CS 553 Cloud Computing
Programming Assignment 2
Sujay Gunjal (CWID: A20351746)
Report

Problem statement:-

This programming assignment covers the Sort application implemented in 3 different ways: Java, Hadoop, and Spark. Sorting application could read a large file and sort it in place (your program should be able to sort larger than memory files). Have to create 2 datasets, a small and a large dataset, which will be used to benchmark the 3 approaches to sorting: 10GB dataset and 100GB dataset.

Methodology:-

I am going to use **merge sort and external sort** for sorting files using shared memory sort. Hadoop hadoop-2.7.2 will be installed on amazon ec2 instance to process 10GB and 100GB file. Similarly I will use spark spark-1.6.0 on amazon ec2 instance to sort 10GB and 100GB files.

Version details of software used for Hadoop and spark installations:-

Linux version:- Ubuntu 14.04.4 LTS

Java Version:- java version "1.7.0_99" and java-8-oracle

OpenJDK Runtime Environment (amzn-2.6.5.0.66.amzn1-x86_64 u99-b00)

OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)

Hadoop Version:- hadoop-2.7.2

Spark Version:- spark-1.6.0

ANT version:- Apache Ant 1.9.6

Hadoop Question and answers:-

File descriptions:-

1) core-site.xml:-

The core-site.xml file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.

2) hdfs-site.xml:-

The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes. Here, we can configure hdfs-site.xml to specify default block replication and permission checking on HDFS. The actual number of replications can also be specified when the file is created.

3) mapred-site.xml:-

The mapred-site.xml file contains the configuration settings for MapReduce daemons; the job tracker and the task-trackers.

4) slaves:-

Slaves contain a list of hosts, one per line, that are needed to host DataNode and TaskTracker servers.

5) master:-

The Masters contain a list of hosts, one per line, that are required to host secondary NameNode servers. The Masters file informs about the Secondary NameNode location to Hadoop daemon. The Masters file at Master server contains a hostname, Secondary Name Node servers.

6) hadoop-env.sh:-

In this file we have to specify JAVA_HOME path and setting up of other environment variables for data transfer.

7) /etc/hosts:-

This file contains private ip address and dns of all slaves and master to connect.

Below modifications I have done to go from one node to multi-node cluster:-

1) slaves file:-

In slaves file of slave include dns of that slave and masters dns.

In masters slave file include dns of all slaves and masters dns as well.

2) hosts file:-

In hosts file of slave include private ip address and dns of that slave and masters ip address and dns.

In masters slave file include private ip address and dns of all slaves and masters ip address and dns as well.

In case of spark we don't have to make any modifications to go from one node to multi node.

- What is a Master node?

Master nodes oversee the following key operations that comprise Hadoop: storing data in the Hadoop Distributed File System (HDFS) and running parallel computations on that data using MapReduce. The NameNode coordinates the data storage function (with the HDFS), while the JobTracker oversees and coordinates the parallel processing of data using MapReduce.

The Master (NameNode) manages the file system namespace operations like opening, closing, and renaming files and directories and determines the mapping of blocks to DataNodes along with regulating access to files by clients.

Master (Jobtracker) is the point of interaction between users and the map/reduce framework. When a map/reduce job is submitted, Jobtracker puts it in a queue of pending jobs and executes them on a first-come/first-served basis and then manages the assignment of map and reduce tasks to the tasktrackers.

- What is a Slaves node?

Slave nodes make up the majority of virtual machines and perform the job of storing the data and running computations. Each slave node runs both a DataNode and TaskTracker service that communicates with, and receives instructions from their master nodes. The TaskTracker service is subordinate to the JobTracker, and the DataNode service is subordinate to the NameNode.

Slaves (DataNodes) are responsible for serving read and write requests from the file system's clients along with perform block creation, deletion, and replication upon instruction from the Master (NameNode).

Slaves (task tracker) execute tasks upon instruction from the Master (Job tracker) and also handle data motion between the map and reduce phases.

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

We need unique available ports on shared environment because master node is going to communicate with slaves nodes through separate threads for each slave. Master which will be doing communication on separate port. If all threads starts communicating through one port then there will be problem to track slaves and miscommunication may happen which will lead to failure of system.

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

Below two property tags can be put in mapred-site.xml file to set up number of mappers and reducers in configuration file.

Set Mapper:-

```
<configuration>
<property>
<name>mapred.tasktracker.map.tasks.maximum</name>
<value>4</value>
</property>
</configuration>
```

Set Reducer:-

```
<configuration>
<property>
<name>mapred.tasktracker.reduce.tasks.maximum</name>
<value>4</value>
</property>
</configuration>
```

Performance Analysis:-

Spark and Hadoop comparison:-

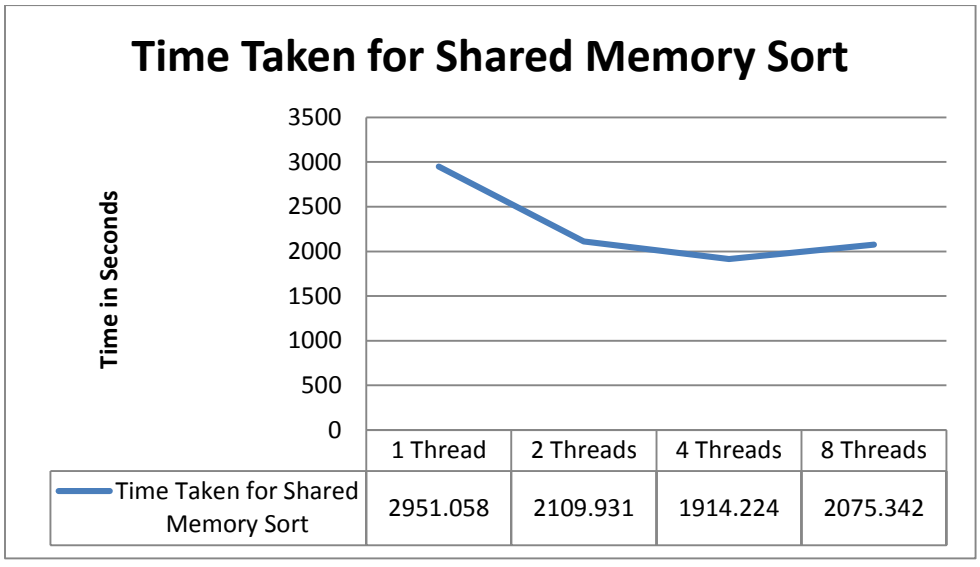
Faster: Spark execute batch processing jobs , about 10 to 100 times faster than the Hadoop MapReduce framework by making the use of in-memory processing compared to persistence storage used by Hadoop.

Spark fast analytics : With Spark, we can use the inbuilt libraries to perform Batch Processing, Streaming, and Machine Learning and Interactive SQL queries in a single cluster unlike Hadoop which only provides Batch Processing at the core.

Caching: One of the reasons of Spark being extremely fast is by making use of caching and in-memory processing and Hadoop on the other hand is completely disk dependent.

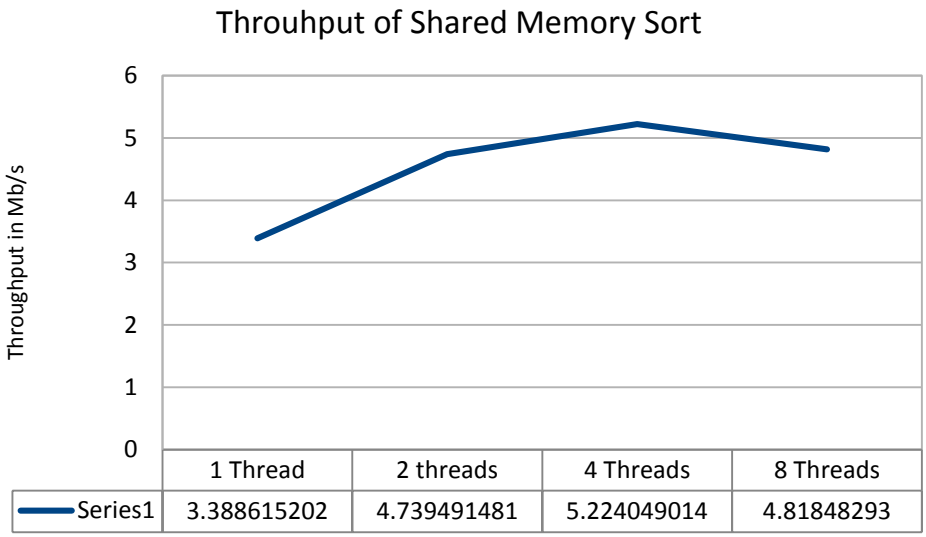
Iterative computations: One of the main aims for building Spark was iterative computations in use cases like machine learning when computation need to be performed multiple times on same set of data.

Shared memory Multi-Thread comparison:-



Time(In seconds) Diagram Descriptions:-

By looking at the graph we can say that time required by 4 Threads is less as compared to time required by 1, 2 and 8 threads simultaneously. Performance is very low at one thread as we increase the number of threads up to 4 we get the best performance. Performance starts decreasing as we increase the threads grater that 4. This is because amazon c3.large instance has 2 virtual cores and 2 main cores so it can handle 4 threads effectively. At 1 thread and 2 simultaneous threads we are not utilizing all its available processors hence getting low performance. At 8 threads there is slight decrease in performance because operating system is spending more time in managing the threads rather than actual execution.



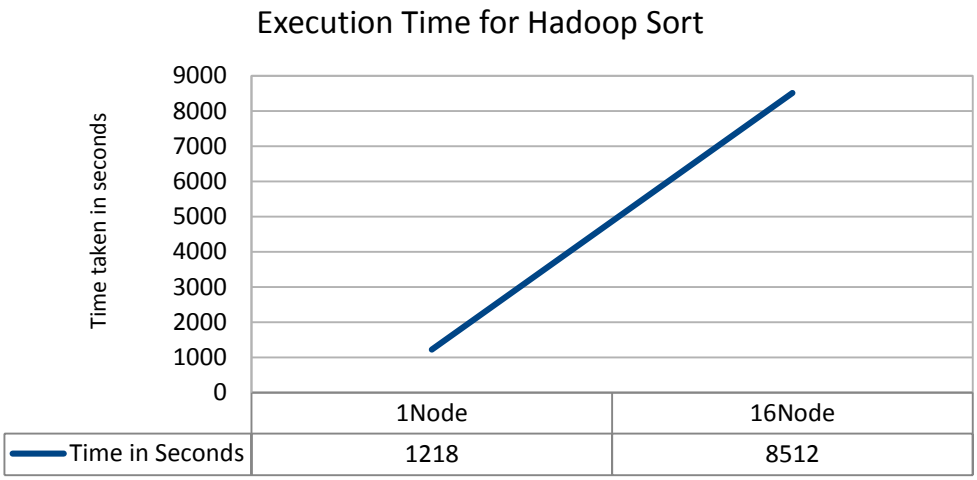
Throughput (Mb/seconds) Diagram Descriptions:-

By looking at the graph we can say that Throughput at 4 Threads is more as compared to Throughput at 1, 2 and 8 threads simultaneously. Throughput is very low at one thread as we increase the number of threads up to 4 we get the best Throughput. Throughput starts decreasing as we increase the threads grater that 4. This is because amazon c3.large instance has 2 virtual cores and 2 main cores so it can handle 4 threads effectively. At 1 thread and 2 simultaneous threads we are not utilizing all its available processors hence getting low performance. At 8 threads there is slight decrease in throughput because operating system is spending more time in managing the threads rather than actual execution.

Hadoop Graphs:-

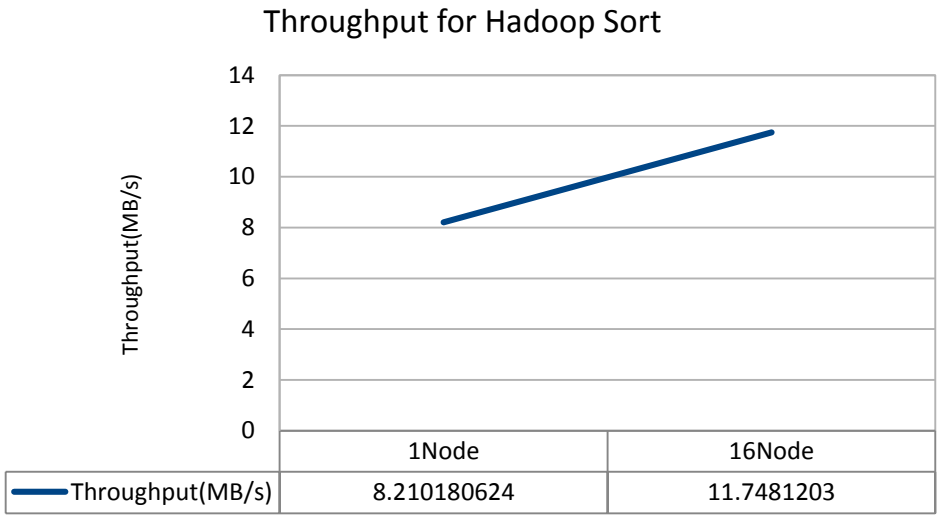
1 Node and 16 Node comparison of Hadoop sort.

Time Required:-



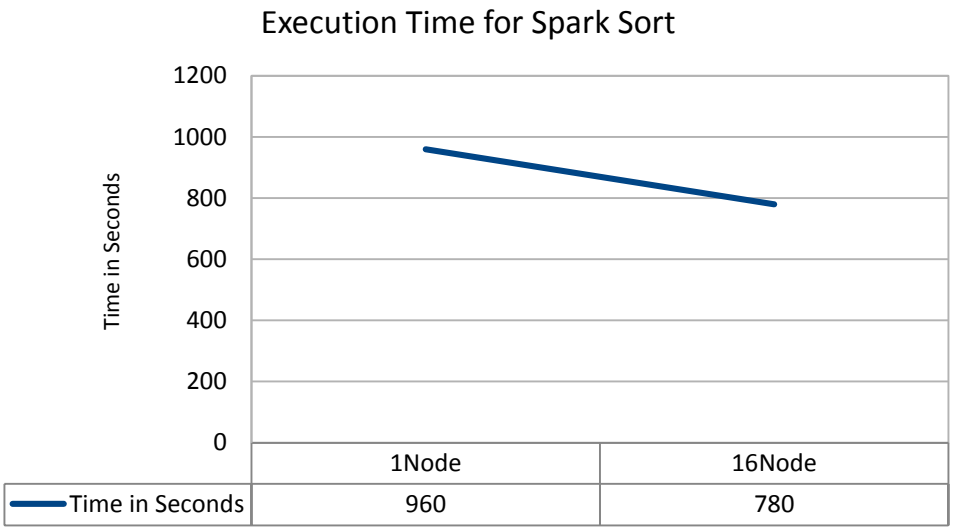
Above diagram shows comparison of time required for Hadoop 1 node to process 10 GB dataset and time required for 16 nodes to process 100 GB dataset.

Hadoop 1 node and 16 nodes throughput:-

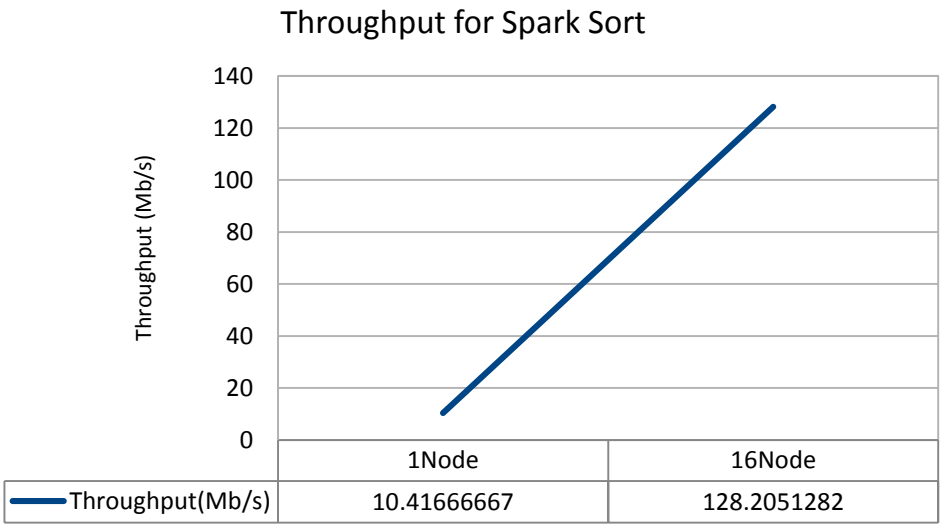


In above diagram we can clearly see that throughput of 16 nodes is greater than 1 node Hadoop execution. Hence we can say that Hadoop performance is better in multi node environment than single node performance.

Spark graphs:-

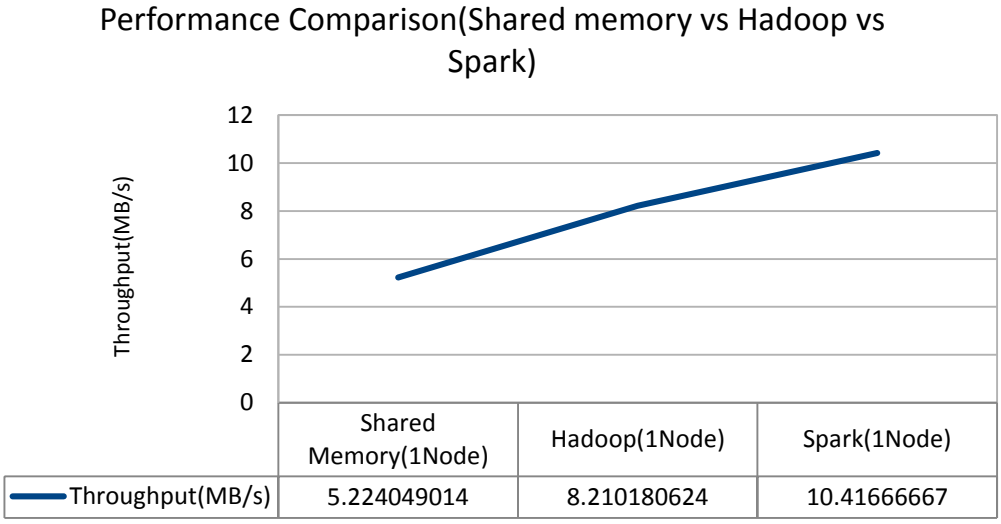


Above diagram shows comparison of time required for spark 1 node to process 10 GB dataset and time required for spark 16 nodes to process 100 GB dataset. As number of nodes increase time to sort goes on decreasing in case if spark.



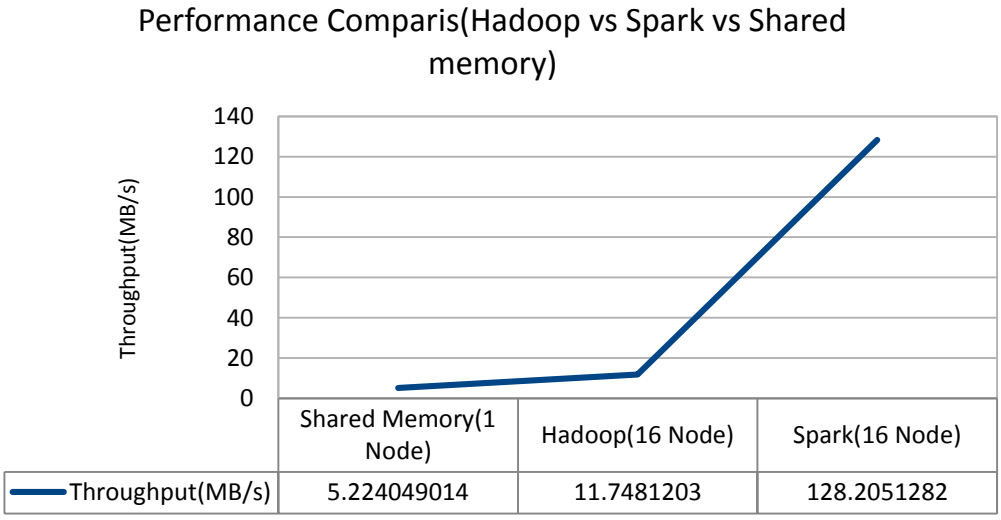
In above diagram we can clearly see that throughput of spark 16 nodes is more than 1 node spark execution. Hence we can say that spark performance is better in multi node environment than single node performance.

Comparison of all three sorts:-



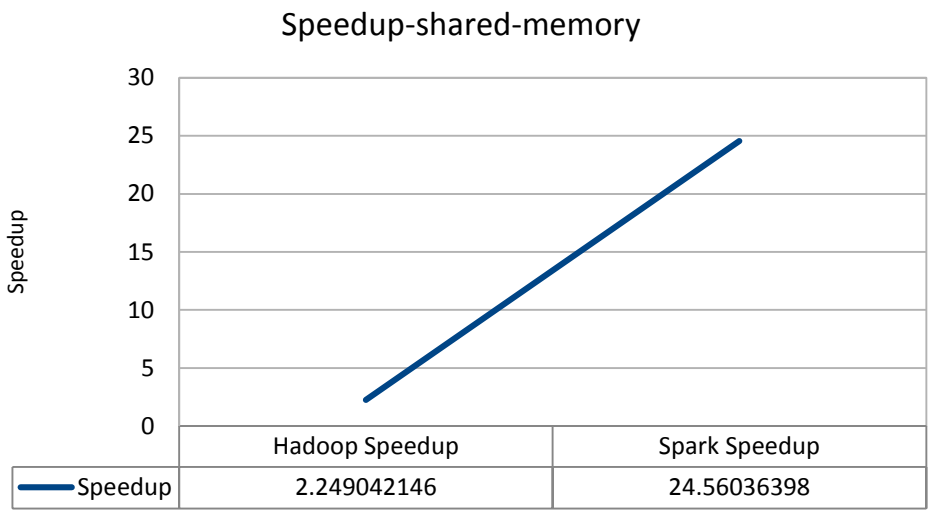
In single node environment spark gives best performance than Hadoop and shared memory. That is because spark uses RDD. RDDs are a 'immutable resilient distributed collection of records' which can be stored in the volatile memory or in a persistent storage (HDFS, HBase etc) and can be converted into another RDD through some of the transformations. An action like count can also be applied on an RDD.

Hence, Throughput of spark is more as compared to Hadoop and shared- memory.

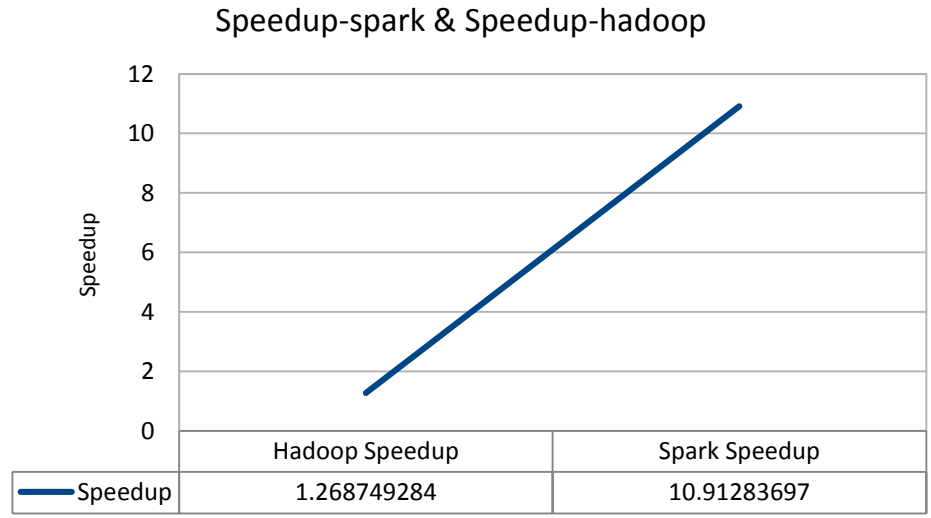


By looking at above diagram we can say that at 16 Node level spark performance is almost 10 times better that that of Hadoop on 16 node scale. This is because spark makes use of RDD's (Resilient Distributed Datasets) to store and process data. That is in main memory hence has better performance.

Speed up charts:-



By looking at above speedup graph compared to shared-memory sort we can say that Hadoop and spark are far better in its performance than that of shared memory sort.



By looking at above speedup chart we can say that spark has good speedup over Hadoop. Spark performance is almost 10 times better than that of Hadoop performance.

Conclusion:-

By looking at all above graphs we can say that spark performance is better than that of Hadoop and shared-memory. Ideally shared-memory's performance at one node should be better because it does not have any startup cost associated with it while in case of Hadoop and spark they have to setup master and slaves communication to manage the work through message exchange.

When we want to process huge data in Gb's and Tb's. We should use spark as it uses RDD's (Resilient Distributed Datasets) to process data and makes effective use of main memory. Spark could be 10 of 100 times faster than Hadoop in long run as it has better architecture of data processing and management.

Which seems to be best at 1 node scale?

- Ideally shared-memory's performance at one node should be better because it does not have any startup cost associated with it while in case of Hadoop and spark they have to setup master and slaves communication to manage the work through message exchange.

How about 16 nodes?

- At 16 nodes scale spark should be better option. As it uses RDD's (Resilient Distributed Datasets) and has better architecture than Hadoop. Spark can be 100 times faster than Hadoop while processing huge data.

Can you predict which would be best at 100 node scale?

- Spark would be better option at 100 Node scale as number of nodes increases. We can get more main memory to store and process data which will provide better performance and failure handling.

How about 1000 node scales?

- Again I will go with spark at 100 node scale. Because from my results I see that spark performance is drastically improving as I move from 1 node to 16 nodes. So we can say that spark will perform better as we increase the nodes even at 1000 node scale.

What can you learn from the CloudSort benchmark ?

- CloudSort benchmark is a new term which makes use of resources available at public cloud to sort data. Through the external sort we can sort huge amount of data in distributed environment. I make use external sort which is representative of many IO-intensive workloads. It's a holistic workload that exercises memory, CPU, OS, file-system, IO, network, and storage. It's simple and therefore easy to port and optimize on cutting-edge technologies.

Winners in 2013 and 2014 who used Hadoop and Spark?

- Daytona gray sort
- Spark is the winner over Hadoop. Spark's performance is better than Hadoop because spark makes efficient use of memory than Hadoop due to it's RDD (Resilient Distributed Datasets) architecture.

Installation steps I took to setup

1) Hadoop Installation steps:-

```
// Execute below steps to Mount EBS Volume
lsblk
sudo file -s /dev/xvdb
sudo mkfs -t ext4 /dev/xvdb
sudo mkdir /sujay
sudo mount /dev/xvdb /sujay
sudo chmod 777 /sujay/

// Execute below steps to install java.
sudo apt-get update
sudo apt-add-repository ppa:webupd8team/java -y
sudo apt-get update
sudo apt-get install oracle-java8-installer -y

// Execute below steps for Ant installation
sudo apt-get install ant
sudo apt-get update

// Execute below steps for GCC installation
sudo apt-get install gcc -y
sudo apt-get update -y

//Open bashrc file and add the below details to the file.
vi .bashrc

export CONF=/home/ubuntu/hadoop-2.7.2/etc/hadoop
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export PATH=$PATH:$/home/ubuntu/hadoop-2.7.2/bin

//Copy code to TeraSort_Hadoop.java and save under hadoop-2.7.2 directory.
vi TeraSort_Hadoop.java

/// setup below veriables to compile TeraSort_Hadoop.java file.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

// Copy and install Hadoop
wget http://www-us.apache.org/dist/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz
tar -xzvf hadoop-2.7.2.tar.gz
chmod 777 hadoop-2.7.2

// Install ssh
sudo apt-get install ssh -y

/// Compile sort progogram
bin/hadoop com.sun.tools.javac.Main TeraSort_Hadoop.java
jar cf TeraSort_Hadoop.jar *.class

// change core-site.xml, hadoop-env.sh, hdfs-site.xml, yarn-site.xml and mapred-site.xml as per below edit DNS of master in below files.
cd /home/ubuntu/hadoop-2.7.2/etc/Hadoop

/// Change Slaves and hosts file as per below.

cd hadoop2.7.2/etc/hadoop
vi slaves

///slave node will have masters and slaves node dns in slaves file.
///master node will have all slaves and master node dns in slaves file.

///for example masters slaves file:-
ec2-52-35-173-51.us-west-2.compute.amazonaws.com
ec2-52-33-164-101.us-west-2.compute.amazonaws.com
ec2-52-36-17-183.us-west-2.compute.amazonaws.com
ec2-52-37-129-190.us-west-2.compute.amazonaws.com
ec2-52-36-247-207.us-west-2.compute.amazonaws.com

//slave1:-
slaves file:-
ec2-52-35-173-51.us-west-2.compute.amazonaws.com
ec2-52-33-164-101.us-west-2.compute.amazonaws.com

sudo vi /etc/hosts
//slave node will have masters and slaves node private ip and dns in hosts file.
//master node will have all slaves and master node private ip and dns in hosts file.

//for example masters hosts file:-
172.31.5.141 ec2-52-35-173-51.us-west-2.compute.amazonaws.com
172.31.2.162 ec2-52-33-164-101.us-west-2.compute.amazonaws.com
172.31.15.215 ec2-52-36-17-183.us-west-2.compute.amazonaws.com
172.31.11.249 ec2-52-37-129-190.us-west-2.compute.amazonaws.com
172.31.5.181 ec2-52-36-247-207.us-west-2.compute.amazonaws.com

//hosts file
172.31.5.141 ec2-52-35-173-51.us-west-2.compute.amazonaws.com
172.31.2.162 ec2-52-33-164-101.us-west-2.compute.amazonaws.com

// Execute below six steps on all nodes.
eval "$(ssh-agent)"
chmod 400 hadoop.pem
ssh-add hadoop.pem
ssh-keygen -t rsa

//provide the masters DNS on all the slaves machine while executing below command.
ssh-copy-id -i ~/.ssh/id_rsa.pub ubuntu@ec2-52-35-173-51.us-west-2.compute.amazonaws.com
chmod 0600 ~/.ssh/authorized_keys

// execute below commands to start hadoop.
cd hadoop-2.7.2/
bin/hadoop namenode -format
ssh localhost
cd sbin
./start-dfs.sh
./start-yarn.sh

////Execute jps command to check if all name nodes and data nodes are up and running.
jps

// Aftre jps command you should see below processess running on master.
4963 Jps
4510 ResourceManager
3714 DataNode
4139 NameNode
4655 NodeManager

// Create and setup hdfs directory
cd
cd hadoop-2.7.2
bin/hadoop fs -mkdir /hdfs_sujay

cd /sujay
mkdir data
cd data

/// Download gensort application and generate file of 100Gb and copy that file to hdfs
cd /data
wget www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
tar -xzvf gensort-linux-1.5.tar.gz
cd 64

./gensort -a 1000000000 100GB_input_file
cd
cd hadoop-2.7.2
bin/hadoop dfs -copyFromLocal /sujay/data/64/100GB_input_file /hdfs_sujay
bin/hadoop dfs -ls /hdfs_sujay/
bin/hadoop dfs -rm -r -f /hdfs_sujay/output

/// Execute hadoop program to sort 100GB file.
bin/hadoop jar TeraSort_Hadoop.jar TeraSort_Hadoop /hdfs_sujay/100GB_input_file /hdfs_sujay/output

///Command to copy the file from hadoop file system to our instance EC2.
bin/hadoop dfs -get /hdfs_sujay/output/part-r-00000 /sujay

// Execute below command to check if file is properly sorted or not
cd /sujay/data/64
./valsort /sujay/part-r-00000

2) spark installation steps:-

//Download spark on your machine.
wget www-eu.apache.org/dist/spark/spark-1.6.0/spark-1.6.0-bin-hadoop2.6.tgz
tar -xzvf spark-1.6.0-bin-hadoop2.6.tgz
cd spark-1.6.0-bin-hadoop2.6
cd ec2

//create security key on your machine and copy that in below commands.
export AWS_ACCESS_KEY_ID=AKIAIOTD7BCLNDQGVJSA
export AWS_SECRET_ACCESS_KEY=lIpFl5JyySzWzX7FA21H2K7QIy0hbOkFni1c4Ls/

// To launch one spark slave and master instance execute below command.
./spark-ec2 -k sparkeast -i /home/sujay/Desktop/PA2/sparkeast.pem -s 1 -t c3.large --spot-price=0.03 launch spark

// To launch 16 spark slaves and one master instance execute below command.
./spark-ec2 -k sparkeast -i /home/sujay/Desktop/PA2/sparkeast.pem -s 16 -t c3.large --spot-price=0.03 launch spark

// Login to spark master instance.
./spark-ec2 -k sparkeast -i /home/sujay/Desktop/PA2/sparkeast.pem login spark

//mount 400GB volume to master.

lsblk
sudo file -s /dev/xvds
sudo mkfs -t ext4 /dev/xvds
sudo mkdir /sujay
sudo mount /dev/xvds /sujay

```
sudo chmod 777 /sujay/
cd /sujay
//Download gensort and create 100GB input file.

wget www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
tar -xzvf gensort-linux-1.5.tar.gz
cd 64
./gensort -a 1000000000 100GB_input_file

//Create directory in hdfs
cd
cd ephemeral-hdfs
bin/hadoop fs -mkdir /sujay
// copy 100 Gb input file to hdfs.
bin/hadoop fs -Ddfs.replication=1 -put /sujay/64/100GB_input_file /sujay
// Check the content of hdfs.
bin/hadoop dfs -ls /sujay/
// Create output directory in hdfs.


// Create scala script to sort file.
cd spark
cd bin
vi sujay_spark_sort.scala

//Copy below below script to sujay_spark_sort.scala file

/** Start timer */
val timer_start = System.currentTimeMillis()
/** Give input file name  */
val file_input=sc.textFile("hdfs:////sujay/100GB_input_file")
/** seperate key and value pair  */
val file_to_sort=file_input.map(line => (line.take(10), line.drop(10)))
/** Sort the input by key  */
val sort = file_to_sort.sortByKey()
/** Map key and value  */
val lines=sort.map {case (key,value) => s"$key $value"}
/** Set the output file location  */
lines.saveAsTextFile("/sujay/spark_sorted_100GB_output")
/** Get the end time  */
val timer_end = System.currentTimeMillis()
/** print total time of execution */
println ("Time taken to sort file :-" + (timer_end - timer_start) + "In Milliseconds")

bin/hadoop fs -chmod 777 /tmp/hive
./spark-shell -i sujay_spark_sort.scala

//Copy file from hdfs to other location.
```

```
bin/hadoop dfs -getmerge /sujay/spark_sorted_100GB_output /sujay/100GB_final_spark_sorted_file
```