Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

CS 553 CLOUD COMPUTING [Fall 2017]

DESIGN DOCUMENT PROGRAMMING ASSIGNMENT#2

TeraSort on Shared-Memory/Hadoop/Spark/MPI

This assignment aims to perform sorting on large data set where key size is 10 bytes and value is of 90 bytes on AWS EC2 instance and compare their result. This document goes though problem statement, Approach, configuration and running, graph and explanation, problem faced and improvement and extension of

1. C (shared-memory sort)

2. Hadoop

3. Spark

4. MPI

5.Relevant Screenshots

6. Comparison between Shared memory/Hadoop/Spark/MPI Sort

7.Q/A

C (Shared-memory sort)

Problem Statement:

To write a program in any language that sort dataset of size 128 GB and 1TB on i3.large, i3.4xlarge disk I/O optimized single node instance. As we have data size much bigger than memory of i3.large & i3.4xlarge; we need to find an approach that will sort that large dataset.

Approach

To solve the above problem, where memory size is smaller than data set to sort I used the below steps

1. Read file data of buffer size/2 and place that data in separate files. Basically, we are dividing the files in such a way that it fits in the memory. We cannot create a chunk equal to buffer size because we need some space for sorting algorithms

2. Using multi-threading open the files and perform sort on each file. Each thread will sort one file. Because of memory limit we are not able to sort all file in parallel.

4. Merging File chunks: As we have number of sorted files there are two ways now we can approach our problem a) two way merging (Merging two files at a time) b) K way-merging (Take

certain MB of data from each file into buffer and perform sort and place 1 chunk of sorted data from buffer to file). We have used k-way merging approach because it is more efficient for sorting large data set and for performance (less I/O operations) .

5. K-way Merging: In k-way merging, I have used priority queue that opens all the file at once and based on priority it sorts the data.

6. Store the output of k-way merging in a file at disk.

Configuration and Experiment

Model	vCPU	Mem (GB)	Storage (GB)
i3.large	2	15.25	475
i3.4xlarge	16	122	3800

Raid configuration script is attached in scripts folder (Mounted RAID0 at /mnt/raid location)

Kernel Version: GNU/Linux

OS USED: Ubuntu 16.04

JAVA VERSION: "1.8.0_151"

HADOOP VERSION:2.7.4

GCC VERSION: 5.4.0

Solving 128 GB dataset:

128 GB data set is a large dataset as we have memory size only of 15.5 GB for an i3large VM Instance & it's an Ok data set for i3.4xlarge VM instance as the memory size for i3.4xlarge is 122 GB. For i3large we have been dividing 128GBfile into file size of 8GB and for 1 TB we have divide into 1 GB. (1000 Files) as we have more memory.

Solving 1 TB dataset:

Solving 1TB data set took me lots of effort and hit-trial on running 10GB dataset to get the estimate of number of threads and files to generate for 1TB dataset as 1TB is of very big size and takes lots of time to run any small problem could have caused my program to terminate. To sort data set of size 1TB, I created 1000 files of 1GB each. It took approximately 8 hours to sort on i3.4xlarge VM instance.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Problem faced

1.Too many files opened exception: This problem will come when you try to divide your 128 GB file into very small file creating too many files which will exceed the limit of OS

2.Dividing the large file into many small files based on memory size due to which memory got full and no place was left for the sorting algorithm to run.

3. How to divide memory for into two parts of loading the input in first part and second part for other things like sorting in memory, variables assignments so that memory do not get full and carefully assigning the thread and files to them so that they do not makes my memory full.

Improvements and extensions

Carefully dividing the memory among threads and creating chunks of small size of a file so that many threads can run in parallel which can improve the performance.

Hadoop TeraSort

Problem Statement:

In this program, we have written a java programs that sort dataset of size 128 GB and 1TB on i3.large, i3.4xlarge disk I/O optimized single node instance & 8 node HDFS cluster on i3.large instance type. As we have data size much bigger than memory of i3.large & i3.4xlarge; we need to find an approach that will sort that large dataset.

Approach

Attempting this problem mainly deals with configuration of Hadoop single node cluster and multi node cluster that we have explained in below configuration section. For code we need to add jar file in our code configuration files as well as in jar files. Code consists of a driver class that runs the map/reduce job. There is one mapper class and reducer class used by driver class. I have used KeyValueTextInputFormat for reading file because for this type of file Hadoop separates key by first tab and remaining data in same line as values, which is perfect fit for our case. We have not used combiner in our approach as we have unique keys.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Configuration and Experiment

Model	vCPU	Mem (GB)	Storage (GB)
i3.large	2	15.25	475
i3.4xlarge	16	122	3800

Raid configuration script is attached in scripts folder (Mounted RAID0 at /mnt/raid location)

Kernel Version: GNU/Linux

OS USED: Ubuntu 16.04

JAVA VERSION: "1.8.0_151"

HADOOP VERSION:2.7.4

Running Experiment on Single node (i3large & i34xlarge)

Hadoop Single node setup: To setup single node of Hadoop we need to do following configuration changes written below in /etc/hadoop/ file:

core-site.xml

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://ec2-13-59-246-215.us-east-2.compute.amazonaws.com:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

hdfs-site.xml

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>

<property>
```

```
  <name>dfs.namenode.name.dir</name>
  <value>/mnt/raid/tmp/namenode</value>
</property>

<property>
  <name>dfs.blocksize</name>
  <value>268435456</value>
</property>

<property>
  <name>dfs.namenode.handler.count</name>
  <value>100</value>
</property>

<property>
  <name>dfs.datanode.data.dir</name>
  <value>/mnt/raid/tmp/datanode</value>
</property>
```

yarn-site.xml

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value></property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>ec2-13-59-246-215.us-east-2.compute.amazonaws.com:9025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>ec2-13-59-246-215.us-east-2.compute.amazonaws.com:9030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>ec2-13-59-246-215.us-east-2.compute.amazonaws.com:9050</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>ec2-13-59-246-215.us-east-2.compute.amazonaws.com:9006</value>
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>ec2-13-59-246-215.us-east-2.compute.amazonaws.com:9008</value>
</property><!---->
<property>
<name>yarn.nodemanager.vmem-pmem-ratio</name>
<value>2.1</value>
</property>
```

/conf/slave

Change this file default local host to aws instance name. After doing the above configuration changes we need to format hdfs

mapred-site.xml

```
<property>
  <name>mapred.job.tracker</name>
  <value>ec2-13-59-246-215.us-east-2.compute.amazonaws.com:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
```

Running Hadoop

./bin/hdfs namenode –format
then we need to start all services ./sbin/start-all.sh through the start all scripts.

Create directory in HDFS
./bin/hadoop dfs –mkdir /mnt/raid/TeraSort/HDFSIn

Copy the file generated by (Gensort) from local directory to a directory in HDFS
./bin/hadoop dfs -copyFromLocal /mnt/raid/TeraSort/Input/actualInput.txt /mnt/raid/TeraSort/HDFSIn/output

For single node i3large cluster, we have run an experiment for 128 GB dataset, and for single node i34xlarge cluster, we have run an experiment for 1 TB dataset.

For an 8 node i3large cluster, we have run an assignment for 1 TB dataset.

Running Experiment on Multiple nodes (8 node i3.large instance)

To set multiple nodes for hadoop we need to do keep the above configuration with few extra changes on below

mapred-site.xml

```
<property>
<name>mapreduce.cluster.local.dir</name>
<value>/mnt/raid/TeraSort</value>
</property>
<property>
<name>mapreduce.jobtracker.system.dir</name>
<value>/mnt/raid/TeraSort</value>
```

```
</property>
<property>
<name>mapreduce.jobtracker.staging.root.dir</name>
<value>/mnt/raid/TeraSort</value>
</property>
<property>
<name>mapreduce.cluster.temp.dir</name>
<value>/mnt/raid/TeraSort</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Xmx2048m-XX:+UseParallelOldGC</value>
</property>
```

hadoop-env.sh

```
export    HADOOP_CLIENT_OPTS="-Xmx2048m $HADOOP_CLIENT_OPTS"
export    HADOOP_PORTMAP_OPTS="-Xmx2048m -XX:+UseParallelOldGC
$HADOOP_PORTMAP_OPTS"
```

./conf/slave

We have written script file to setup all nodes in the environment i.e. copying configuration files to other Servers (Slaves), and the hostname must be setup for all the nodes in the cluster in /etc/hosts file.

Running Hadoop

Once we have made the above described changes, we need to start the dfs and yarn from master node as done for single node cluster.

Problem faced

1.JVM out of heap space: For this issue, I changed the JVM heap size in environment variable
2.Cannot create a tmp directory or space full. For this we have changed the path of tmp directory to our disk so that tmp directory can contain large amount of data. Also changed the directory permission as 777 so that we do not face any problem
3.Terminal closed in between after running for several hours.
4.Getting an approval from amazon to setup 8 node cluster as we have limit of 1.
5.Additional cost apart from credit provided
6.Data distribution among mappers.

Improvements and Extensions:

1.Changing the block size of HDFS so that large blocks can be read at a time which will improve performance.

2.Carefully setting the number of mappers and reducers in map reduce program and distribute equal amount of data among mappers.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Spark TeraSort

Problem Statement

In this part of assignment, we have written a “SCALA program” to sort data set of 128 GB and 1 TB on single and 8 nodes hadoop cluster. This experiment aims to explore spark configuration, installation and learn spark API to write program and perform benchmarking experiment.

Approach

This problem mainly deals with configuration of spark cluster that I have explained in below configuration section.

Configuration and Experiment

Model	vCPU	Mem (GB)	Storage (GB)
i3.large	2	15.25	475
i3.4xlarge	16	122	3800

Kernel Version: GNU/Linux

OS USED: Ubuntu 16.04

JAVA VERSION: "1.8.0_151"

HADOOP VERSION:2.7.4

SPARK VERSION: 1.6.3

Raid configuration script is attached in scripts folder (Mounted RAID0 at /mnt/raid location

Running Experiment on Single node (i3large & i34xlarge)

Spark Single node setup: Spark setup is much easier than hadoop as they have script written to run install on AWS EC2 instance.

Spark programs can be written in python and scala by using the Spark API. It is easy to write a scala code as compared to python. Reference to write scala code for spark have been provided in Reference.txt file. Scala will contains only few lines to execute TeraSort program.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Running Experiment on Single and Multiple nodes

Need to export AWSAccessKeyId and AWSSecretKey then we need to run the below

~/home/sandeep/Desktop/CS553/spark-1.6.3-bin-hadoop-2.7.4/ec2/spark-ec2 --keypair=Ec2s --identity-file=/home/sandeep/Desktop/EC2/PA2sandeepabhinav.pem –region=us-east-2 –zone=us-east-2a –instance-type=c3.large –slaves=7 –ebs-vol-size=1024 –spot-price=0.03 launch spark

We need to do additional configuration as running sort for 128 GB and 1 TB as EBS volumes are ephemeral and hence we need to make these disks permanent.

Changes that must be done to ensure persistent hdfs:

1. ./ephemeral-hdfs/bin/stop-all.sh
2. sed -I s#vol/persistent-hdfs#vol0/persistent-hdfs#g'~/persistent-hdfs/conf/core-site.xml
3. ~/spark-ec2/copy-dir.sh~/persistent-hdfs/conf/core-site.xml
4. ~/persistent-hdfs/bin/hadoop namenode -format
5. ~/persistent-hdfs/bin/start-all.sh
6. ./persistent-hdfs/bin/hadoop dfs -mkdir /inputdir

   Now change spark configuration to use persistent-hdfs instead of ephemeralhdfs and set default directory from /mnt to /vol0.

Go to spark config: /spark/conf/core-site.xml

1. Change the hdfs url port from 9000 to 9010
2. Change spark-env.sh
3. spark default directory from /mnt to /vol0
4. Restart Spark "Go to sbin and stop-all.sh then start-all.sh

For spark experiment I have created 7 slaves and 1 master node with data set of 128 GB and 1024 GB.

Problem Faced

1.Cannot create tmp directory same problem as that of Hadoop.

2.Running spark on 8 nodes. For this we have attached links in Reference.txt

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

MPI TeraSort

Problem Statement:

In this program, we have written a c programs using mpi that sort dataset of size 128 GB and 1TB on i3.large, i3.4xlarge disk I/O optimized single node instance & 8 node HDFS cluster on i3.large instance type. As we have data size much bigger than memory of i3.large & i3.4xlarge; we need to find an approach that will sort that large dataset.

Approach

Start and initialize MPI.

1.Firstly we have used c code to divide our large file into small chunks so that they can fit in the buffer

2.We have created a sort program with mpi to sort individual files. The mpi will work in the following way

i. Under the root process MASTER, load all the data into buffer from that file.:
    a. Read the line L from an input file.
    b. Initialize the main array globaldata with L.
    c. Start the timer.
ii. Divide the input size SIZE by the number of participating processes npes to get each chunk size localsize.
iii. Distribute globaldata proportionally to all processes:
    a. From MASTER scatter globaldata to all processes.
    b. Each process receives in a sub data localdata.
iv. Each process locally sorts its localdata of size localsize based on key.
v. Master gathers all sorted localdata by other processes in globaldata.
    a. Gather each sorted localdata.
    b. Free localdata.
vi. Under MASTER perform a final sort of globaldata.
    a. Final sort of globaldata.
    b. Stop the timer.
    c. Write the output to file.
    d. Sequentially check that globaldata is properly and correctly sorted.
    e. Free globaldata.
vii. Finalize MPI.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

3. Lastly, we perform merge using mpi for all the small chunks into a single sorted file

The approach works in the following way:

I. Open all the files

ii. Take small amount of data from each file of same size. Use MPI approach mentioned above for a single file

iii. Put the first chunk of buffer into a sorted file. Fill this chunk with data from the file in sequential order and repeat step 2 and 3 until you have loaded the last chunk in MB of data from last file.

iv. Place all the data from buffer into the sorted file which is opened in step 3.

v. Free buffer memory and close the sorted file.

Configuration and Experiment

Model	vCPU	Mem (GB)	Storage (GB)
i3.large	2	15.25	475
i3.4xlarge	16	122	3800

Kernel Version: GNU/Linux

OS USED: Ubuntu 16.04

MPI VERSION: "3.2.1"

GCC VERSION:5.4.0

Running MPI on single node

The setup for single node is easy. We have written the steps below

tar -xzf mpich2-1.4.tar.gz

cd mpich2-1.4

./configure --disable-fortran

make; sudo make install

Compile your 3 files:

gcc -o TeraSort mpi_Terasort.c
mpii -o sort sort.c
mpii -o merge merge.c

Now Run you file
./Terasort input_file_name

Running MPI on 8 node cluster

sudo easy_install StarCluster

starcluster help

Since StarCluster is not configured, it will print out the following

StarCluster - (http://web.mit.edu/starcluster) (v. 0.93.3)

Software Tools for Academics and Researchers (STAR)

Please submit bug reports to starcluster@mit.edu

!!! ERROR - config file /Users/wesleykendall/.starcluster/config does not exist

Options:

[1] Show the StarCluster config template

[2] Write config template to /Users/wesleykendall/.starcluster/config

[q] Quit

Please enter your selection:

Enter the number 2 and StarCluster will generate a default configuration file in your home directory under ~/.starcluster/config.

Obtain your AWS access key, secret access key, and your 12-digit user ID from your AWS account.

open your default config file (~/.starcluster/config) with your favorite text editor. Find the line with [aws info] and enter all of your AWS information into the proper fields:

AWS_ACCESS_KEY_ID = # Your Access Key ID here
AWS_SECRET_ACCESS_KEY = # Your Secret Access Key here
AWS_USER_ID = # Your 12-digit AWS Account ID here (no hyphens)

save the config file and then create a public/private key pair that will be uploaded to Amazon and used to authenticate your machine when you log into your cluster

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

starcluster createkey mykey -o ~/.ssh/mykey.rsa

configuring cluster parameters

KEYNAME = mykey

CLUSTER_SIZE = put the size of cluster

CLUSTER_USER = username genrated

CLUSTER_SHELL = bash

NODE_IMAGE_ID = virtual machine image

NODE_INSTANCE_TYPE = instance type

enable the mpich2 plugin for Starcluster. Compile your program in the same way as single node and run your program.

Problem Faced

1.Setting up star cluster in 8 nodes.

2.The size of the chunk the file needs to be divided so that memory does not get full.

3.Terminal connection closed for 1 TB in between due to which we decided to run on small size .

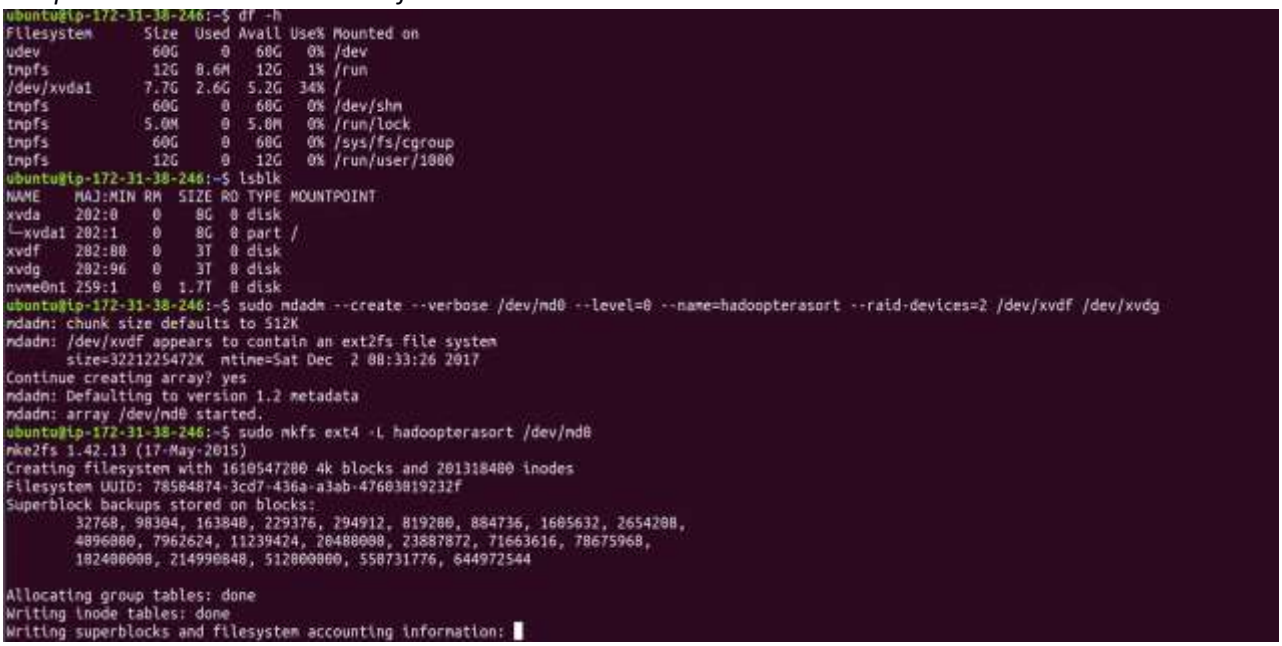
4.Amazon credit limit exhausted.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

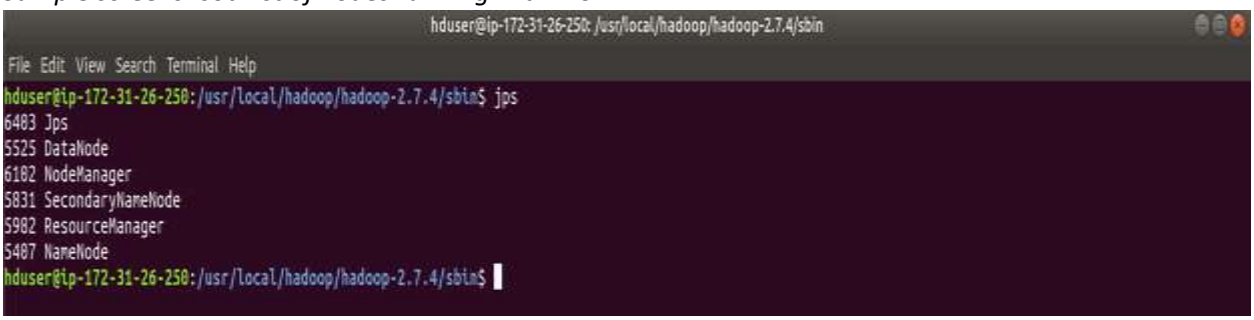
Screenshots

NOTE: All the screenshots are also attached in a separate file called Screenshot.pdf attached with this report. Below is a summary of 1 screenshot for each type

Sample screenshot Raid Creation for Data Node:

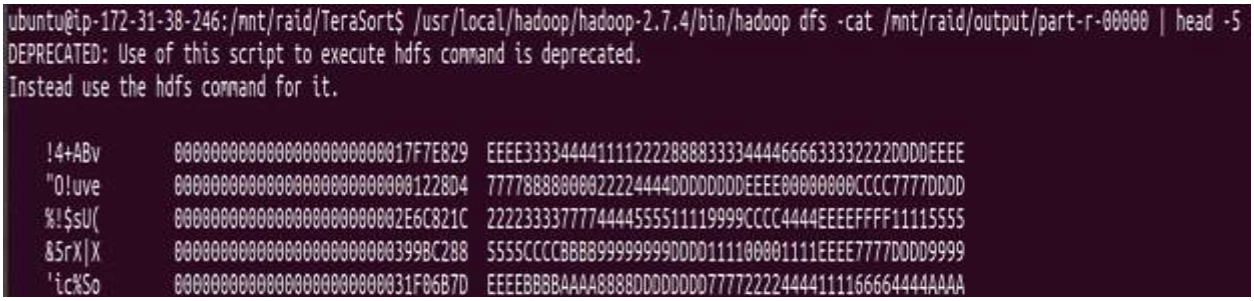


Sample screenshsot List of nodes running with JPS

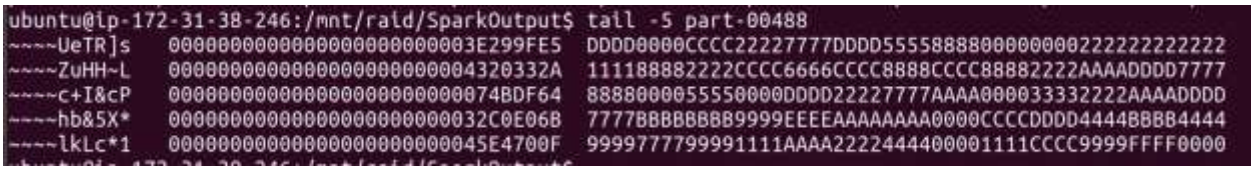


Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Sample screenshot of Output(head)



Sample screenshot of output tail



Sample screenshot of valsort

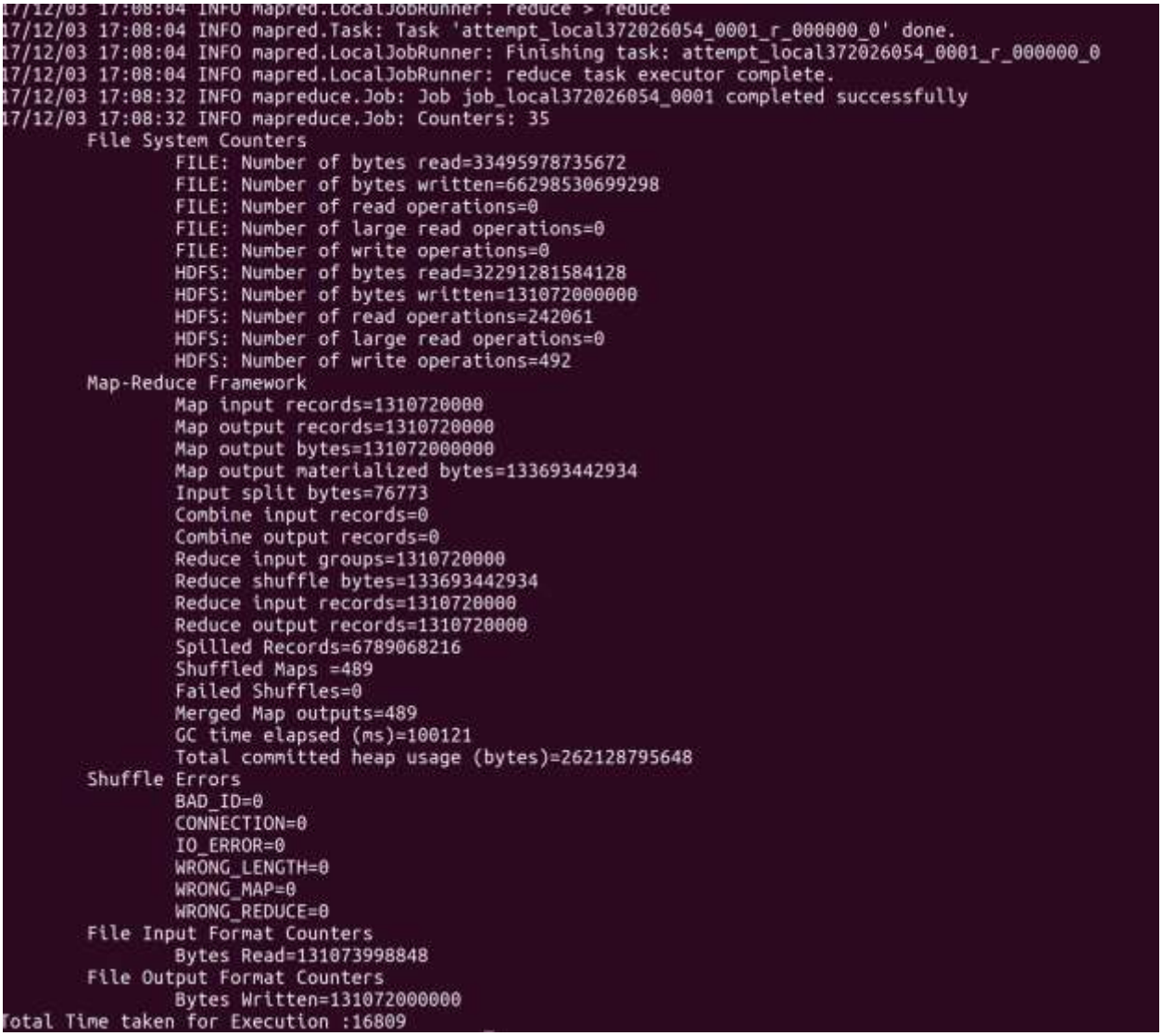
Sample screenshot of raid configuration

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

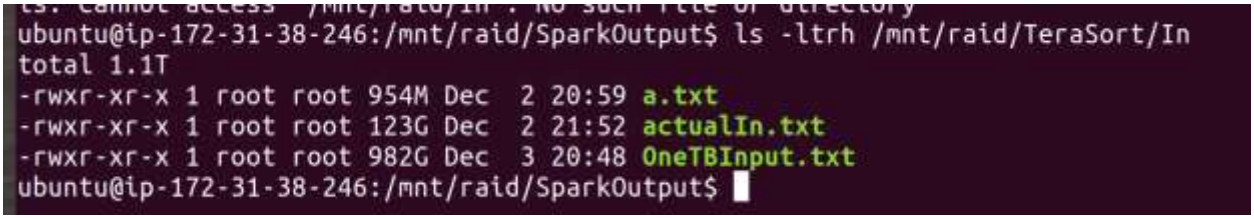
Sample screenshot of Spark

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Sample screenshot of time output



Screenshot of input files generated



Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Sample Screenshot of input file in HDFS

```
ubuntu@ip-172-31-38-246:/mnt/raid/SparkOutput$ /usr/local/hadoop/hadoop-2.7.4/bin/hadoop dfs -ls /mnt/raid/HDFSIn
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 1 items
-rw-r--r--   1 hduser supergroup 131072000000 2017-12-03 03:26 /mnt/raid/HDFSIn/actualIn.txt
ubuntu@ip-172-31-38-246:/mnt/raid/SparkOutput$
```

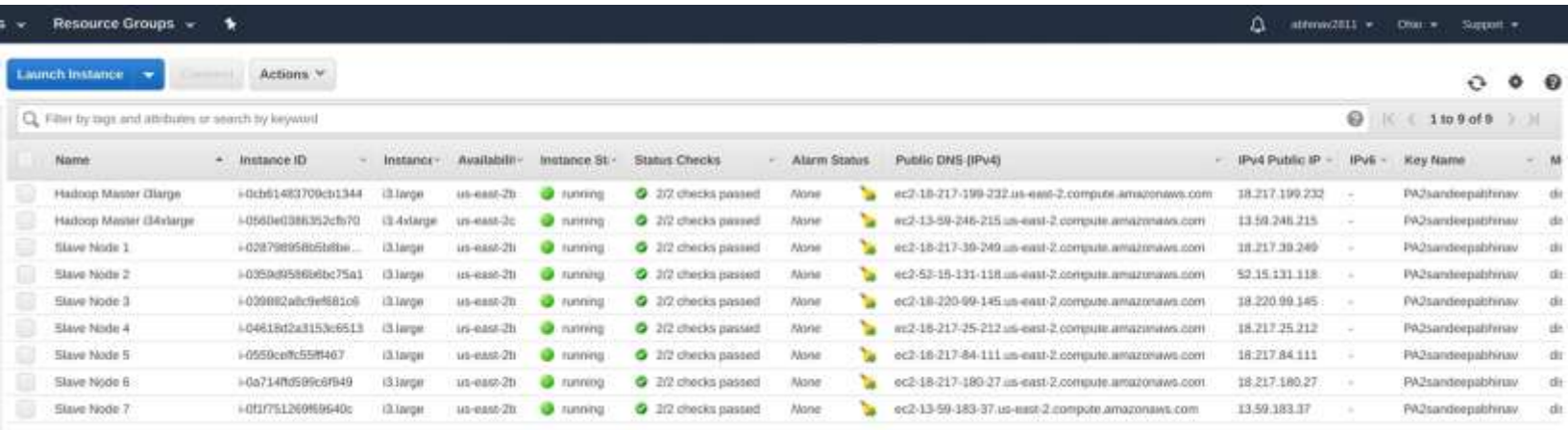
Screenshot of TeraSort sorted Output for 1 TB:

```
ubuntu@ip-172-31-26-250:cat /mnt/raid/SharedMemoryOutputfor1TB/SortedOutput.txt | head -5

   !"L5R;   0000000000000000000000001A68EC8A8   AAAA0000BBBBDDDD4444AAAAFFFF1111DDDD2222DDDD33339999
   !4+ABv   00000000000000000000000017F7E829   EEEE333344441111222288883333444466663333222DDDDEEEE
   "O!uve   000000000000000000000000001228D4   777788880000222244444DDDDDDDDEEEE00000000CCCC7777DDDD
   $h#_DN   00000000000000000000001BCB770C2   55554444CCCCCCCC444422221111999955552222777711111FFFF
   %q/-'7   000000000000000000000000A674E940   7777FFFF1111EEEE4444DDDDCCCC8888AAAACCCC1111CCCC1111

ubuntu@ip-172-31-26-250:cat /mnt/raid/SharedMemoryOutputfor1TB/SortedOutput.txt | tail -5

~~~UeTR]s        0000000000000000000000003E299FE5   DDDD0000CCCC22227777DDDD555588880000000222222222222
~~~ZuHH~L        00000000000000000000000004320332A   111188882222CCCC6666CCCC8888CCCC88882222AAAADDDD7777
~~~c+I&cP        000000000000000000000000074BDF64   8888000055550000DDDD22227777AAAA000033332222AAAADDDD
~~~hb&5X*        0000000000000000000000032C0E06B   7777BBBBBBBB9999EEEEAAAAAAAA0000CCCCDDDD4444BBBB4444
~~~lkLc*1        00000000000000000000000045E4700F   9999777799991111AAAA2222444400001111CCCC9999FFFF0000
```

Screenshot of single node instance-type

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

Screenshot of 8 node cluster



Experiment (instance/dataset)	Shared Memory TeraSort	Hadoop TeraSort	Spark TeraSort	MPI
Compute Time (sec) [1xi3.large 128GB]	10200	16809	19333	9605
Data Read (GB) [1xi3.large 128GB]	123	123	123	123
Data Write (GB) [1xi3.large 128GB]	123	123	123	123
I/O Throughput (MB/sec) [1xi3.large 128GB]	12.35	7.49	6.51	13.11
Compute Time (sec) [1xi3.4xlarge 1TB]	53340	42900	34620	44940
Data Read (GB) [1xi3.4xlarge 1TB]	982	982	982	982
Data Write (GB) [1xi3.4xlarge 1TB]	982	982	982	982
I/O Throughput (MB/sec) [1xi3.4xlarge 1TB]	18.85	23.43	29.04	22.37
Compute Time (sec) [8xi3.large 1TB]	N/A	33420	25620	41820
Data Read (GB) [8xi3.large 1TB]	N/A	982	982	982
Data Write (GB) [8xi3.large 1TB]	N/A	982	982	982
I/O Throughput (MB/sec) [8xi3.large 1TB]	N/A	30.08	39.24	24.04
Speedup (weak scale)	NA	1.28	1.35	1.07

Performance evaluation table of TeraSort

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

NOTES

128 GB is calculated as 128*1024*10^6(Reason for 123 GB data generation instead of 128GB)

1 TB calculation =1054*10^9(so as to get approx. 1 TB).

Speedup calculation for shared memory is not calculated because we have different data size for i3.large and i3.4x large and also i3.large 8 node cluster calculation is not available

Speedup (Throughput on i3.4xlarge)/throughput on i3.large for 1 TB for shared memory.

Questions and Answers

1) What conclusions can you draw?

Solution) For small datasize ideally shared and memory and MPI works better than hadoop/spark because hadoop/spark requires additional startup cost by setting up the master and slave communication.

2) Which seems to be best at 1 node scale?

Solution) Sparks seems to best at 1 node scale as spark uses RDD and makes effective use of memory.

3) How about 8 nodes? Can you predict

Solution) Sparks seems to best at 1 node scale as spark uses RDD and makes effective use of memory.

4) Which would be best at 100 node scales ?

Solution) Sparks seems to best at 100 node scales as spark uses RDD and makes effective use of memory and from the result we see spark performance drastically improved as we keep increasing nodes

5) How about 1000 node scales?

Solution) Sparks seems to best at 1000 node scale as spark uses RDD and makes effective use of memory and from the result we see spark performance drastically improved as we keep increasing nodes.

NOTE: For small dataset Shared memory and MPI would be best option to choose while in large dataset with 100 or 1000 node scale Spark will be best.

Sandeep Vuzzini [A20243379]
Abhinav Pimpalgaonkar[A20387324]
Shruthi Akkala [A20376505]

6) Compare your results with those from the Sort Benchmark [9], specifically the winners in 2013 and 2014 who used Hadoop and Spark.

Solution) Looking at the 2013 and 2014 result of Hadoop and Spark Gray Sort Hadoop took 102.5 TB in 4,328 seconds in 2013 and in 2014 Apache Spark took 100 TB in 1,406 seconds Hadoop took 102.5 TB in 4,328 seconds. Both the result is far better than the result we calculated in this sorting benchmark

7) Also, what can you learn from the CloudSort benchmark, a report can be found at [10]. All of these questions must be addressed in your final report write-up for this assignment.

Solution) This Benchmark helps us in understanding the concept of aws, shared memory hadoop and spark and concepts of external sorting.

----------END of DOCUMENT----------