CSC 550 Big Data
Signature Assignment

93842	**Xiaofang Yu**
94090	**Qian Cheng**
94053	**Jialian Zhang**
89722	**Xinxin Wang**
94270	**Shuyuan Chang**

Content

Overview

Hadoop is an open source software project. Large datasets can be efficiently processed using Hadoop. Instead of using one large computer to process and store the data, Hadoop analyze massive data sets in parallel. It allows clustering commodity hardware together.

MapReduce decomposes large manipulation jobs into small tasks. These tasks can be executed in parallel cross a cluster of servers. Then we can put the results of tasks together to compute final results.

By implementing the sorting method of a 10G random data, we can better understand the working of the MapReduce programming paradigm. This project contains two parts, which are map and reduce. The Map task maps the data in the file and sort in a specific order. The outcome of this task is passed to reduce task which combines and reduces the data to output the final result.

Purpose

The purpose of this project is to enhancing student learning outcome by doing MapReduce sorting with Hadoop. HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers.
The project consists the following steps:

- Install Virtualbox and Hadoop
- Programm TeraSort
- 3 phases of TeraSort: Mapper, Shuffle, Reducer
- Compile and make an executable file
- Execute and record performance
- Summarize

3-steps of TeraSort

- Data Generation

Use teraGen to generate a random 10G file and put the test file in Hadoop file system.

- TeraSort
  Use mapreduce to sort the content shows in the test file. At the meantime, track the CPU usage as well as the disk IO for performance analysis.

- Result Analysis
  Validate the generated result to see whether the process sorts the file correctly. Also, output the disk IO and CPU usage data to analyze the performance.

Configuration

We test and run this mapreduce project in the laptop with the powerful 2.7 GHz Intel Core i7 CPU and 16 GB 2133 MHz LPDDR3. The system configuration of this project is Ubuntu 14.04 with parallels(hypervisor) based on Mac High Sierra 10.13.4.

Configuration file of Hadoop

Configuration of Hadoop needs to be done right after installing of Hadoop.
To configure Hadoop, we need to edit the following files:
1.core-site.xml
The file contains information such as port number, memory limit and size of buffers.
2.hdfs-site.xml
The file contains information such as the value of replication data, namenode path, and datanode paths of your local file systems.
3.yarn-site.xml
This file is used to configure yarn into Hadoop.
4.mapred-site.xml

Mapper and Reducer

This file is used to specify which MapReduce framework we are using.

We implemented the Mapper and Reducer interfaces to provide the map and reduce methods. These methods play important roles to complete the job.

Mapper maps input key/value pairs to a set of intermediate key/value pairs. The Hadoop MapReduce framework produces one map task for each InputSplit generated b the InputFormat for the job. The number of maps is decided by the total size of the inputs, which equals to the total numbers of blocks of the input files. The right number of parallelism for maps works best to be around 10-100 maps per node.

Reducer reduces a set of intermediate values which share a key to a smaller set of values. Reducer has three major phases:
1. Shuffle: in this phase, the framework gets the relevant partition of the output of all the mappers via HTTP.
2. Sort: the framework groups Reducer inputs by keys. This stage occurs simultaneously with the shuffle phase.
3. Reduce: the reduce (WritableComparable, Iterator, OutputCollector, Reporter) method is called for each <key, (list of values)> pair in the grouped inputs in this phase.
The proper number of reduces is 0.95 or 1.75 multiplied by (<no. of nodes> *mapred.tasktracker.reduce.tasks.maximum). With 0.95, all the reduces can launch right away and start transferring map outputs as the maps finish. With 1.75, the increasing number of reduces increase the framework overload but at the same time, it increases load balancing and lowers the cost of failures.

TeraSort Process

In this section, we will demonstrate the process of our TeraSorting in detail.

Step 1 Data Generation

- We first use the following command to generate a random 10G file:

hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraGen 10000000000 /teraInput

- The below screenshot shows that the 10G file has been generated successfully:

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/kms/tomcat/webapps/kms/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/httpfs/tomcat/webapps/webhdfs/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/07/30 17:45:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/07/30 17:45:46 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
18/07/30 17:45:46 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
18/07/30 17:45:46 INFO terasort.TeraSort: Generating 10000000000 using 1
18/07/30 17:45:46 INFO mapreduce.JobSubmitter: number of splits:1
18/07/30 17:45:46 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1721036068_0001
18/07/30 17:45:47 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
18/07/30 17:45:47 INFO mapreduce.Job: Running job: job_local1721036068_0001
18/07/30 17:45:47 INFO mapred.LocalJobRunner: OutputCommitter set in config null
18/07/30 17:45:47 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/07/30 17:45:47 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
18/07/30 17:45:47 INFO mapred.LocalJobRunner: Waiting for map tasks
18/07/30 17:45:47 INFO mapred.LocalJobRunner: Starting task: attempt_local1721036068_0001_m_000000_0
18/07/30 17:45:47 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/07/30 17:45:47 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
18/07/30 17:45:47 INFO mapred.MapTask: Processing split: org.apache.hadoop.examples.terasort.TeraGen$RangeInputFormat$RangeInputSplit@8ee522b
18/07/30 17:45:48 INFO mapreduce.Job: Job job_local1721036068_0001 running in uber mode : false
18/07/30 17:45:48 INFO mapreduce.Job:  map 0% reduce 0%
18/07/30 17:45:53 INFO mapred.LocalJobRunner: map > map
18/07/30 17:45:56 INFO mapred.LocalJobRunner: map > map
18/07/30 17:45:59 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:02 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:05 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:08 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:11 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:14 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:17 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:20 INFO mapred.LocalJobRunner: map > map
18/07/30 17:46:23 INFO mapred.LocalJobRunner: map > map
```

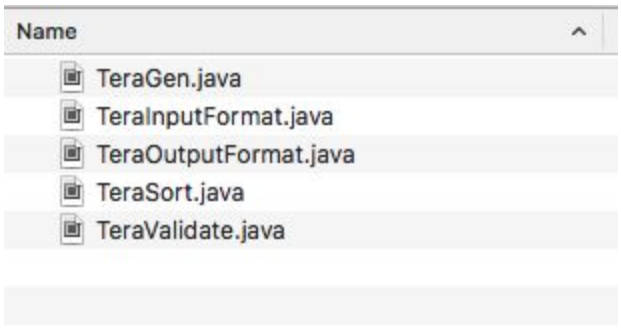
- For further processing, we now need to put the testfile into the hadoop file system. By running the hdfs dfs -put command, the 10G testfile is ready to process.

```
parallels@ubuntu:/usr/local/hadoop/WordCount$ hdfs dfs -put testfile/10Gtest.txt /user/hadoopuser/input/final_proj
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/kms/tomcat/webapps/kms/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/httpfs/tomcat/webapps/webhdfs/WEB-INF/lib/slf4j-log4j12-1.7.10.j
ar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/06/28 20:04:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classe
s where applicable
```

Step 2 TeraSort

The terasort program is designed to sort the big data in order. We use TimeUnit.SECONDS.sleep(20) to pause the process, in order to monitor performance wave.

Blow is the list of the source code files. For further detail, please check out the attached files to view the source code:



- **Mapreduce Execution Command:**

We run the following command to run the java program we wrote before to execute the mapreduce using the file located in the input folder.

hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraSort /teraInput /teraInput/out

```
parallels@ubuntu:/usr/local/hadoop/terasort$ hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraSort /teraInput2 /teraInput2/out
18/07/30 17:54:15 INFO terasort.TeraSort: starting
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/kms/tomcat/webapps/kms/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/httpfs/tomcat/webapps/webhdfs/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/07/30 17:54:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/07/30 17:54:16 INFO input.FileInputFormat: Total input paths to process : 1
Spent 122ms computing base-splits.
Spent 2ms computing TeraScheduler splits.
Computing input splits took 125ms
Sampling 1 splits of 1
Making 1 from 100000 sampled records
Computing partitions took 463ms
Spent 591ms computing partitions.
18/07/30 17:54:17 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
18/07/30 17:54:17 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
18/07/30 17:54:17 INFO mapreduce.JobSubmitter: number of splits:1
18/07/30 17:54:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local195843222_0001
18/07/30 17:54:18 INFO mapred.LocalDistributedCacheManager: Creating symlink: /usr/local/hadoop/temp/mapred/local/1532998457881/_partition.lst <- /usr/local/hadoop/terasort/_partition.lst
18/07/30 17:54:18 INFO mapred.LocalDistributedCacheManager: Localized hdfs://localhost:9000/teraInput2/out/_partition.lst as file:/usr/local/hadoop/temp/mapred/local/1532998457881/_partition.lst
18/07/30 17:54:18 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
18/07/30 17:54:18 INFO mapreduce.Job: Running job: job_local195843222_0001
18/07/30 17:54:18 INFO mapred.LocalJobRunner: OutputCommitter set in config null
18/07/30 17:54:18 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/07/30 17:54:18 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
18/07/30 17:54:18 INFO mapred.LocalJobRunner: Waiting for map tasks
18/07/30 17:54:18 INFO mapred.LocalJobRunner: Starting task: attempt_local195843222_0001_m_000000_0
18/07/30 17:54:18 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/07/30 17:54:18 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
18/07/30 17:54:18 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/teraInput2/part-m-00000:0+100000000
18/07/30 17:54:18 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
18/07/30 17:54:18 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
18/07/30 17:54:18 INFO mapred.MapTask: soft limit at 83886080
18/07/30 17:54:18 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
18/07/30 17:54:18 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
18/07/30 17:54:18 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
18/07/30 17:54:19 INFO mapreduce.Job: Job job_local195843222_0001 running in uber mode : false
18/07/30 17:54:19 INFO mapreduce.Job:  map 0% reduce 0%
```

- **Find TeraSort PID:**

When running the TeraSort, we use the following command to find the process ID of the TeraSort and use it to track the disk IO and CPU usage for later performance analysis.

```
parallels@ubuntu:/usr/local/hadoop/sort$ ps aux | grep Tera*
parall+  4561  167 12.3 1069112 125588 pts/3  Sl+  00:11   0:05 /usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/logs -Dhadoop.l
og.file=hadoop.log -Dhadoop.home.dir=/usr/local/hadoop -Dhadoop.id.str=parallels -Dhadoop.root.logger=INFO,console -Dhadoop.policy.file=hadoop-policy.xml -Djava.net.preferIPv4Stack=true -Xmx512m -Dhadoop.s
ecurity.logger=INFO,NullAppender org.apache.hadoop.util.RunJar TeraSort.jar org.apache.hadoop.examples.terasort.TeraGen 10000000 /teraInput3
```

Step 3 Result Analysis

As the following screenshot shows, our mapreduce job has done successfully:

```
18/06/28 20:11:52 INFO mapred.LocalJobRunner: Finishing task: attempt_local936611078_0001_r_000000_0
18/06/28 20:11:52 INFO mapred.LocalJobRunner: reduce task executor complete.
18/06/28 20:11:52 INFO mapreduce.Job:  map 100% reduce 100%
18/06/28 20:11:52 INFO mapreduce.Job: Job job_local936611078_0001 completed successfully
18/06/28 20:11:52 INFO mapreduce.Job: Counters: 35
```

Given the length of the whole results, the complete result doesn't show here. We did the validation:

hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraValidate /teraInput2 /teraInput2/val

```
parallels@ubuntu:/usr/local/hadoop/terasort$ hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraValidate /teraInput2 /teraInput2/val
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/kms/tomcat/webapps/kms/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/httpfs/tomcat/webapps/webhdfs/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/07/30 18:01:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/07/30 18:01:31 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
18/07/30 18:01:31 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
18/07/30 18:01:31 INFO input.FileInputFormat: Total input paths to process : 2
Spent 72ms computing base-splits.
Spent 2ms computing TeraScheduler splits.
18/07/30 18:01:31 INFO mapreduce.JobSubmitter: number of splits:2
18/07/30 18:01:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1478547831_0001
18/07/30 18:01:31 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
18/07/30 18:01:31 INFO mapreduce.Job: Running job: job_local1478547831_0001
18/07/30 18:01:31 INFO mapred.LocalJobRunner: OutputCommitter set in config null
18/07/30 18:01:31 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/07/30 18:01:31 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
18/07/30 18:01:31 INFO mapred.LocalJobRunner: Waiting for map tasks
18/07/30 18:01:31 INFO mapred.LocalJobRunner: Starting task: attempt_local1478547831_0001_m_000000_0
18/07/30 18:01:31 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
18/07/30 18:01:31 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
18/07/30 18:01:31 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/teraInput2/part-m-00000:0+100000000
18/07/30 18:01:32 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
18/07/30 18:01:32 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
18/07/30 18:01:32 INFO mapred.MapTask: soft limit at 83886080
18/07/30 18:01:32 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
18/07/30 18:01:32 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
18/07/30 18:01:32 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
18/07/30 18:01:32 INFO mapreduce.Job: Job job_local1478547831_0001 running in uber mode : false
18/07/30 18:01:32 INFO mapreduce.Job:  map 0% reduce 0%
18/07/30 18:01:34 INFO mapred.LocalJobRunner:
18/07/30 18:01:34 INFO mapred.MapTask: Starting flush of map output
```

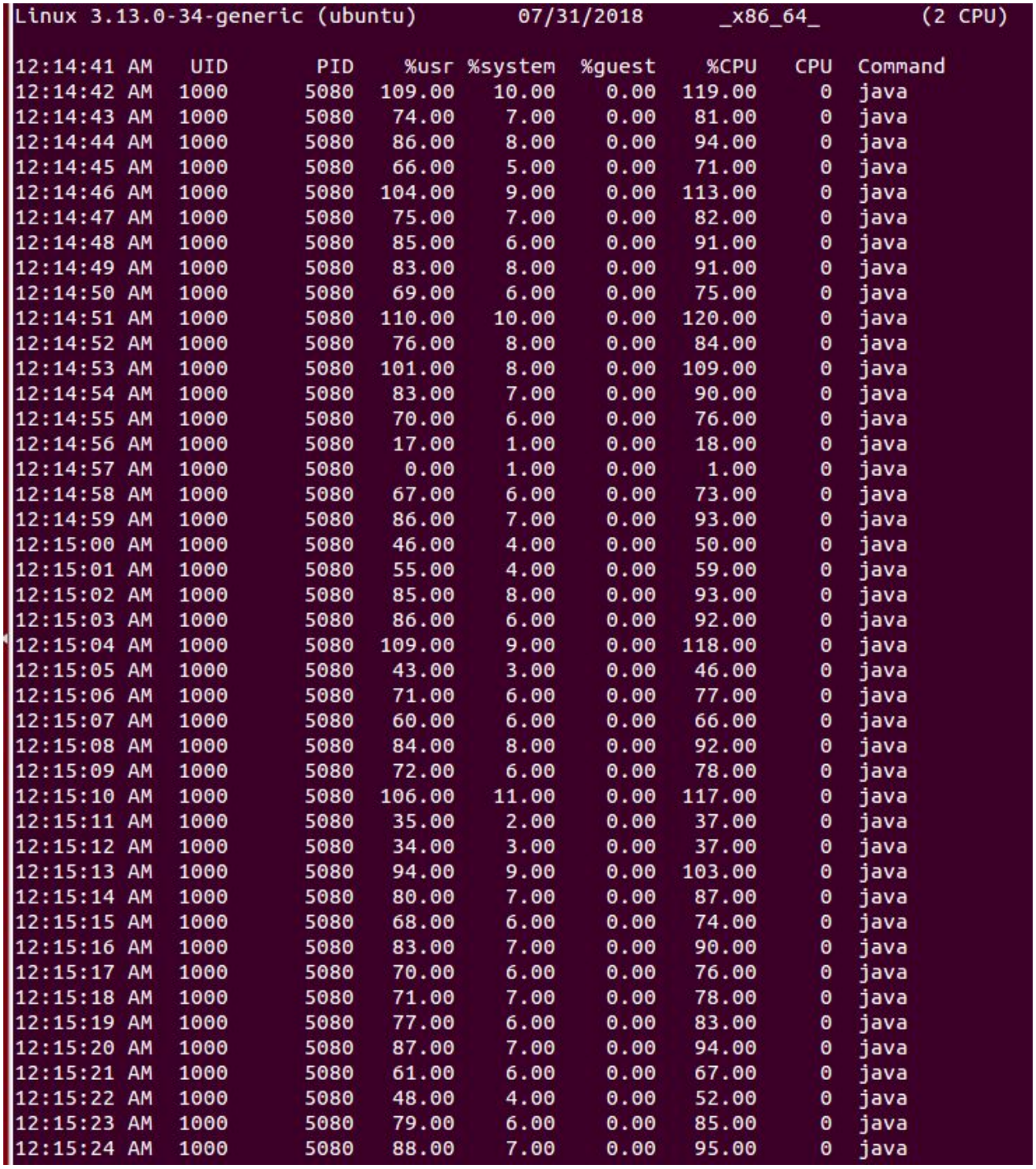
Performance

By tracking the task's process ID, we output the CPU usage information into a text file named "cpuUsage.txt" by using the following command:

Pidstat 1 -p 5080 >> cpuUsage.txt

CPU usage result:
The result of CPU usage for running the TeraSort program has shown as below.
The screenshot is the result stored in the cupUsage.txt file.

Screenshot of part of the CPU usage result:



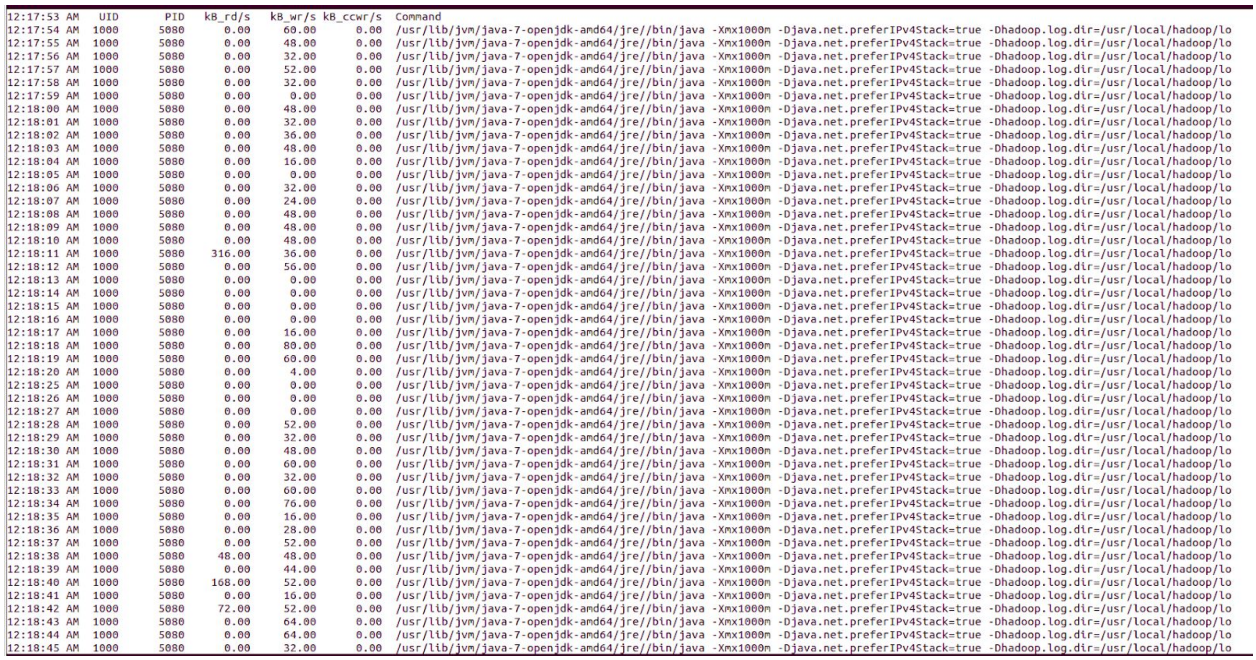
```
Linux 3.13.0-34-generic (ubuntu)        07/31/2018      _x86_64_        (2 CPU)

12:14:41 AM   UID       PID    %usr %system  %guest    %CPU   CPU  Command
12:14:42 AM  1000      5080  109.00   10.00    0.00  119.00     0  java
12:14:43 AM  1000      5080   74.00    7.00    0.00   81.00     0  java
12:14:44 AM  1000      5080   86.00    8.00    0.00   94.00     0  java
12:14:45 AM  1000      5080   66.00    5.00    0.00   71.00     0  java
12:14:46 AM  1000      5080  104.00    9.00    0.00  113.00     0  java
12:14:47 AM  1000      5080   75.00    7.00    0.00   82.00     0  java
12:14:48 AM  1000      5080   85.00    6.00    0.00   91.00     0  java
12:14:49 AM  1000      5080   83.00    8.00    0.00   91.00     0  java
12:14:50 AM  1000      5080   69.00    6.00    0.00   75.00     0  java
12:14:51 AM  1000      5080  110.00   10.00    0.00  120.00     0  java
12:14:52 AM  1000      5080   76.00    8.00    0.00   84.00     0  java
12:14:53 AM  1000      5080  101.00    8.00    0.00  109.00     0  java
12:14:54 AM  1000      5080   83.00    7.00    0.00   90.00     0  java
12:14:55 AM  1000      5080   70.00    6.00    0.00   76.00     0  java
12:14:56 AM  1000      5080   17.00    1.00    0.00   18.00     0  java
12:14:57 AM  1000      5080    0.00    1.00    0.00    1.00     0  java
12:14:58 AM  1000      5080   67.00    6.00    0.00   73.00     0  java
12:14:59 AM  1000      5080   86.00    7.00    0.00   93.00     0  java
12:15:00 AM  1000      5080   46.00    4.00    0.00   50.00     0  java
12:15:01 AM  1000      5080   55.00    4.00    0.00   59.00     0  java
12:15:02 AM  1000      5080   85.00    8.00    0.00   93.00     0  java
12:15:03 AM  1000      5080   86.00    6.00    0.00   92.00     0  java
12:15:04 AM  1000      5080  109.00    9.00    0.00  118.00     0  java
12:15:05 AM  1000      5080   43.00    3.00    0.00   46.00     0  java
12:15:06 AM  1000      5080   71.00    6.00    0.00   77.00     0  java
12:15:07 AM  1000      5080   60.00    6.00    0.00   66.00     0  java
12:15:08 AM  1000      5080   84.00    8.00    0.00   92.00     0  java
12:15:09 AM  1000      5080   72.00    6.00    0.00   78.00     0  java
12:15:10 AM  1000      5080  106.00   11.00    0.00  117.00     0  java
12:15:11 AM  1000      5080   35.00    2.00    0.00   37.00     0  java
12:15:12 AM  1000      5080   34.00    3.00    0.00   37.00     0  java
12:15:13 AM  1000      5080   94.00    9.00    0.00  103.00     0  java
12:15:14 AM  1000      5080   80.00    7.00    0.00   87.00     0  java
12:15:15 AM  1000      5080   68.00    6.00    0.00   74.00     0  java
12:15:16 AM  1000      5080   83.00    7.00    0.00   90.00     0  java
12:15:17 AM  1000      5080   70.00    6.00    0.00   76.00     0  java
12:15:18 AM  1000      5080   71.00    7.00    0.00   78.00     0  java
12:15:19 AM  1000      5080   77.00    6.00    0.00   83.00     0  java
12:15:20 AM  1000      5080   87.00    7.00    0.00   94.00     0  java
12:15:21 AM  1000      5080   61.00    6.00    0.00   67.00     0  java
12:15:22 AM  1000      5080   48.00    4.00    0.00   52.00     0  java
12:15:23 AM  1000      5080   79.00    6.00    0.00   85.00     0  java
12:15:24 AM  1000      5080   88.00    7.00    0.00   95.00     0  java
```

We also tracked the Disk IO and output the data into a text file named “diskIO.txt” by using command:

Pidstat -dl 1 -p 5080 >> diskIO.txt

Screenshot of part of the Disk IO result:



12:17:53 AM	UID	PID	kB_rd/s	kB_wr/s	kB_ccwr/s	Command
12:17:54 AM	1000	5080	0.00	60.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:17:55 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:17:56 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:17:57 AM	1000	5080	0.00	52.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:17:58 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:17:59 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:00 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:01 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:02 AM	1000	5080	0.00	36.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:03 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:04 AM	1000	5080	0.00	16.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:05 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:06 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:07 AM	1000	5080	0.00	24.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:08 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:09 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:10 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:11 AM	1000	5080	316.00	36.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:12 AM	1000	5080	0.00	56.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:13 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:14 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:15 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:16 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:17 AM	1000	5080	0.00	16.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:18 AM	1000	5080	0.00	80.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:19 AM	1000	5080	0.00	60.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:20 AM	1000	5080	0.00	4.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:25 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:26 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:27 AM	1000	5080	0.00	0.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:28 AM	1000	5080	0.00	52.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:29 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:30 AM	1000	5080	0.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:31 AM	1000	5080	0.00	60.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:32 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:33 AM	1000	5080	0.00	60.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:34 AM	1000	5080	0.00	76.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:35 AM	1000	5080	0.00	16.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:36 AM	1000	5080	0.00	28.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:37 AM	1000	5080	0.00	52.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:38 AM	1000	5080	48.00	48.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:39 AM	1000	5080	0.00	44.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:40 AM	1000	5080	168.00	52.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:41 AM	1000	5080	0.00	16.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:42 AM	1000	5080	72.00	52.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:43 AM	1000	5080	0.00	64.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:44 AM	1000	5080	0.00	64.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo
12:18:45 AM	1000	5080	0.00	32.00	0.00	/usr/lib/jvm/java-7-openjdk-amd64/jre//bin/java -Xmx1000m -Djava.net.preferIPv4Stack=true -Dhadoop.log.dir=/usr/local/hadoop/lo

Process execution time: We use Time command to monitor.

time hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraGen 10000000000 /teraInput2

About 4 hours on our 4-node cluster

time hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraSort /teraInput2 /teraInput2/out

About 5 hours on our 4-node cluster

time hadoop jar TeraSort.jar org.apache.hadoop.examples.terasort.TeraValidate /teraInput2 /teraInput2/val

If something went wrong, TeraValidate's output contains the problem report.

Summary

In this hands-on study, we used Hadoop to build a TeraSort application to sort the 10G big data. The MapReduce programming processes were able to be executed without failures and errors. The TeraSort results and system performances were also successfully recorded. The results of TeraSort passed the validation and we have tracked the system performance throughout the project.

Through this project, we have gained significantly more understanding, knowledge, and experience on Hadoop system. We are now able to program and implement examples of big data applications using open source Hadoop, HDFS, MapReduce, etc. We have also improved our problem solving skills and techniques on re-configure software components in virtual machine environment in big data analytics problems.

Reference

https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/apache-hadoop-terasort-example/

https://www.sas.com/en_us/insights/big-data/hadoop.html

https://www.dezyre.com/hadoop-tutorial/hadoop-mapreduce-terasort-tutorial

https://hortonworks.com/apache/hdfs/

https://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm

https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

https://ems.itu.edu/student/sections/5714/syllabus