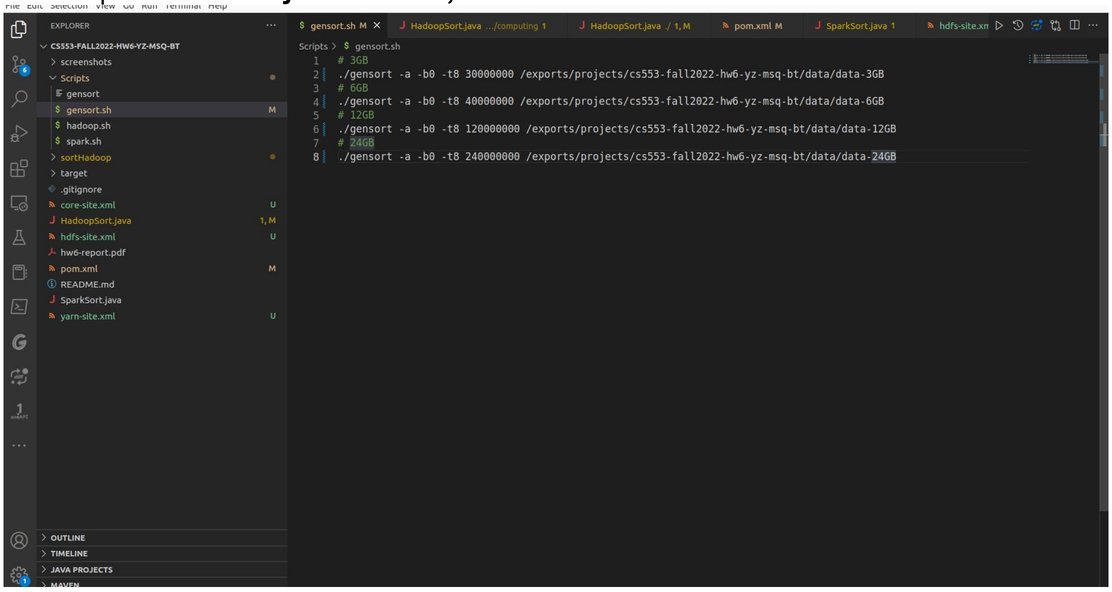
hw6-report

Devops:

1. Remote to Chameleon Physical Machine
   a) ssh -i .ssh/chameleon cc@64.131.114.58
2. To configure Container, first configure namenode (4-cores, 8GB ram, 20GB disk), then select each of the following
   a) largenode(16-cores, 32GB ram, 180GB disk)
   b) smallnode(4-cores, 8GB ram, 45GB disk)
   c) datanode1 + datanode2 + datanode3 + datanode4 (each for 4-cores, 8GB ram, 45GB disk)

   For example: create smallnode
   lxc launch images:ubuntu/22.04 smallnode --storage storage-lvm --config limits.cpu=4 --config limits.memory=8GB
   lxc config device set smallnode1 root size=45GB

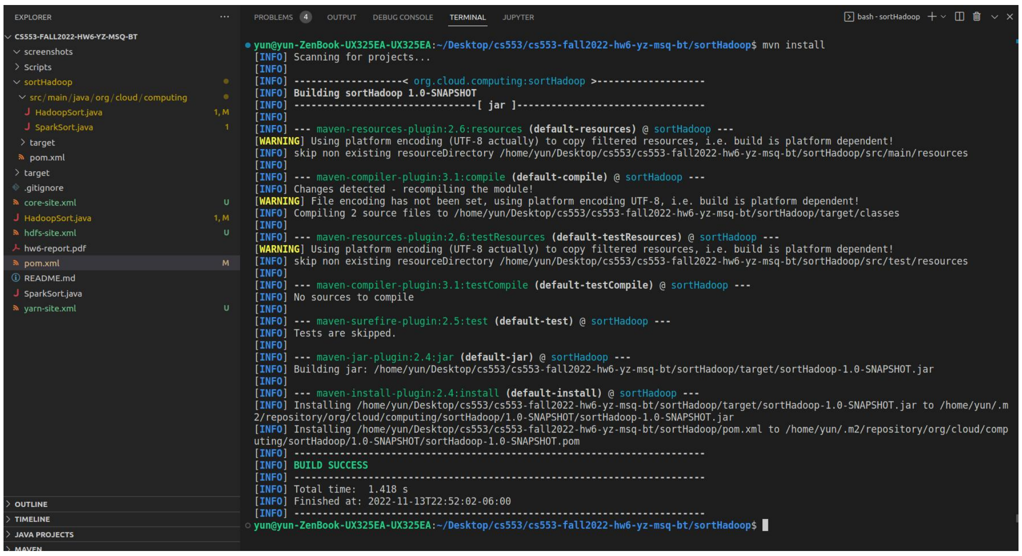
3. Make sure the **namenode** can login to **itself and all datanodes via ssh**
   ssh ubuntu@10.151.244.200 => largenode
   ssh ubuntu@10.151.244.114 => namenode
4. Create a new **/exports/projects** folder on the physical machine, all contents of this folder are shared to all machines
5. install spark, hadoop, java to shared directory. configure environment variables to **~/.bashrc**
6. Tune parameters for **yarn-site.xml, core-site.xml and hdfs-site.xml**



```
# 3GB
./gensort -a -b0 -t8 30000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-3GB
# 6GB
./gensort -a -b0 -t8 40000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-6GB
# 12GB
./gensort -a -b0 -t8 120000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-12GB
# 24GB
./gensort -a -b0 -t8 240000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-24GB
```

Project.

1. Use maven to generate the pom file, giving all the jar packages that this project depends on, for the complete project see the **sortHadoop/** directory



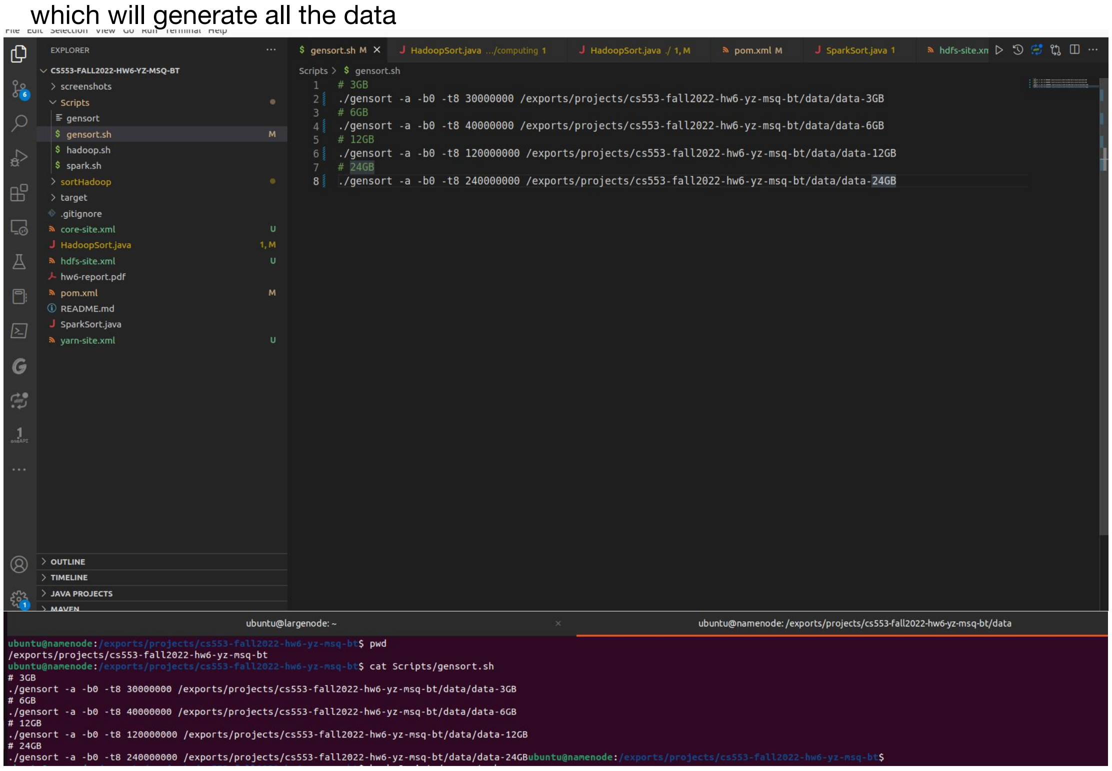
```
yun@yun-ZenBook-UX325EA-UX325EA:~/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.cloud.computing:sortHadoop >-------------------
[INFO] Building sortHadoop 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ sortHadoop ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/yun/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ sortHadoop ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 2 source files to /home/yun/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ sortHadoop ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/yun/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ sortHadoop ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.5:test (default-test) @ sortHadoop ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ sortHadoop ---
[INFO] Building jar: /home/yun/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop/target/sortHadoop-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ sortHadoop ---
[INFO] Installing /home/yun/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop/target/sortHadoop-1.0-SNAPSHOT.jar to /home/yun/.m
2/repository/org/cloud/computing/sortHadoop/1.0-SNAPSHOT/sortHadoop-1.0-SNAPSHOT.jar
[INFO] Installing /home/yun/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop/pom.xml to /home/yun/.m2/repository/org/cloud/comp
uting/sortHadoop/1.0-SNAPSHOT/sortHadoop-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.418 s
[INFO] Finished at: 2022-11-13T22:52:02-06:00
[INFO] ------------------------------------------------------------------------
yun@yun-ZenBook-UX325EA-UX325EA:~/Desktop/cs553/cs553-fall2022-hw6-yz-msq-bt/sortHadoop$
```

2. Push our project code to git, and then clone the current project down on the Chameleon physical machine (here we have to configure the ssh key to github in advance) in **/exports/projects**.

3. Switch to the directory and execute:

    cd /exports/projects/cs553-fall2022-hw6-yz-msq-bt/Scripts && bash gensort.sh

   which will generate all the data



```
# 3GB
./gensort -a -b0 -t8 30000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-3GB
# 6GB
./gensort -a -b0 -t8 40000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-6GB
# 12GB
./gensort -a -b0 -t8 120000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-12GB
# 24GB
./gensort -a -b0 -t8 240000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-24GB
```

```
ubuntu@namenode:/exports/projects/cs553-fall2022-hw6-yz-msq-bt$ pwd
/exports/projects/cs553-fall2022-hw6-yz-msq-bt
ubuntu@namenode:/exports/projects/cs553-fall2022-hw6-yz-msq-bt$ cat Scripts/gensort.sh
# 3GB
./gensort -a -b0 -t8 30000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-3GB
# 6GB
./gensort -a -b0 -t8 40000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-6GB
# 12GB
./gensort -a -b0 -t8 120000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-12GB
# 24GB
./gensort -a -b0 -t8 240000000 /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-24GBubuntu@namenode:/exports/projects/cs553-fall2022-hw6-yz-msq-bt$
```

4. linux sort, we have to notice the usage of memory. We have to use **-S** flag to limited the memory.

    like: **time sort ./data-24GB -o ./sort-24GB --parallel=16 -S 55%**

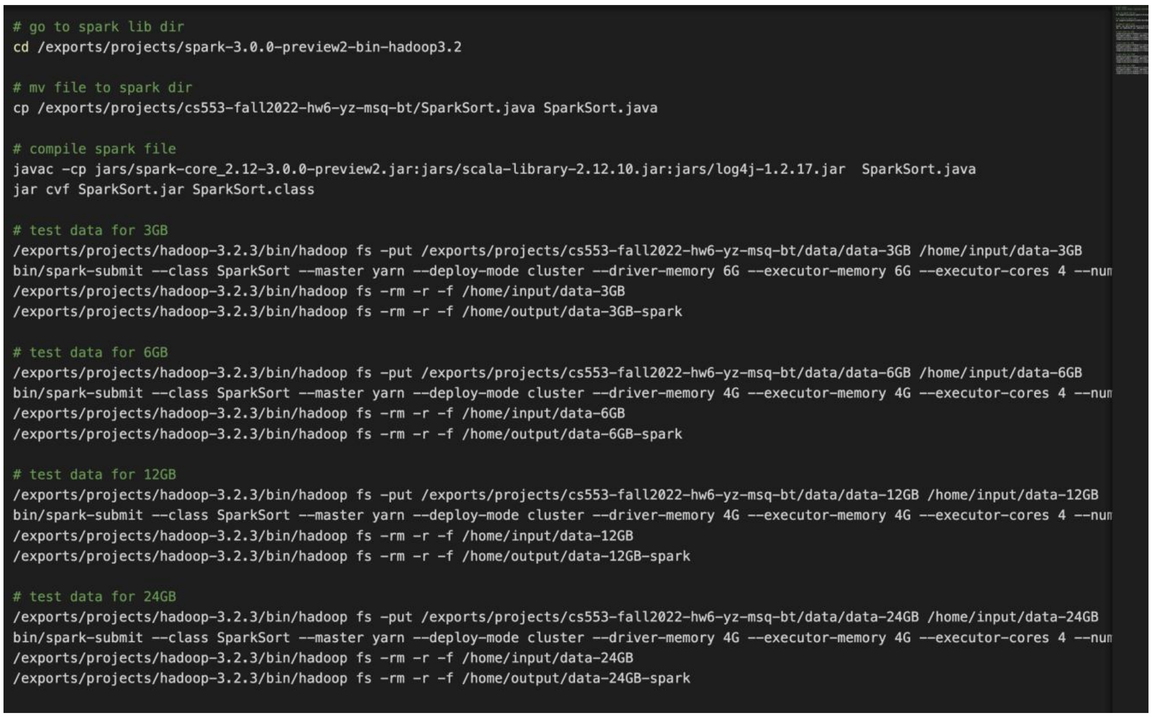
   all of the scripts are **Scripts/linsort.sh**

5.Execute **Scripts/spark.sh** to use spark program test program

We have to tune some spark parameter to optimize the performance, I test a lot of parameters. The good choice for parameters are:

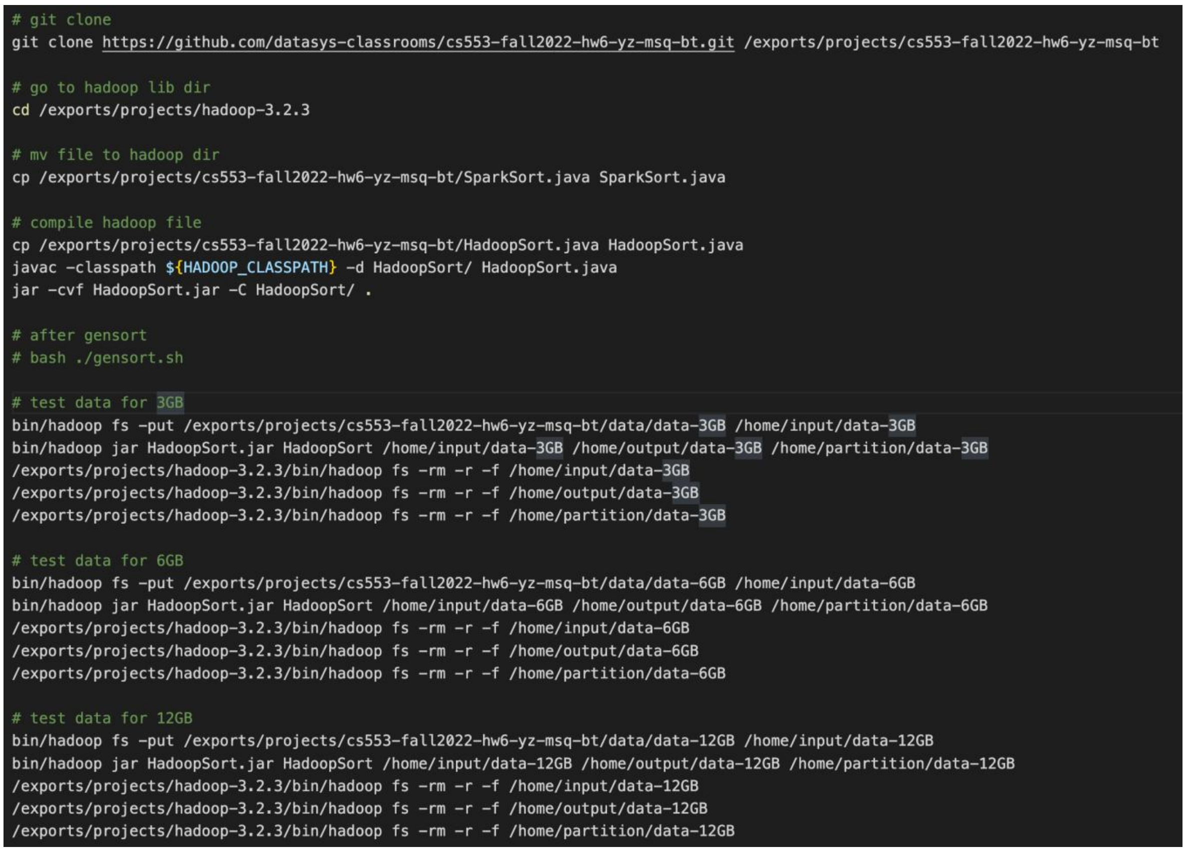
```
--master yarn
--deploy-mode cluster
--driver-memory 6G
--executor-memory 6G // task memory
--executor-cores 4 // task cores
--num-executors 80 // task service - test
  --conf "spark.default.parallelism=250"
```

The sparkSort() is more simplified than mapreducer. It provide some function, we can use and implement automatic to map and reduce. It also provide sort(), repartition() function.



```
# go to spark lib dir
cd /exports/projects/spark-3.0.0-preview2-bin-hadoop3.2

# mv file to spark dir
cp /exports/projects/cs553-fall2022-hw6-yz-msq-bt/SparkSort.java SparkSort.java

# compile spark file
javac -cp jars/spark-core_2.12-3.0.0-preview2.jar:jars/scala-library-2.12.10.jar:jars/log4j-1.2.17.jar  SparkSort.java
jar cvf SparkSort.jar SparkSort.class

# test data for 3GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-3GB /home/input/data-3GB
bin/spark-submit --class SparkSort --master yarn --deploy-mode cluster --driver-memory 6G --executor-memory 6G --executor-cores 4 --num
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-3GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-3GB-spark

# test data for 6GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-6GB /home/input/data-6GB
bin/spark-submit --class SparkSort --master yarn --deploy-mode cluster --driver-memory 4G --executor-memory 4G --executor-cores 4 --num
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-6GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-6GB-spark

# test data for 12GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-12GB /home/input/data-12GB
bin/spark-submit --class SparkSort --master yarn --deploy-mode cluster --driver-memory 4G --executor-memory 4G --executor-cores 4 --num
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-12GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-12GB-spark

# test data for 24GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-24GB /home/input/data-24GB
bin/spark-submit --class SparkSort --master yarn --deploy-mode cluster --driver-memory 4G --executor-memory 4G --executor-cores 4 --num
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-24GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-24GB-spark
```

5. Execute **Scripts/hadoop.sh** to use spark program test program.



```
# git clone
git clone https://github.com/datasys-classrooms/cs553-fall2022-hw6-yz-msq-bt.git /exports/projects/cs553-fall2022-hw6-yz-msq-bt

# go to hadoop lib dir
cd /exports/projects/hadoop-3.2.3

# mv file to hadoop dir
cp /exports/projects/cs553-fall2022-hw6-yz-msq-bt/SparkSort.java SparkSort.java

# compile hadoop file
cp /exports/projects/cs553-fall2022-hw6-yz-msq-bt/HadoopSort.java HadoopSort.java
javac -classpath ${HADOOP_CLASSPATH} -d HadoopSort/ HadoopSort.java
jar -cvf HadoopSort.jar -C HadoopSort/ .

# after gensort
# bash ./gensort.sh

# test data for 3GB
bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-3GB /home/input/data-3GB
bin/hadoop jar HadoopSort.jar HadoopSort /home/input/data-3GB /home/output/data-3GB /home/partition/data-3GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-3GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-3GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/partition/data-3GB

# test data for 6GB
bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-6GB /home/input/data-6GB
bin/hadoop jar HadoopSort.jar HadoopSort /home/input/data-6GB /home/output/data-6GB /home/partition/data-6GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-6GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-6GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/partition/data-6GB

# test data for 12GB
bin/hadoop fs -put /exports/projects/cs553-fall2022-hw6-yz-msq-bt/data/data-12GB /home/input/data-12GB
bin/hadoop jar HadoopSort.jar HadoopSort /home/input/data-12GB /home/output/data-12GB /home/partition/data-12GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/input/data-12GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/output/data-12GB
/exports/projects/hadoop-3.2.3/bin/hadoop fs -rm -r -f /home/partition/data-12GB
```

Data:

Experiment	**Linux**	**Hadoop Sort/s**	**Spark Sort/s**
1 small.instance, 3GB *dataset*	1m15.765s	8m2.75s	6m58.67s
1 small.instance, 6GB *dataset*	2m24.654s	16m45.78s	16m58.97s
1 small.instance, 12GB *dataset*	5m4.589s	41m4.78s	39m39.56s

1 large.instance, 3GB *dataset*	1m4.356s	4m15.33s	4m2.223s
1 large.instance, 6GB *dataset*	1m57.316s	8m65.65s	7m57.583s
1 large.instance, 12GB *dataset*	3m57.989s	20m67.5s	15m78.667
1 large.instance, 24GB *dataset*	11m26.987s	N/A	N/A
4 small.instances, 3GB *dataset*	N/A	6.8864m	5.6513m
4 small.instances, 6GB *dataset*	N/A	13.738m	11.7965m
4 small.instances, 12GB *dataset*	N/A	44.02m	41.34m
4 small.instances, 24GB *dataset*	N/A	N/A	N/A

Through the observation of data we can see that the performance of linux sort is the best, we can fully use memory and multi-threading, while the running performance of hadoop and spark is very dependent on parameter configuration, and the overall performance of hadoop is not as good as spark, and spark is much better than hadoop in terms of programming simplicity and ease of use.
We weren't able to run out 24G of data due to the length of time.