CS 553 Programming Assignment #2

# SORT ON HADOOP-SPARK-SHARED MEMORY REPORT A20356333

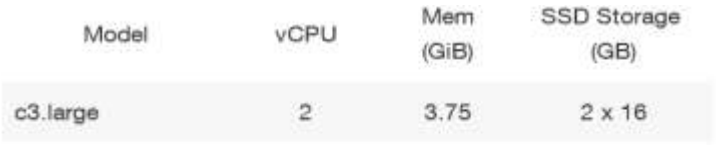
## SORT IMPLEMENTATION USING JAVA (SHARED MEMORY)/HADOOP/SPARK

- The assignment aims at implementing sort of huge data sets using Hadoop and spark
- The sorting implementation are run on Amazon ec2 instances – C3.large
- Shared memory sorting is implemented using JAVA
- Data sets used

  Single node Hadoop/Spark/Shared Memory – 10GB
  17 node (1 Master, 16 Slaves) Hadoop/Spark – 100GB
- For 17 node data sets test an additional EBS volume of 500 GB mounted in the case of Hadoop
- For 17 node data sets test an additional EBS volume of 750 GB mounted in the case of Spark
- Hadoop/ Spark sorting programs implemented using JAVA
- Data generated using gensort.
- Shared Memory Sort attempted on d2.xlarge for 1 TB sort, screenshots attached.

## SYSTEM SPECIFICATIONS

- Hadoop: Apache Hadoop 2.7.2
- Spark : spark-1.6.1-bin-hadoop2.6 (Spark 1.6.1 prebuilt for Hadoop 2.6.2)
- Java : oracle-jdk7(java-1.7.0_79)
- Build tool : Maven (For Spark)
- Amazon EC2: C3.large



| Model | vCPU | Mem (GiB) | SSD Storage (GB) |
|---|---|---|---|
| c3.large | 2 | 3.75 | 2 x 16 |

- Linux : Ubuntu
- D2.large ( sample run of Hadoop /shared memory done)
- Gensort : gensort-linux-1.5

## DESIGN & INSTALLATION

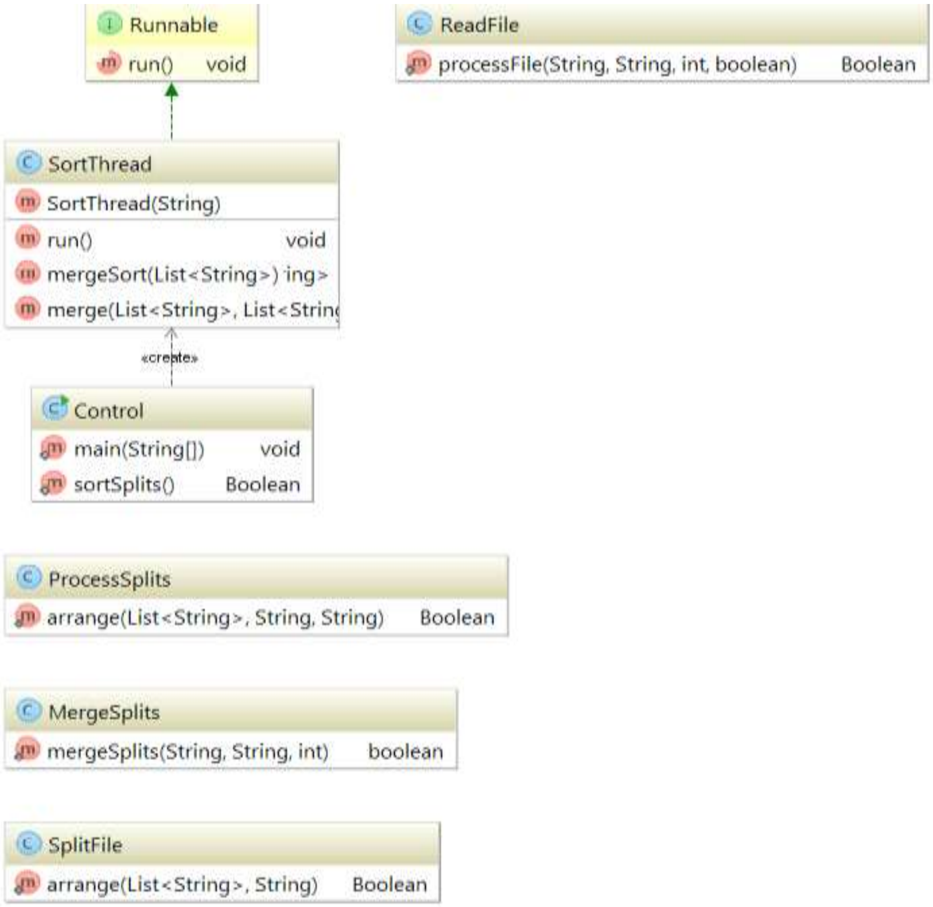
### a) Shared Memory Sorting(Java)

- Implemented using Java

## SORT ON HADOOP-SPARK-SHARED MEMORY REPORT A20356333

- Multithreaded sort: The multithreading is done only for the sort part of the code and is not done for the split part of the files. This was done to avoid the out of memory cases.
- Phases
  1) Read File
  2) Split File(based on first ascii character)
  3) Secondary Split the files created on step 2(Split the files again on second asci character)
  4) Run sorting on the secondary splits (Merge sort)
  5) Merge the split sorted files to a single output file
  6) The Temp files are deleted once merged to the output file
- Single read buffer size of files restricted to 5000000 to avoid memory exceptions
- Class Diagram

  Code explained in SharedMemoryCode.pdf

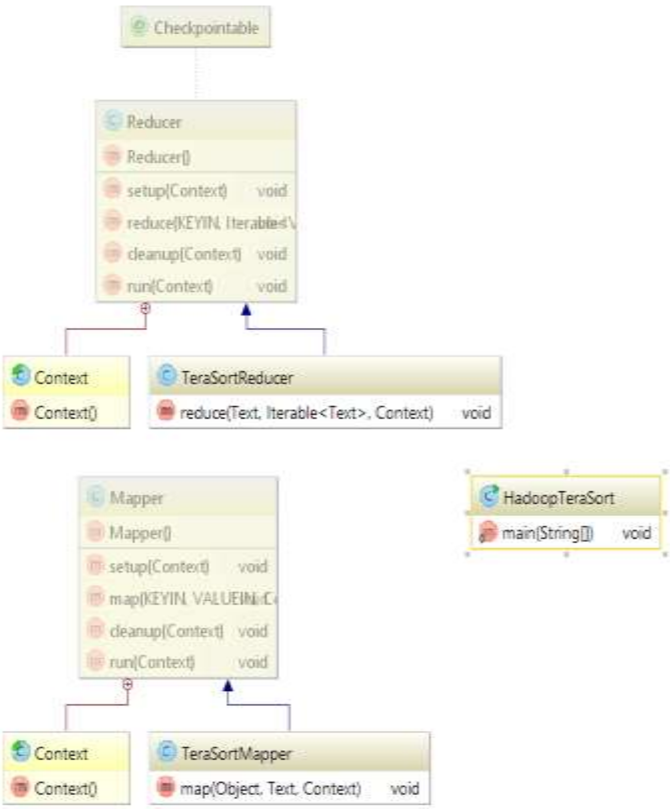

Runnable
run() void
ReadFile
processFile(String, String, int, boolean) Boolean
SortThread
SortThread(String)
run() void
mergeSort(List<String>) ing>
merge(List<String>, List<Strin
«create»
Control
main(String[]) void
sortSplits() Boolean
ProcessSplits
arrange(List<String>, String, String) Boolean
MergeSplits
mergeSplits(String, String, int) boolean
SplitFile
arrange(List<String>, String) Boolean

- Installation steps

  - cd SharedMemorySort/src
  - javac read/*.java
  - java -Xms1024m -Xmx2048m read.control
  - The program asks to enter the number of threads
  - Enter the threads as 1,2,4 & 8

**b) HADOOP (2.7.2)**

- Pseudo Distributed mode for single node installation
- Cluster implemented in 17 nodes(1master & 16 slaves)
- Java Code Design



CS 553 Programming Assignment #2

# SORT ON HADOOP-SPARK-SHARED MEMORY REPORT A20356333

## INSTALLATION STEPS HADOOP

### PSEUDO DISTRIBUTED

1) Install Java

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update && sudo apt-get install oracle-jdk7-installer
```

2) Download and extract Hadoop

```
 wget http://www-eu.apache.org/dist/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz
`  tar -xzvf hadoop-2.7.2.tar.gz
  mv hadoop-2.7.2 hadoop
```

3) Set up password less ssh

```
eval `ssh-agent -s`
chmod 600 terasort.pem
ssh-add terasort.pem
```

4) sudo vi .bashrc -- add the below lines

```
export CONF=/home/ubuntu/hadoop/etc/hadoop
 export JAVA_HOME=/usr/lib/jvm/java-7-oracle
 export PATH=:$PATH:$/home/ubuntu/hadoop/bin
```

5) Edit the Hadoop Configuration files

   cd hadoop/etc/hadoop

   a) vi core-site.xml

```
<property>
            <name>fs.default.name</name>
            <value>hdfs://ec2-52-27-182-219.us-west-
            2.compute.amazonaws.com:8020</value>
  </property>
   <property>
             <name>hadoop.tmp.dir</name>
            <value>/data/hadoop/tmp</value>
            <description>A base for other temporary directories.</description>
</property>
```

   b) vi hadoop-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
```

   c) vi hdfs-site.xml

```
<property>
<name>dfs.replication</name>
 <value>1</value>
 </property>
 <property>
<name>dfs.permissions</name>
<value>false</value>
</property>
```

d) cp mapred-site.xml.template mapred-site.xml

e) vi mapred-site.xml

```
<property>
        <name>mapreduce.jobtracker.address</name>
        <value>hdfs://ec2-52-27-182-219.us-
west.2.compute.amazonaws.com:8021</value>
</property>
<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
</property>
```

f) vi yarn-site.xml

```
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.resourcemanager.scheduler.address</name>
                <value>ec2-52-27-182-219.us-west-
    2.compute.amazonaws.com:8030</value>
        </property>
        <property>
                <name>yarn.resourcemanager.address</name>
                <value>ec2-52-27-182-219.us-west-
2.compute.amazonaws.com:8032</value>
        </property>
        <property>
                <name>yarn.resourcemanager.webapp.address</name>
                <value>ec2-52-27-182-219.us-west-
2.compute.amazonaws.com:8088</value>
        </property>
        <property>
                <name>yarn.resourcemanager.resource-tracker.address</name>
                <value>ec2-52-27-182-219.us-west-
2.compute.amazonaws.com:8031</value>
        </property>
        <property>
                 <name>yarn.resourcemanager.admin.address</name>
```

```
                    <value>ec2-52-27-182-219.us-west-
2.compute.amazonaws.com:8033</value>
            </property>
```

g) vi slaves

```
ec2-52-38-50-76.us-west-2.compute.amazonaws.com
```

**<u>CLUSTER CONFIGURATION</u>**

h) Repeat the above steps as pseudo mode installation in the master node

i) Install Hadoop and java in all the slave nodes as well

j) Move all the configuration file changes to slaves from the master node

<u>Master Node</u>

k) cd hadoop/etc/Hadoop

l) vi slaves

Enter the public dns of all the slave nodes in the file

<u>Slave Nodes</u>

m) Enter its own public dns in the slaves configuration file

    vi slaves

n) Make sure SSH works in between all the nodes

**<u>STEPS TO RUN THE HADOOP APPLICATION</u>**

a) cd hadoop/bin

    ./hdfs namenode -format

    cd ..

b) cd hadoop/sbin

c) ./start-dfs.sh

    jps

d) ./start-yarn.sh

    if any issue : format namenode delete dfs folders and restart

e) cd hadoop/bin

f) ./hadoop dfs -copyFromLocal /home/ubuntu/64/onegb.txt  /onegb

g) ./hadoop jar /home/ubuntu/HadoopSort.jar /onegb   /sortedonegb

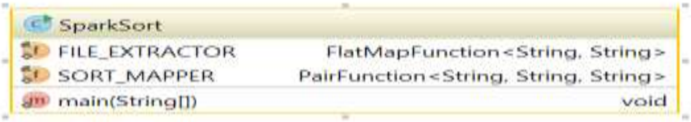
h) ./hadoop dfs -copyToLocal   /sortedonegb  /home/ubuntu/64/onegb.txt

i) cd Hadoop/sbin

j) ./stop-dfs.sh
k) ./stop-yarn.sh

C) SPARK

- Single node Spark
- 17 Node spark ( 1master+16 slaves)
- Spark program for sorting implemented using Java
- Maven used as build tool for spark code



```
SparkSort
FILE_EXTRACTOR          FlatMapFunction<String, String>
SORT_MAPPER        PairFunction<String, String, String>
main(String[])                                     void
```

- **INSTALLATION STEPS:**

1) Login to the AWS console. Under the account name at the top right, select security credentials
Download access from security credential
Access Key ID:
Secret Access Key:
2) Launch a default instance to boot the spark cluster
3) Login to the instance using key pair from putty
4) copy the pem file to instance using winscp
5) Download spark

- wget http://www-us.apache.org/dist/spark/spark-1.6.1/spark-1.6.1-bin-hadoop2.6.tgz
- tar -xzvf spark-1.6.1-bin-hadoop2.6.tgz
- mv spark-1.6.1-bin-hadoop2.6.tgz spark

6) export AWS_ACCESS_KEY_ID=
7) export AWS_SECRET_ACCESS_KEY=
8) cd spark/ec2
9) Below command to launch the spark cluster with 16 slaves and one master

```
./spark-ec2 --key-pair=hadoopec2cluster --identity-file=/home/ubuntu/hadoopec2cluster.pem
--region=us-east-1 --slaves=16 --instance-type=c3.large --ebs-vol-size=750 --ami=ami-877142ed
--spot-price=0.04 launch spark
```

10) Login into spark cluster

```
./spark-ec2 -k hadoopec2cluster -i /home/ubuntu/hadoopec2cluster.pem -r us-east-1 login Spark
```

11) Deploy the spark jar file to run sorting
12) Login to the master node using winscp

```
~/spark-ec2/copy-dir  /root/SparkSort-1.0-SNAPSHOT.jar
```

13) cd ephemeral-hdfs/bin
14) ./hadoop dfs -copyFromLocal /datafile.txt /datain
15) Run the below command to submit the job

```
./spark-submit --class SparkSort --master local[8] --executor-cores 2 --conf spark.driver.memory=2g
  --conf spark.executor.memory=2g /root/SparkSort-1.0-SNAPSHOT.jar /sorted100gb /sortedout
```

16) cd ephemeral-hdfs/bin
17) ./hadoop dfs -copyToLocal /dataout /sortedout

**Note: Performance evaluation in prog2_report_performance_evaluation.pdf and the answers to the performance related questions also present in the evaluation pdf.**

**Questions:**

After you run Hadoop on 1 node first, you will likely have to modify these configuration files as you go to a multi-node run:

1) conf/master

2) conf/slaves

3) conf/core-site.xml

4) conf/hdfs-site.xml

5) conf/mapred-site.xml

You are to write a description of the function of each file, and what modifications you had to make to go from 1 node to multiple nodes. Please answer the following questions:

1) **<u>conf/master</u>**:
   This configuration file holds the hostname of the hadoop secondary name node server and name node. Hadoop daemon learns **about the host name and secondary name node** server information from this file.
2) **<u>conf/slaves</u>** :
   This configuration file holds the hostnames of the **hadoop data nodes and task trackers.** In the master node all the slave node hostnames are mentioned in this file. In the slaves, only the node specific hostname is mentioned
3) **conf/core-site.xml**:
   This file holds the information regarding **the name node**. Hadoop daemon learns the name node location in the cluster using the property in core-site.xml. Also this has the property which defines the **temporary directory** for hadoop files.
   fs.default.name: URI of name node

4) **conf/hdfs-site.xml:**
Name node, data node and secondary name node configuration settings are present.
Property to set **Block replication** for the hadoop is maintained in hdfs-site.xml
dfs.name.dir: Path on the local filesystem where the NameNode stores the namespace and transactions logs persistently
dfs.data.dir: list of paths where data node stores its blocks

5) conf/mapred-site.xml:
Job Tracker and task tracker hadoop deamons configuration settings are added in mapred-site.xml.
Also the path in which the mapreduce framework stores the system files on hdfs.

The above configurations used for Hadoop configuration can be found under the installation steps.

1) What is a Master node? What is a Slaves node?

Master Node: This node runs the name node and the job tracker

Slave nodes: These nodes runs the Data node and Task Tracker for the Hadoop

2) Why do we need to set unique available ports to those configuration files on a shared environment? What errors or side-effects will show if we use same port number for each user?

Ensure resource availability

Avoid port number conflicts

3) How can we change the number of mappers and reducers from the configuration file?

This can be configured in the mapred-site.xml

mapred.tasktracker.{map|reduce}.tasks.maximum – defaults to 2 , but can be changed based on the number of cores available

The number of reducers can be changed by setting the job.setNumReduceTasks() programmatically. The number of mappers are usually determined by the number of input splits.