CS553 Programming Assignment #2
Sort on Hadoop/Spark
A20370450
Vickyben Patel

This programming assignment covers the Sort application implemented in 3 different ways:

1) Using normal programming language with multithread.
2) Using Apache Hadoop.
3) Using Spark

I need to perform this experiment on Amazon EC2 – c3.large instance with a cluster of 1 and 16 nodes. I also need to compare the results in terms of time and speed-up for all three approaches.

VIRTUAL CLUSTER (1- NODE)

For this assignment, I first created a single node virtual cluster on Amazon EC2. The steps followed were as below:

Go to amazon AWS link
Sign up with credentials
Then go to Services and Select EC2
Click on Instances
Click on Launch Instance
Select your operating system
Choose H/W or instance type as per the requirement (Used c3.large) '
c3.large- two virtual CPU and 3.75 RAM and 2*16 GB SSD
Select spot instances and click on Review and launch

Click on Step 6: Configure Security group
The current type is SSH, protocol is TCP on port 22 and source can be from anywhere
So if you will ping it it will not respond so in order to enable it click on Add rule and select ALL ICMP and ALL TCP

Make its source as "Anywhere" and then click on Review and Launch

Next important thing to do is Storage and for that go to "4. Add Storage" and change the size as per the requirement

Next step is to click on Launch
A pop will be displayed in which you have to enter the key value pair and click on "Download key pair"

Click on Launch Instances

Then click on Spot requests and you will see the spot request which we have created
Then go to instances tab you will see your instance
In the instances tab click on Connect button a pop up will be displayed

AMI and instance details:

AMI - Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-fce3c696
Instance Type – c3.large

Shared-Memory Sort

I have developed the shared memory program in JAVA. To work on big file efficiently, I split the files into number of chunks so that it becomes possible to process them in memory.
shared memory is memory that may be simultaneously accessed by multiple programs with an intent to provide communication among them or avoid redundant copies. Shared memory is an efficient means of passing data between programs.
Time taken = 2400 sec

VIRTUAL -CLUSTER (16- NODES)

For the 16 node cluster, we used the same instance we created in the -VIRTUAL CLUSTER (1-NODE). We saved our instance as an image to launch 17 instances as cluster
Update the config file with correct details of access keys, AMIs and worker details.
Run the setup.
Launched 17 instances – 1 head-node and 16 worker nodes.
Able to connect to the head-node and all the workers from Launchpad, so the cluster setting was successful.

AMI and instance details:

AMI – Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-fce3c696
Instance Type: c3.large

Hadoop

map()= is called on every item in the input and emits a series of intermediate key/value pairs.
    Iterate over a large number of records.
    Extract something a large number of records
    Shuffle and sort intermediate results

reduce()= is called on every unique key, and its value list, and emits a value that is added to the output.
    Aggregate intermediate results generate final output
    Generate final output

To set up Hadoop Cluster, we first set up Hadoop on a single node. Since Java is required for Hadoop, I first installed Java.

Version details:

Download **JAVA 8**

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Download **hadoop-2.7.2.tar.gz**

```
wget http://download.nextag.com/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz
tar -xvf hadoop-2.7.2.tar.gz
```

Configure the ~/.bashrc file.

"vi ~/.bashrc"

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export CONF=/home/ubuntu/PA2/hadoop-2.7.2/etc/hadoop
export PATH=$PATH:$/home/ubuntu/PA2/hadoop-2.7.2/bin
```

Set directory "hadoop-2.7.2/etc/hadoop"

Configure 'hadoop-env.sh':

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Configure 'core-site.xml'

```
<configuration>
<property>
   <name>fs.default.name</name>
   <value>hdfs://ec2-52-87-163-11.compute-1.amazonaws.com:8020</value>
</property>
</configuration>
```

Configure 'hdfs-site.xml'

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

Configure 'mapred-site.xml'

```
<configuration>
<property>
<name>mapreduce.jobtracker.address</name>
<value>hdfs://ec2-52-87-163-11.compute-1.amazonaws.com:8020</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>
        <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
        <description>The framework for running mapreduce jobs</description>
        </property>
        <property>
        <name>mapreduce.job.reduces</name>
        <value>4</value>
        </property></configuration>
```

Configure 'yarn-site.xml'

```
        <configuration>
        <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
        </property>
        <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>ec2-52-87-163-11.compute-1.amazonaws.com:8020</value>
        </property>
        <property>
        <name>yarn.resourcemanager.address</name>
        <value>ec2-52-87-163-11.compute-1.amazonaws.com:8020</value>
        </property>
        <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>ec2-52-87-163-11.compute-1.amazonaws.com:8020</value>
        </property>
        <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>ec2-52-87-163-11.compute-1.amazonaws.com:8020</value>
        </property>
        <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>ec2-52-87-163-11.compute-1.amazonaws.com:8020</value>
        </property>
        </configuration>
```

'slaves'

```
         ec2-52-87-163-11.compute-1.amazonaws.com
```

Runtime steps:

```
        hadoop/bin/hdfs namenode -format
        hadoop/sbin/start-dfs.sh
```

```
hadoop/sbin/start-yarn.sh
jps (to check if all nodes are running.)
```

16 Nodes Setup:

Just as normal cluster, to set up a hadoop cluster of 16 instances, I used the image created from instance on which I installed hadoop in previous part. Able to connect to slave nodes from head-node with ssh, so cluster was successful.

'Slaves'

```
ec2-54-85-40-32.compute-1.amazonaws.com
ec2-52-87-231-120.compute-1.amazonaws.com
ec2-54-164-78-231.compute-1.amazonaws.com
ec2-52-91-160-106.compute-1.amazonaws.com
ec2-52-91-154-243.compute-1.amazonaws.com
ec2-52-207-207-253.compute-1.amazonaws.com
ec2-54-152-254-177.compute-1.amazonaws.com
ec2-52-3-252-29.compute-1.amazonaws.com
ec2-52-87-231-192.compute-1.amazonaws.com
ec2-54-152-232-73.compute-1.amazonaws.com
ec2-52-87-155-38.compute-1.amazonaws.com
ec2-54-86-178-94.compute-1.amazonaws.com
ec2-54-152-232-143.compute-1.amazonaws.com
ec2-52-90-22-111.compute-1.amazonaws.com
ec2-52-91-137-14.compute-1.amazonaws.com
ec2-54-152-53-120.compute-1.amazonaws.com
```

HADOOP SORT

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

The Master nodes oversee the two key functional pieces that make up Hadoop:

Storing lots of data (HDFS), and

Running parallel computations on all that data (Map Reduce).

The Name Node oversees and coordinates the data storage function (HDFS), while the Job Tracker oversees and coordinates the parallel processing of data using Map Reduce.

Slave Nodes make up the vast majority of machines and do all the work of storing the data and running the computations. Each slave runs both a Data Node and Task Tracker daemon that communicate with and receive instructions from their master nodes. The Task Tracker daemon is a slave to the Job Tracker, the Data Node daemon a slave to the Name Node.

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?

ports are assigned for different usages, different functions. Port is set for Master node, HDFS, Jobtrackers separately. There would be port collision between different function modules if we use same port number.

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

file?
You can change the total amount of mapper and reducer by editing the configuration files “conf/mapred-site.xml”
set it as same as the amount of cores (cpu).

```
<property>
        <name>mapreduce.job.maps</name>
        <value>2</value>
</property>
<property>
      <name>mapreduce.job.reduces</name>
      <value>2</value>
</property>
```

1 NODE

Time taken : 1682 sec

16 Node

Time taken : 7210 sec

Spark

Download **spark-1.6.1-bin-hadoop2.6**

create new access key

export AWSAWSAccessKeyId and AAWSSecretKey

```
        export AWS_ACCESS_KEY_ID=AKIAIKRB5CM2Y5XOPHWA
        export
AWS_SECRET_ACCESS_KEY=AWSSecretKey=Jfnrp921qn7nhL5BMXfNXQ/Rbh5VeKaAoXZIPJ
m/
```

```
./spark-ec2 -k spark -i /root/spark.pem -s 1 -t c3.large -r us-east-1 --spot-price=0.027 launch Spark
```

Download gensort-linux-1.5.tar.gz

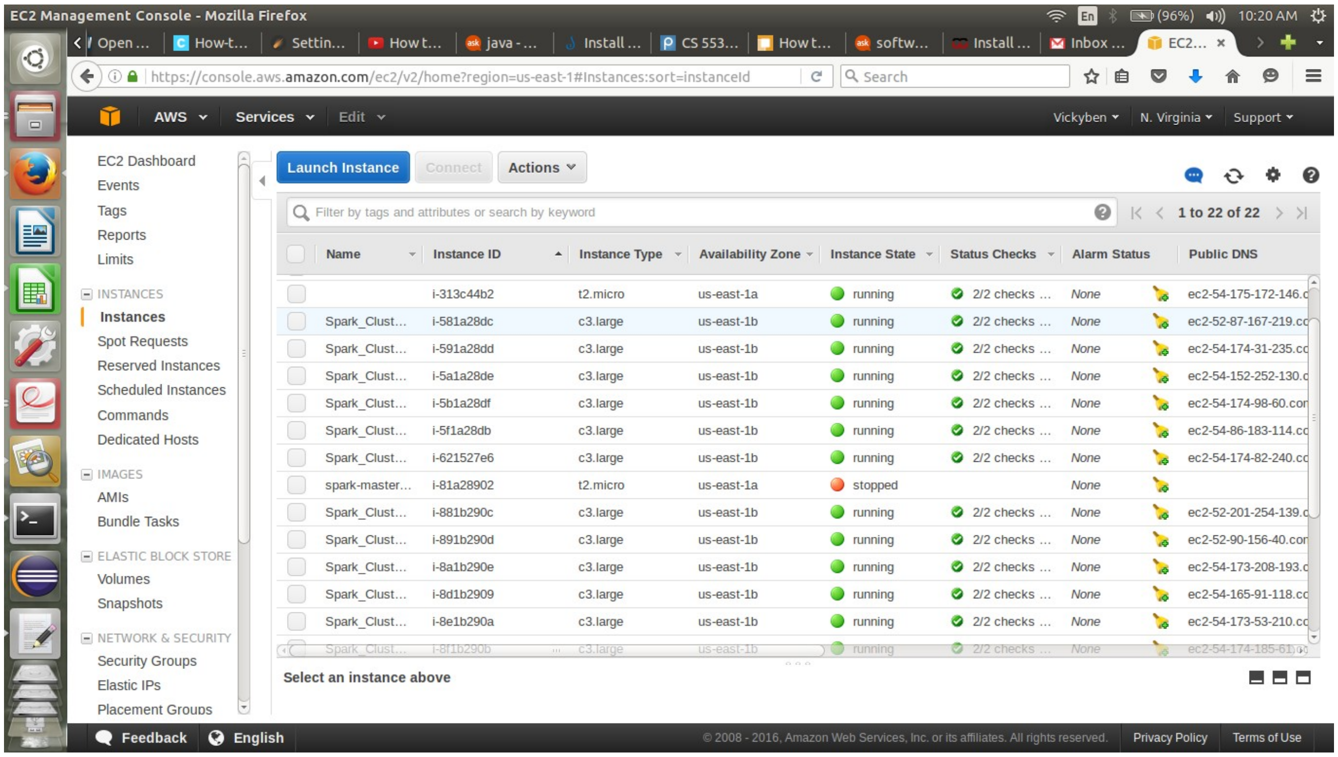
```
        wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
        tar xvf gensort-linux-1.5.tar.gz
```

masters
ec2-52-201-212-65.compute-1.amazonaws.com

slaves
ec2-54-85-40-32.compute-1.amazonaws.com

16 Nodes setup

```
./spark-ec2 -k spark -i /root/spark.pem -s 16 -t c3.large -r us-east-1 --spot-price=0.027 launch Spark
```



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-313c44b2	t2.micro	us-east-1a	running	2/2 checks ...	None	ec2-54-175-172-146.c
Spark_Clust...	i-581a28dc	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-52-87-167-219.cc
Spark_Clust...	i-591a28dd	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-174-31-235.cc
Spark_Clust...	i-5a1a28de	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-152-252-130.c
Spark_Clust...	i-5b1a28df	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-174-98-60.cor
Spark_Clust...	i-5f1a28db	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-86-183-114.cc
Spark_Clust...	i-621527e6	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-174-82-240.cc
spark-master...	i-81a28902	t2.micro	us-east-1a	stopped		None	
Spark_Clust...	i-881b290c	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-52-201-254-139.c
Spark_Clust...	i-891b290d	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-52-90-156-40.cor
Spark_Clust...	i-8a1b290e	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-173-208-193.c
Spark_Clust...	i-8d1b2909	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-165-91-118.cc
Spark_Clust...	i-8e1b290a	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-173-53-210.cc
Spark_Clust...	i-8f1b290b	c3.large	us-east-1b	running	2/2 checks ...	None	ec2-54-174-185-61

Select an instance above

Spark Sort

1 Node

Time taken : 868 sec

16 Node

Time taken : 1200 sec

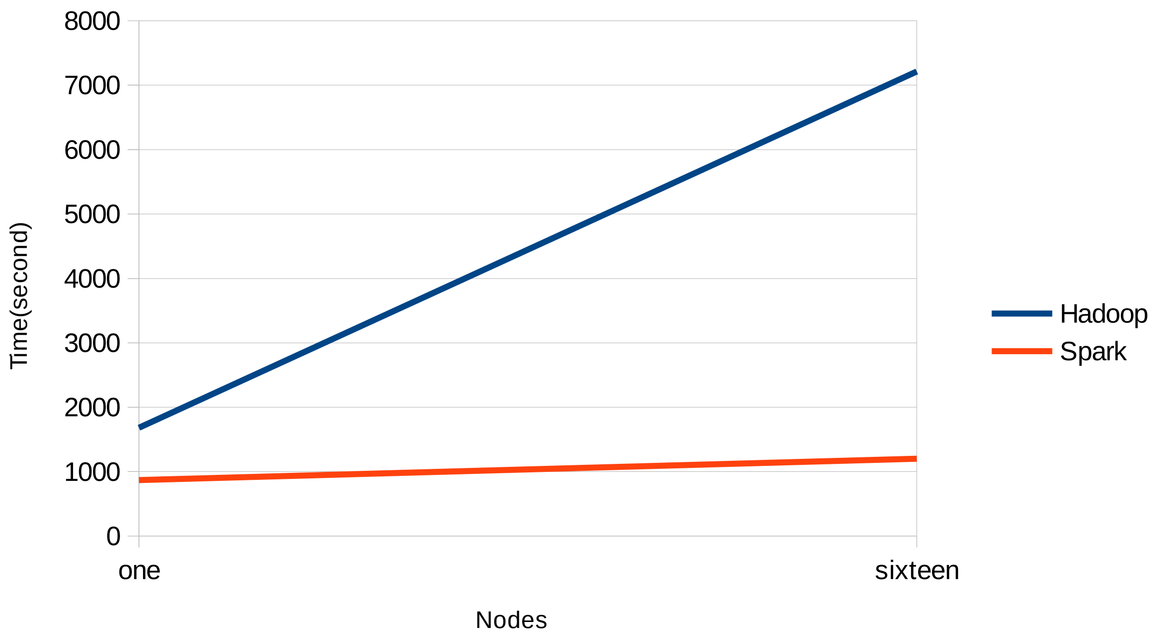
All screenshorts can be found in screenshorts file.

Source code can be found in source_code folder

Performance:

Execution line chart for Hadoop and Spark on 1 node and 16 node

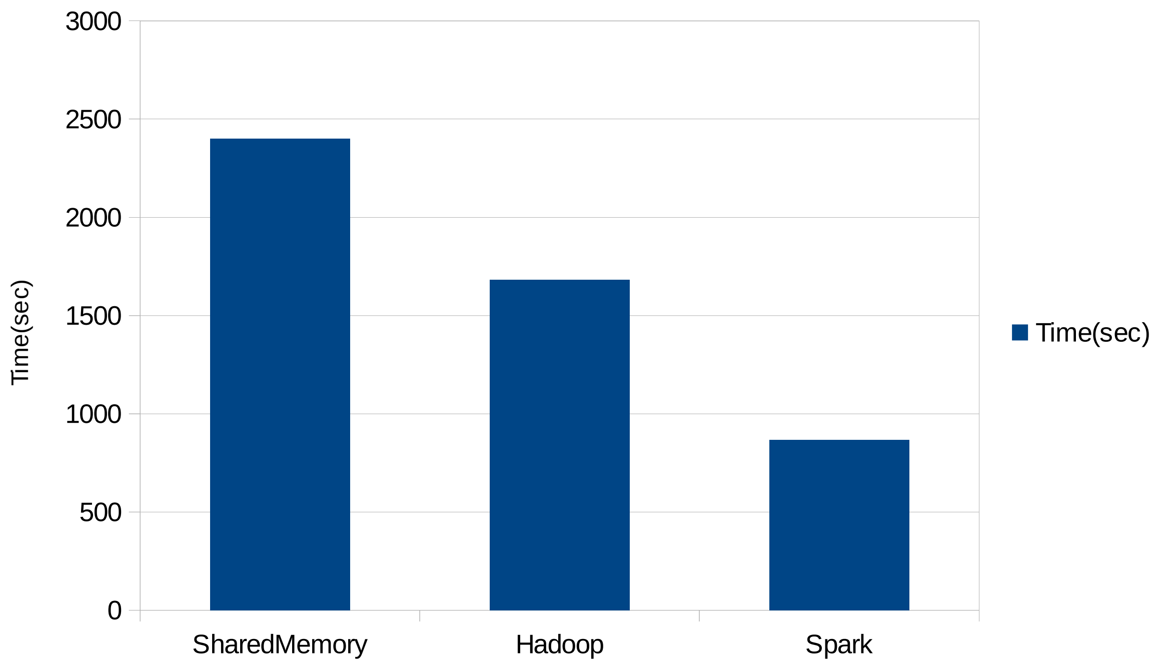
Nodes	Hadoop	Spark
1	1682	868
16	7210	1200



For 1 node there is not more difference in execution time of Hadoop and Spark but as nodes increases Spark takes less execution time and Hadoop takes more because Hadoop stores data on disc and Spark stores data in-memory. Spark makes it possible by reducing number of read/write to disc. It stores this intermediate processing data in-memory. This helps to reduce most of the disc read and write – the main time consuming factors – of data processing. So It is best for multiple nodes.

Compare the performance of the three versions of Sort (Shared-Memory, Hadoop, and Spark) on 1 node scale

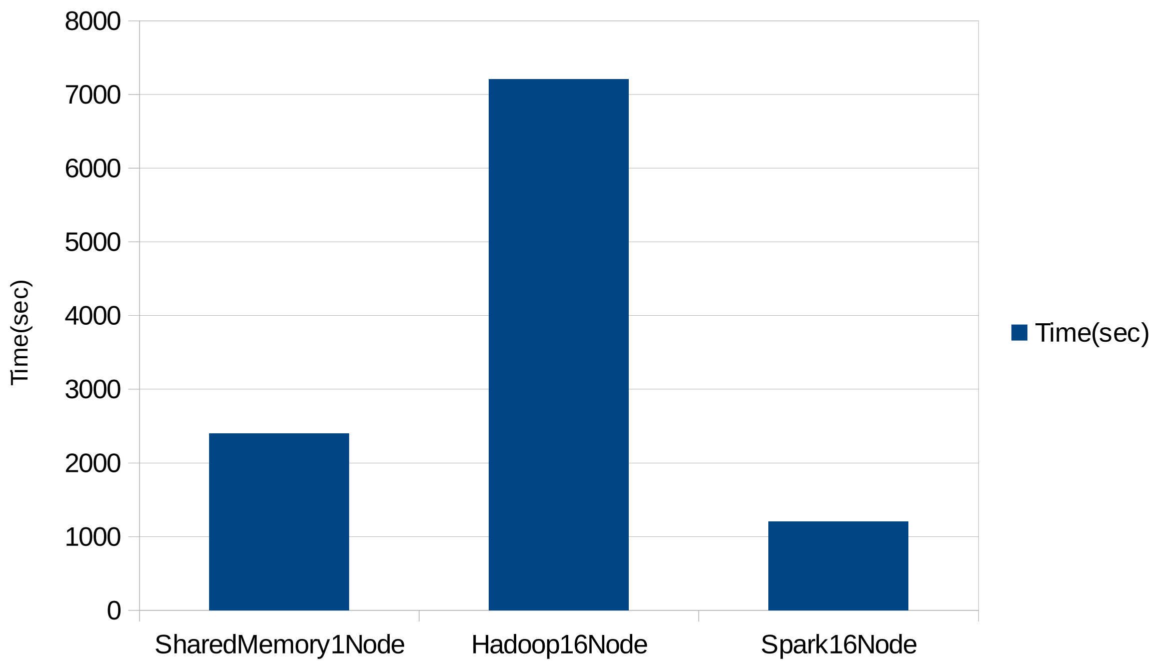
	Time(sec)
Shared-Memory	2400
Hadoop	1682
Spark	868



From chart we can see that SharedMemory sort takes more time compare to Hadoop sort and Spark sort because Hadoop and spark are parallel data processing framework so they takes less time.

compare the Shared-Memory performance of 1 node to Hadoop and Spark Sort at 16 node scales

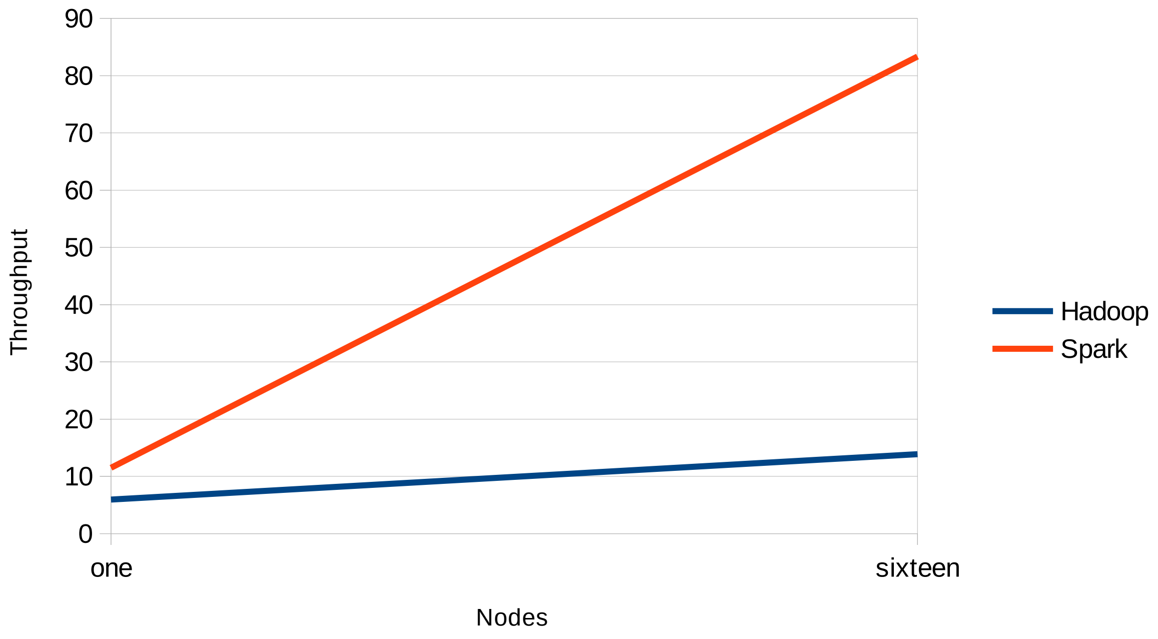
Shared-Memory1Node	2400
Hadoop16Node	7210
Spark16Node	1200



Hadoop takes more time.

Speedup

Node	Hadoop Throughput	Spark Throughput
1	5.94	11.52
16	13.87	83..33



Spark throughput is very high compare to Hadoop throughput from 1 node to 16 node. As Spark execution time is less for 16 node compare to Hadoop.
Speedup in throughput for hadoop=2.33
Speedup in throughput for spark=7.23
Speedup Hadoop=0.233
Speedup Spark=0.7233

What conclusions can you draw? Which seems to be best at 1 node scale? How about 16 nodes? Can you predict which would be best at 100 node scale? How about 1000 node scales?

I compared the results of all experiments and concluded that at 1 node scale Hadoop can be efficient. At 16 nodes Spark seems to be best.
I can predict that spark would be best at 100 node scale and 1000 node scale.
As Spark stores data in-memory. Spark makes it possible by reducing number of read/write to disc. It stores this intermediate processing data in-memory. This helps to reduce most of the disc read and write – the main time consuming factors – of data processing. So It is best for multiple nodes.
The MapReduce workflow looks like this: read data from the cluster, perform an operation, write results to the cluster, read updated data from the cluster, perform next operation, write next results to the cluster, etc. Spark, on the other hand, completes the full data analytics operations in-memory and in near real-time: "Read data from the cluster, perform all of the requisite analytic operations, write results to the cluster so Spark is faster than Hadoop.

After working on this assignment, I learnt many new things. I learnt how to setup 16 nodes on Amazon EC2, learnt about Hadoop and Spark, what is the difference between hadoop and spark, which one is more efficient and also learnt about hdfs file system and many other things.