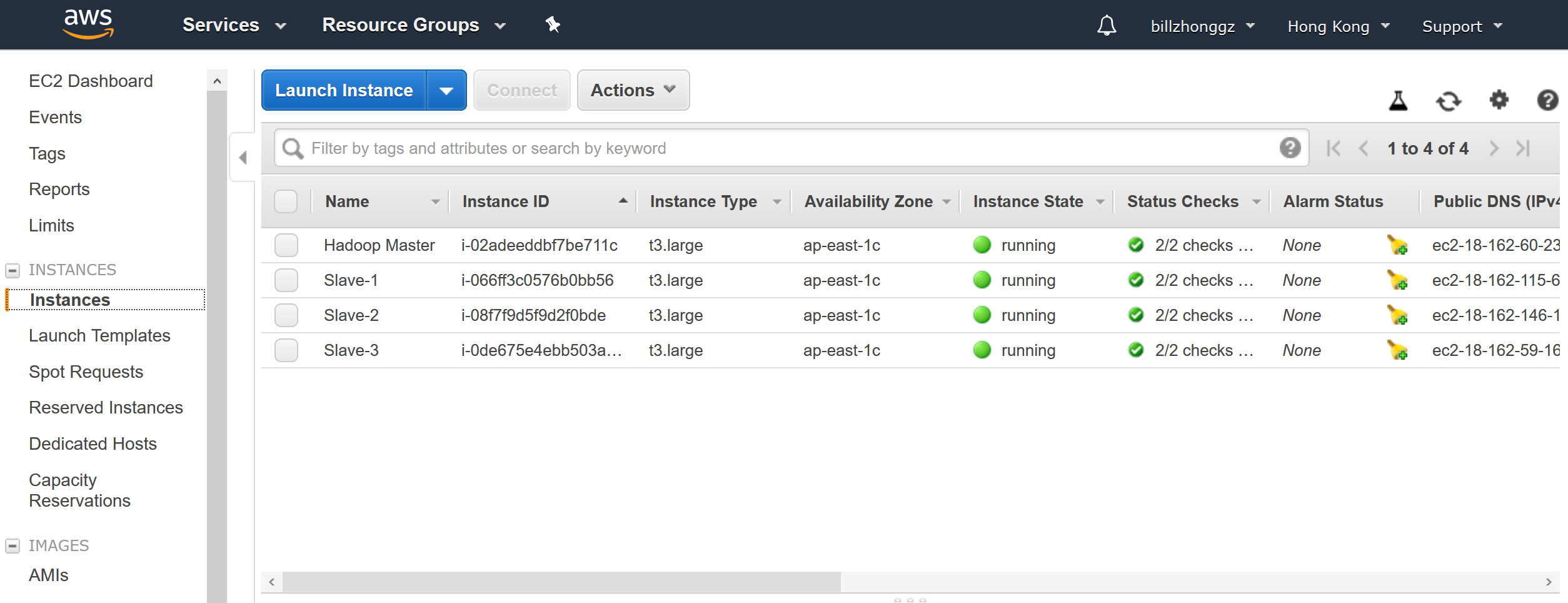
Here is the screenshot of the YARN web interface, showing three tasks on Terasort are finished.

图片包含 屏幕截图, 天空

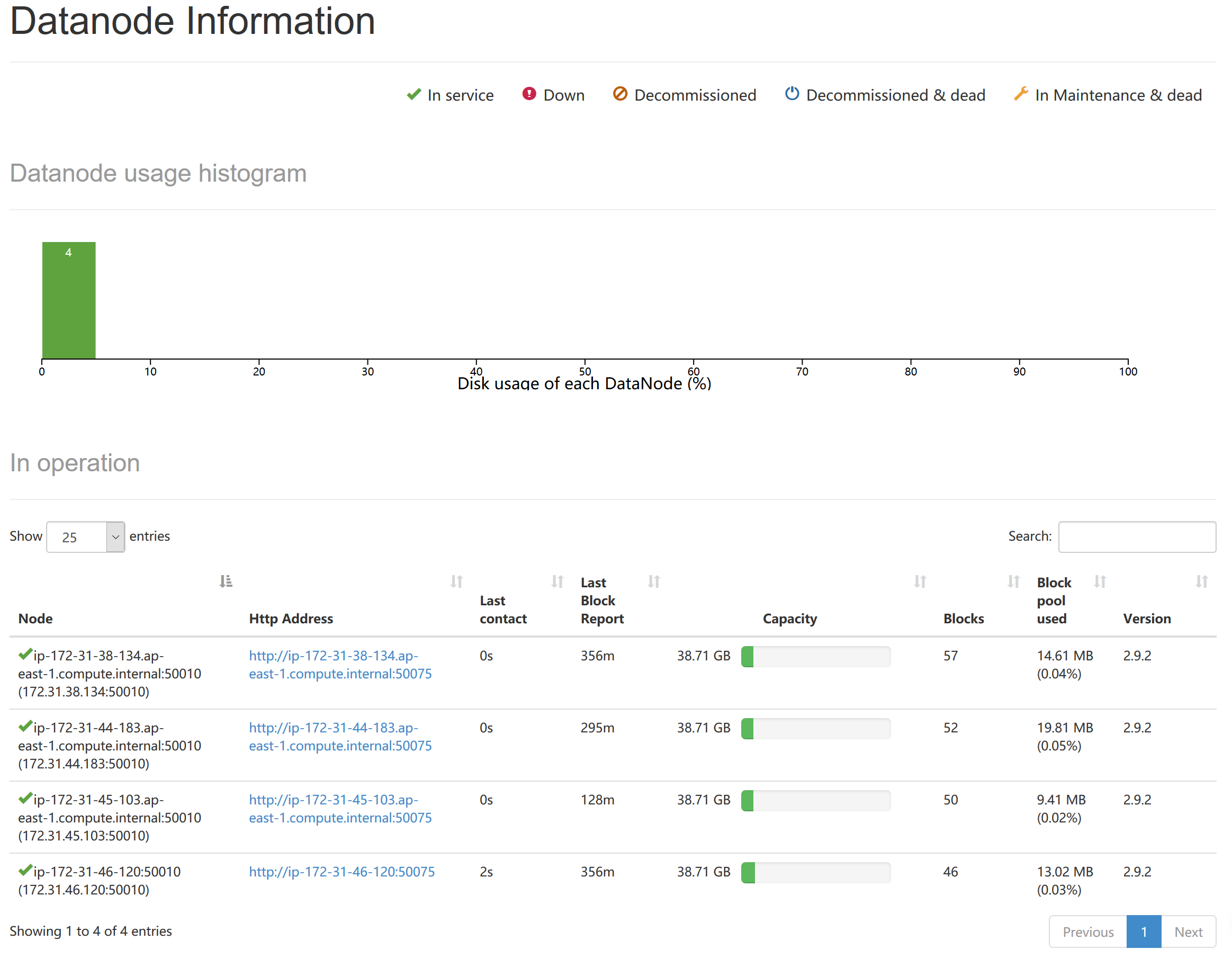
描述已自动生成

1. **Configure a Distributed Hadoop Cluster**

I configurated the cluster with 4 instances on AWS. One as the master, three as the slaves.



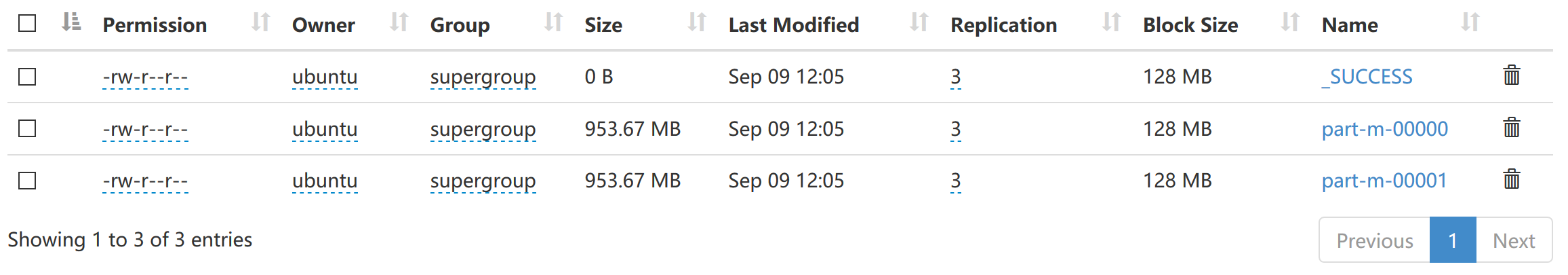
From the namenode web interface, information of 4 datanodes are shown. Due to the TeraSort 20G is extremely resource hungry, I also add the master machine into the computation and storage pool (datanode).



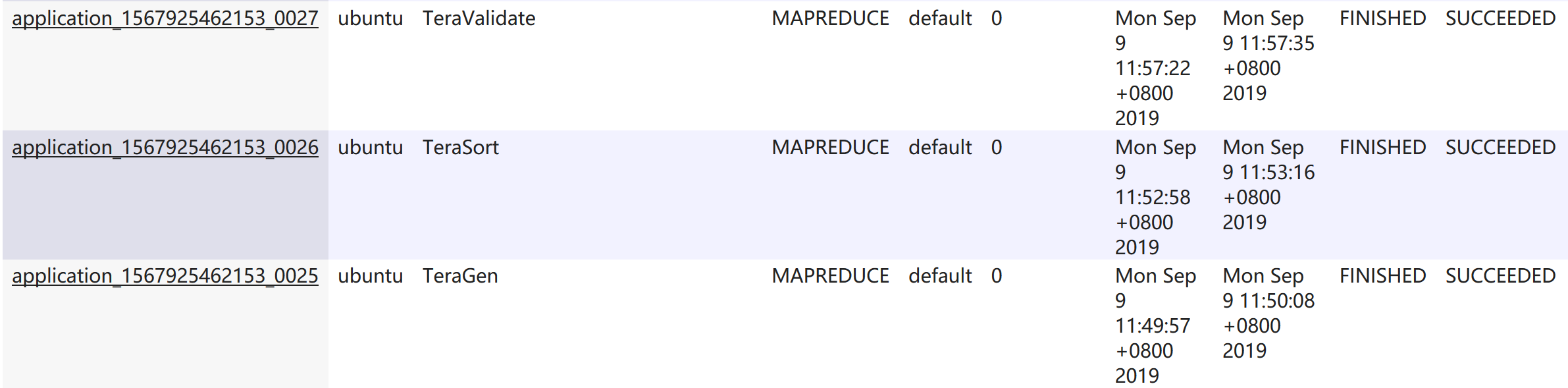
Then, I used the following command to generate 2GB of data,

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar teragen 20000000 terasort/input

Then the 2GB data is settled down in the file system.



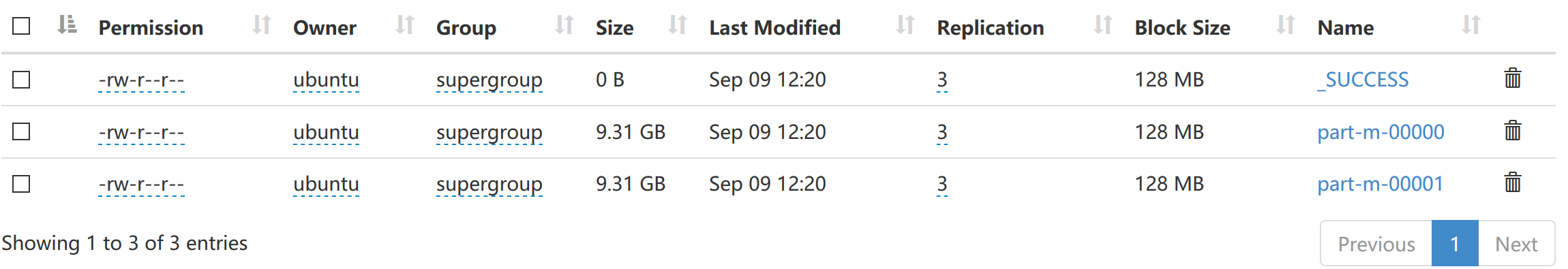
Then the actual TeraSort and TeraValidate programme can be run by this command, same as the previous ones. All of the three tasks were successful. The running time of these three tasks are shown below on Table 1.



Now move to TeraSort 20G. I generated the data by the following command,

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar teragen 200000000 terasort/input

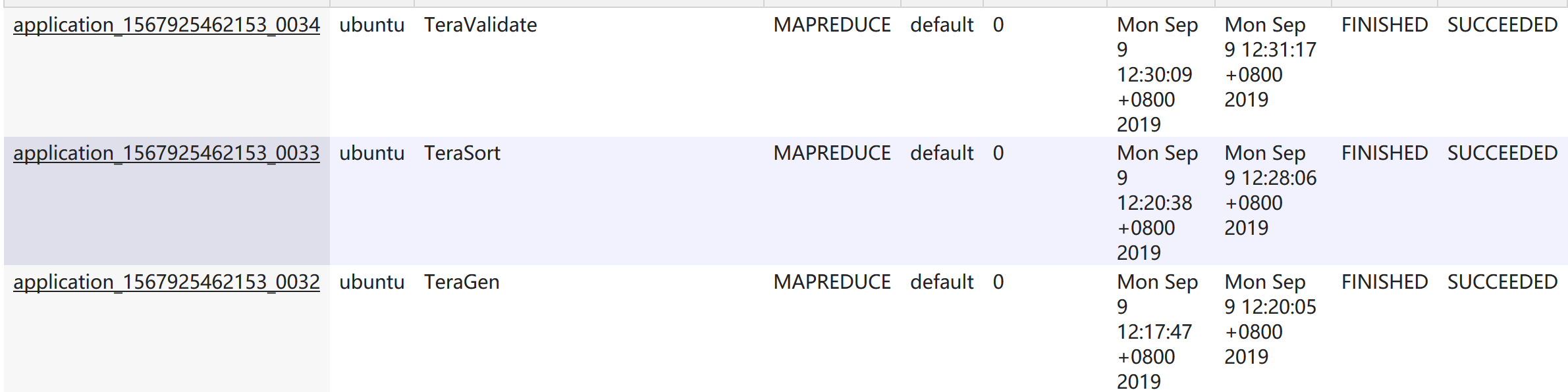
20GB of dataset was generated.



For the TeraSort 20G, I failed to run the sorting part in the default setting. By default, the TeraSort task runs only one reduce task, which be run on only one machine. However, the reduce process takes a lot of hard disk space that one machine cannot handle. So, by adding the parameter which increases the number of reduce tasks, the reduce tasks can be distributed to the entire cluster. Then it can be done by making use of all spaces on the whole cluster. The command with the parameter is like the follows, and I set 30 reduce tasks for this sorting job.

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar terasort -Dmapred.reduce.tasks=30 terasort/input terasort/output

Finally, I ran the TeraValidate task. All of the three tasks for TeraSort 20GB have finished. The running times can be seen on Table 1.



|  |  |  |
| --- | --- | --- |
| **Task Name** | **2GB dataset** | **20GB dataset** |
| TeraGen | 20 sec | 2 min, 18 sec |
| TeraSort (1 reduce task) | 1 min, 16 sec | Failed |
| TeraSort (30 reduce tasks) | 1 min, 11 sec | 7 min, 28 sec |
| TeraValidate | 24 sec | 1 min, 7 sec |

Table 1 TeraSort Running Times

1. **Run a Python Programme Through Streaming**

To run the Python word count programme, I need to download the scripts and dataset to the master machine. Then put the dataset to HDFS by this command,

# Create the folder on HDFS

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hdfs dfs -mkdir -p wordcount/input

# Put the dataset to HDFS

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hdfs dfs -put ../wordcount/large.txt wordcount/input

In my case, I am using Ubuntu 18.04 across the cluster. Since Ubuntu 18.04 does not come with a Python 2.x, I need to install it by this command across the cluster (I have tried to modify the script to Python 3, but it failed),

sudo apt install python

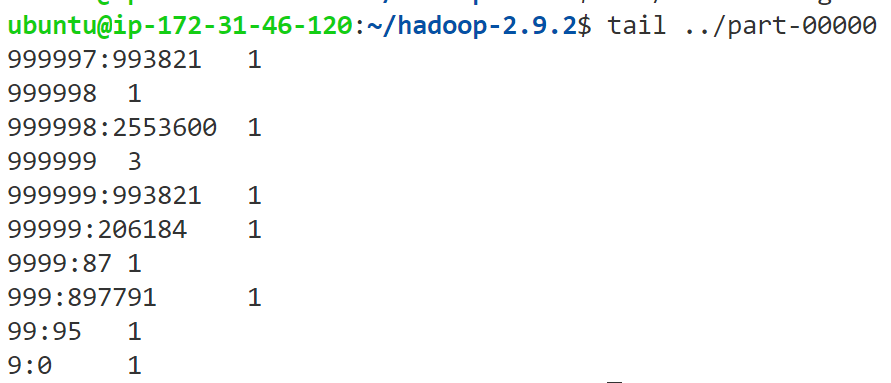
Then, the programme can be run by this command,

ubuntu@ip-172-31-46-120:~/hadoop-2.9.2$ bin/hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar -file mapper.py -mapper mapper.py -file reducer.py -reducer reducer.py -input /user/ubuntu/wordcount/input -output /user/ubuntu/wordcount/output

The result of it looks like following. And the running time can be seen on Table 2.

|  |  |
| --- | --- |
| **WordCount Tasks** | **Running Time** |
| Python Through Streaming | 3 mins, 27 sec |
| Compiled Java Source File | 1 min, 36 sec |

Table 2 WordCount Running Times



1. **Compile and Run a Java Programme**

Following the instruction on <https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>, I compiled the Java source file, packed the jar package right on the virtual machine. It can be done by these commands.

# Get the environment variables set.

ubuntu@ip-172-31-46-120:~/wordcount$ export JAVA\_HOME=/usr/java/default

ubuntu@ip-172-31-46-120:~/wordcount$ export PATH=${JAVA\_HOME}bin:${PATH}

ubuntu@ip-172-31-46-120:~/wordcount$ export HADOOP\_CLASSPATH=${JAVA\_HOME}lib/tools.jar

# Compile the Java source file.

ubuntu@ip-172-31-46-120:~/wordcount$ ../hadoop-2.9.2/bin/hadoop com.sun.tools.javac.Main WordCount.java

# Pack the jar package

ubuntu@ip-172-31-46-120:~/wordcount$ jar cf wc.jar WordCount\*.class

# Run the jar package

ubuntu@ip-172-31-46-120:~/wordcount$ ../hadoop-2.9.2/bin/hadoop jar wc.jar WordCount wordcount/input wordcount/output

The running time of this programme can be found on Table 2 on the pervious page. It can be seen that the Java programme is much faster than the Python one.

­­­**----- End of this submission. Last modified on Sept 9 -----**