CS 553: CLOUD COMPUTING

Design Document for Programming Assignment #2

Sort on Shared-Memory/Hadoop/Spark

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. Java (shared-memory sort)
2. Hadoop
3. Spark

Then I have compared the execution time and throughput in section

4. Comparison between Shared memory/Hadoop/Spark Sort

In last section I have written all question and Answer of PA-2

5. PA-2 Question and Answers

Java (Shared-memory sort)

Problem Statement

To write a java programs that sort dataset of size 1GB and 1TB on d2.xlarge Dense-storage instance. As we have data size much bigger than memory of d2.xlarge 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 in Input Stream because file size is bigger than memory so that it should not run out of memory
2. Divide big file on number of lines or total number of file chunks to create. We also need to keep in mind that we should not create too many files like 2000 because operating system and java have limit of how many files we can open at once. As well as we also need to keep in mind that one file size should not be more than memory size.
3. While dividing the files I have used quick sort to sort those chunk files. Sorting is done by another thread that is responsible for sorting data and storing data in file system.
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 b)K way-

merging. I used k-way merging approach in my program 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 sort the data.
6. Store the output of k-way merging in a file at disk.

Configuration and Experiment

Configuration

EC2 Instance:

Model	vCPU	Mem (GB)	Storage (GB)
D2.xlarge	4	30.5	8000

Created raid0 at /mnt/raid location (Script is written in setup file)

Kernel Version: GNU/Linux

OS version:
NAME="Ubuntu"
VERSION="14.04.3 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.3 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL=http://bugs.launchpad.net/ubuntu/

Java Version : 1.7.0.95 openJDK

Solving 1GB dataset:

1GB data set is very small as we have memory size much bigger than data set size. We can sort this data in memory as well as by splitting it on multiple files.

I used multiple files approach where

I have 10 files each of 100mb size. Total time it took to sort using my java program is 64.23 sec

I have 100 files each of 10mb size. Total time it took to sort using my java program is 51.97 sec

Solving 1TB dataset:

Solving 1TB data set took me lots of effort and hit-trial on running 100GB 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.

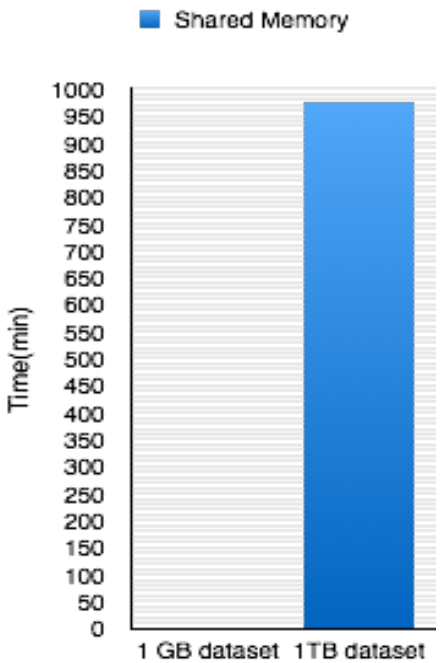
To sort data set of size 1TB, I created 1000 files of 1GB each. It took 16.21 hours to sort. I ran the code with 2 threads working on queue of max size of 4,so that our code does not go out of memory as in worst case we will be having 6 GB of data inside memory plus Java own objects like string and others.

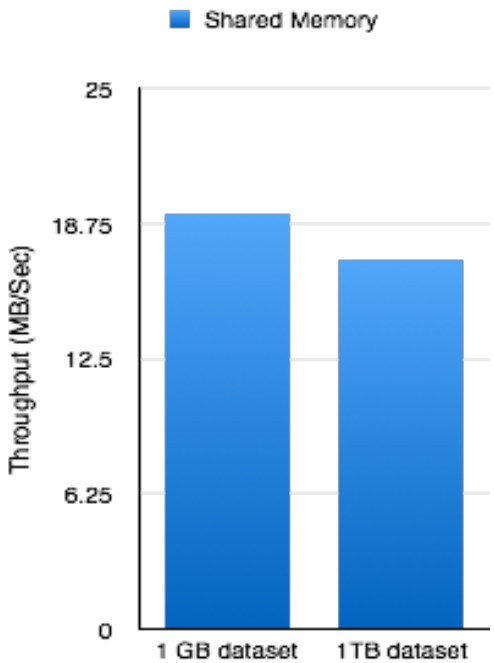
Graph and Explanation

Data Set: 1GB and 1TB
Number of Nodes: 1

Shared memory sort	1 GB dataset	1TB dataset
Time (min)	0.866	972.6

Shared memory Sort	1 GB dataset	1TB dataset
Throughput (MB/sec)	19.24	17.13





Explanation

As we can see from the above graph through put for my java program to sort data as not changed by big margin from very small data set to large data set.

Reason behind this is by carefully choosing the number of threads and Max number of data that we can store in queue as my approach uses producer and consumer. For 1GB data set I have used 100 threads and queue size max capacity to 150. But scenario for 1 TB is not same so I used 2 threads working on queue of max size of 4 because data sizes of file chunk are of 1GB. Though my program is well suited in terms of through put for single node but on distributed system it will lack performance due to network bandwidth and latency also lot of modification will be required to run my code in distributed system.

In conclusion I think my code for running shared-memory sort has given me a good result on one node.

Problem faced

JVM out of heap space: for this issue, I changed the JVM heap size

GC overhead exceeded: For this issue I used faster garbage collection algorithm

Too many files opened exception: For this I have created 1000 files.

Carefully considering number of files to generate and number of thread to use because there are lots of chances to get above exceptions.

Improvements and extensions

To solve the shared-memory sort problem I went through numerous blogs and followed their ways of implementation, which I thought is good for that large dataset. However there are few ways we can still increase the performance of sort. One way I think of is to firstly divide the file in number of chunks without sorting, as done my HDFS file system then sort those files and merge them. In this approach out sort time will start after file chunk creation so we can skip few I/O cost of file chunk generation.
To run my code on distributed system will need lots of modification because this code is designed to use multiple threads and we need to also consider the fact of network while running the code in distributed environment.

.

Hadoop Sort

Problem Statement

In this part of assignment, i have written a "java program" to sort data set of 1GB, 10GB, and 100GB as well as tried for 1TB (not successful due to AWS credits) on single and 17 nodes hadoop cluster. This experiment aims to explore hadoop configuration, installation and learn Hadoop API use in program.

Approach

This problem mainly deals with configuration of hadoop cluster that I 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.

Configuration and Experiment

Configuration

EC2 Instance:

Model	vCPU	Mem (GB)	Storage (GB)
D2.xlarge	2	3.75	400 (EBS)

Created raid0 at /mnt/raid location (Script is written in setup file)

Kernel Version: GNU/Linux

OS version:

NAME="Amazon Linux AMI"
VERSION="2016.03"
ID="amzn"
ID_LIKE="rhel fedora"
VERSION_ID="2016.03"
PRETTY_NAME="Amazon Linux AMI 2016.03"
ANSI_COLOR="0;33"
CPE_NAME="cpe:/o:amazon:linux:2016.03:ga"
HOME_URL="http://aws.amazon.com/amazon-linux-ami/"

Java Version : 1.7.0.95 openJDK
Hadoop Version: hadoop-2.7.2

Running Experiment on Single node

Hadoop Single node setup**:** To setup single node of Hadoop we need to do following configuration changes written below in conf file.

Core-site.xml

```
<configuration>
        <property>
                <name>fs.default.name</name>
                <value>hdfs://ec2-** compute.amazonaws.com:9000</value>
        </property>
</configuration>
```

Change the value tag in above property to master node instance name

hdfs-site.xml

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

Change the value tag in above property to point data directory of hadoop used for hdfs storage

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

Change the value tag in above property to point node directory of hadoop

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

Change the value tag in above property for number of replication to use

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

mapred-site.xml

```
<property>
        <name>mapreduce.job.tracker</name>
        <value>hdfs:/ec2-54-218-32-87.us-west-2.compute.amazonaws.com:8001</value>
</property>
```

Change the value tag in above property to master node instance name

yarn-site.xml

```
<property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>ec2-54-218-32-87.us-west-2.compute.amazonaws.com:8025</value>
</property>
<property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>ec2-54-218-32-87.us-west-2.compute.amazonaws.com:8035</value>
```

```
</property>
<property>
        <name>yarn.resourcemanager.address</name>
        <value>ec2-54-218-32-87.us-west-2.compute.amazonaws.com:8050</value>
</property>
```

Change the value tag in above property to master node instance name.

Conf/slave

Change this file default local host to aws instance name.

After doing the above configuration changes we need to format hdfs

Running Hadoop

./bin/hdfs namenode –format

then we need to start dfs ./sbin/start-dfs.sh

and then we need to start yarn ./sbin/start-yarn.sh

./bin/hadoop fs –mkdir /input

./bin/hadoop fs –put /vol0/input.txt /input

For single node, I have run experiment for 1GB and 10 GB dataset.

For 1GB dataset my sort program run for 1 min 30 sec.

For 10GB dataset my sort program run for 15 min.

Running Experiment on Multiple nodes

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

Mapred-site.xml

```
<property>
  <name>mapreduce.cluster.local.dir</name>
  <value>/mnt/raid/local</value>
</property>
<property>
  <name>mapreduce.jobtracker.system.dir</name>
  <value>/mnt/raid/local</value>
</property>
<property>
  <name>mapreduce.jobtracker.staging.root.dir</name>
  <value>/mnt/raid/local</value>
</property>
<property>
 <name>mapreduce.cluster.temp.dir</name>
 <value>/mnt/raid/local</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"
change heap size

Conf/slave

On each slave we need to add aws instance name
For master node we need to add all aws instance name running as slaves.

I have written script file to setup all nodes environment i.e coping configuration file to other slaves.
As well as we need to go to every instance and set hotname using below command.
sudo hostmane ec2-xxxx.amazonaws.com
sudo vi /etc/hosts
then add below line in file
172.31.40.142 ec2-xxxx.amazonaws.com

Running Hadoop

Once we have done those changes we need to start dfs and yarn from master node as done for single node.

For hadoop experiment I have created 16 slaves and 1 master node with data set of 100GB. To run that experiment my program took 3 hr.

Graph and Explanation

Data Set: 1GB and 10GB
Number of Nodes: 1

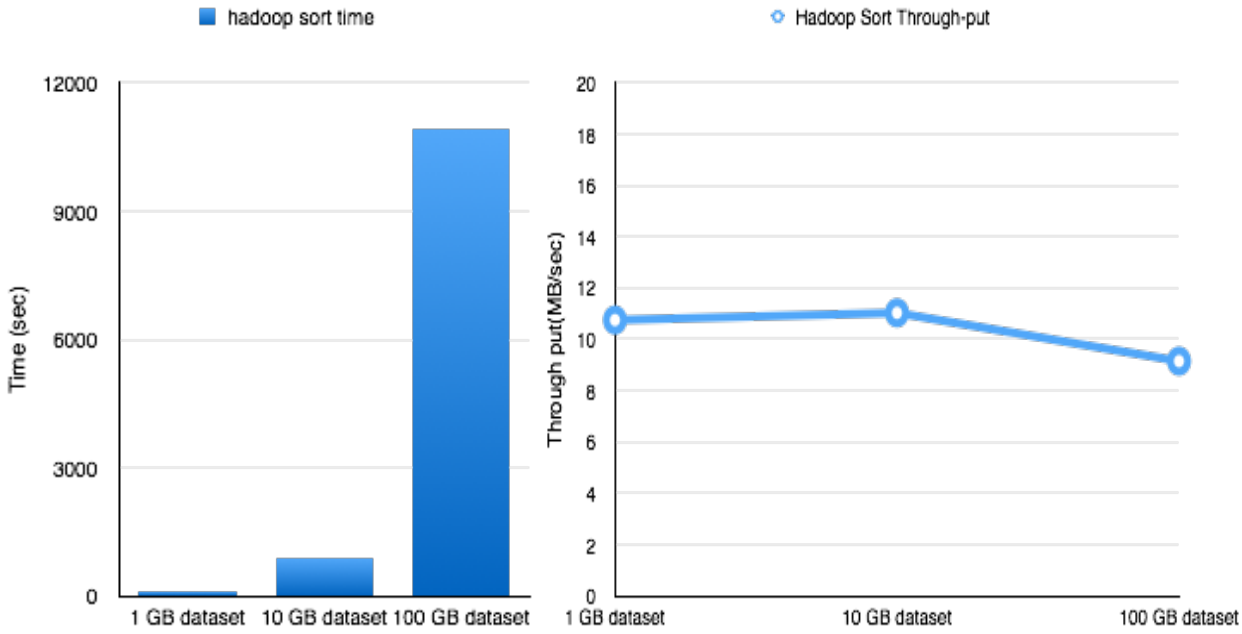
Hadoop	1 GB dataset	10GB dataset
Time (sec)	93	906

Data Set: 100GB
Number of Nodes: 16

Hadoop	100 GB dataset
Time (sec)	10920

Through-Put on different data set

Hadoop	1 GB dataset	10 GB dataset	100 GB dataset
Throughput (MB/sec)	10.75	11.03	9.15



Explanation

From above graph we see there is high thoughput for sorting 10GB data but for 100 GB data the throughput have decreased by 18 %. I think one of the major reason of this decrease is in throughput is network bandwidth and latency as data is shared across all worker node. Master nodes though have multiple mapper and reducer working on those data they still need to perform network operation and network operation are costly than I/O operation. Another reason of this throughput decrease is hadoop have centralized scheduler which is again a bottleneck.

Problem faced

JVM out of heap space: for this issue, I changed the JVM heap size in environment variable

GC overhead exceeded: For this issue I used faster garbage collection algorithm

Need to change the local directory and temp directory of hadoop because intermediary data are much larger, so we need to point that directory to mnt

Spark Sort

Problem Statement

In this part of assignment, i have written a "java program" to sort data set of 1GB, 10GB, and 100GB on single and 17 nodes hadoop cluster. This experiment aims to explore spark configuration, installation and learn spark API to write program.

Approach

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

For code we need to add jar file in our code to use spark configuration.

Code consists of a driver class that runs the map/reduce job. There is one mapper class used by driver class. I have used sortbykey() functionality for sorting data and saved the file using saveasTextFile function.

Configuration and Experiment

Configuration

Instance:

Model	vCPU	Mem (GB)	Storage (GB)
D2.xlarge	2	3.75	400(EBS)

Created raid0 at /mnt/raid location (Script is written in setup file)

Kernel Version: GNU/Linux

OS version:
NAME="Amazon Linux AMI"
VERSION="2016.03"
ID="amzn"
ID_LIKE="rhel fedora"
VERSION_ID="2016.03"
PRETTY_NAME="Amazon Linux AMI 2016.03"
ANSI_COLOR="0;33"
CPE_NAME="cpe:/o:amazon:linux:2016.03:ga"
HOME_URL=http://aws.amazon.com/amazon-linux-ami/

Java Version : 1.7.0.95 openJDK

Spark Version: spark-1.6.0-bin-hadoop2.6

Running Experiment on Single node

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

Command:

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

./Desktop/Spring/553/HW2/spark-1.6.0-bin-hadoop2.6/ec2/spark-ec2 --key-pair=Ec2s --identity-file=/Users/Aakash/Desktop/EC2/Ec2s.pem --region=us-west-2 --zone=us-west-2a --instance-type=c3.large --ebs-vol-size=100 --spot-price=0.035 launch spark

Just by following above steps We can easily install spark.

For single node, I have run experiment for 1GB and 10 GB dataset.

For 1GB dataset my sort program run for 1 min 10 sec.

For 10GB dataset my sort program run for 17 min.

Running Experiment on Multiple nodes

To set multiple nodes for spark we need to do below steps.

Need to export AWSAccessKeyId and AWSSecretKey then we need to run below
./Desktop/Spring/553/HW2/spark-1.6.0-bin-hadoop2.6/ec2/spark-ec2 --key-pair=Ec2s --identity-file=/Users/Aakash/Desktop/EC2/Ec2s.pem --region=us-west-2 --zone=us-west-2a --instance-type=c3.large –slaves=16 --ebs-vol-size=600 --spot-price=0.035 launch spark

However just by doing these configuration not worked out from me for 100GB data. I needed to do some extra configuration for running sort 100GB data because spark by default store data /mnt and /mnt2 directory and uses ephemeral-hdfs that also store data in /mnt and /mnt2 directory.

To solve this problem I used following steps.

I changed my hdfs to persistent-hdfs instead of ephemeral-hdfs because it uses /mnt directory and it not uses /vol0

Changes I did to use 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 fs -mkdir /inputdir

Now change spark configuration to use persistent-hdfs instead of ephemeral-hdfs 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

spark default directory from /mnt to /vol0

3: restart spark :: go to sbin and stop-all.sh then start-all.sh

For spark experiment I have created 16 slaves and 1 master node with data set of 100GB. To run that experiment my program took 3.87 hr

Graph and Explanation

Data Set: 1GB and 10GB
Number of Nodes: 1

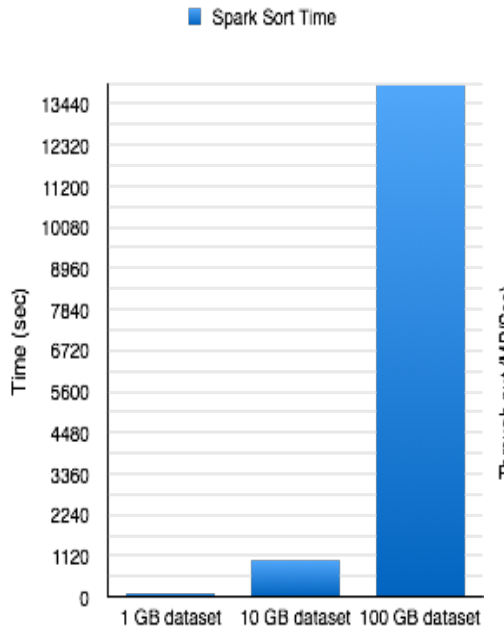
Spark	1 GB dataset	10GB dataset
Time (sec)	70	1020

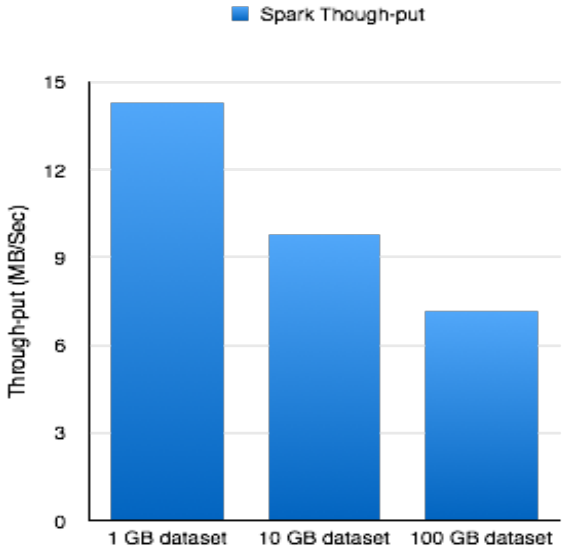
Data Set: 100GB
Number of Nodes: 16

Spark	100 GB dataset
Time (sec)	13932

Through-Put on different data set

Spark	1 GB dataset	10 GB dataset	100 GB dataset
Throughput (MB/sec)	14.28	9.8	7.17





Explanation

In my above graph I find there is high throughput for size 1GB dataset but for another dataset through put keep on decreasing. Its very tough to explain why through-put for spark decreased from 1GB to 10GB dataset may be due to AWS machine, I was working on that time was used by lots of people as the machine was not dedicated. For 100 GB data set still the performance dropped, I think the reason is same as it has happened with hadoop as well as maybe with some AWS instance related problems. Another reason I can think of is, I launched the instance with default script of spark ec2 script that may have some setting done on default that was reducing performance.

Problem faced

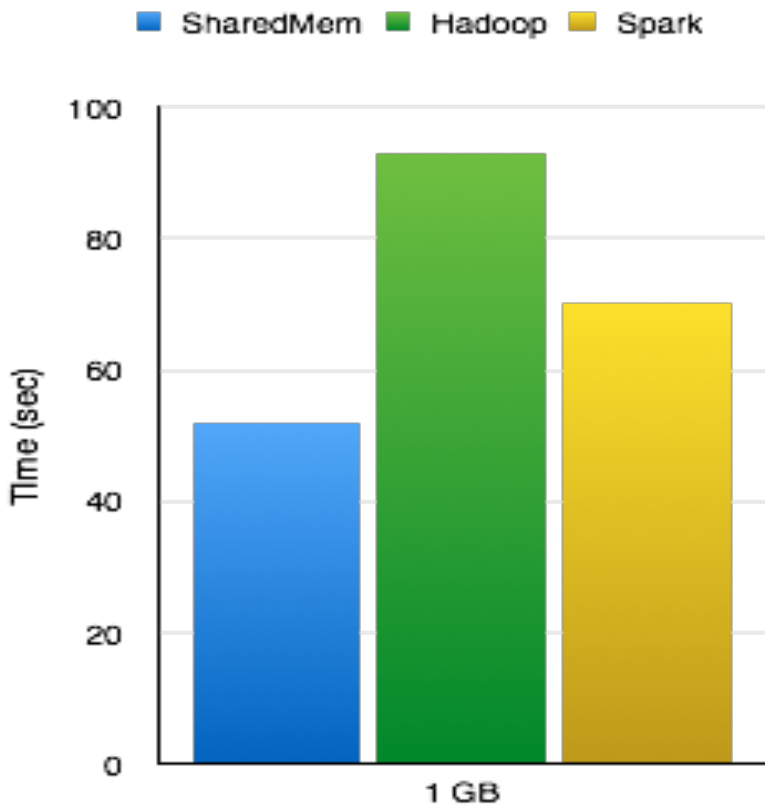
JVM out of heap space: for this issue, I changed the JVM heap size in environment variable

GC overhead exceeded: For this issue I used faster garbage collection algorithm

No space left on device: As spark script doesnot mount ebs volume to /mnt, I need to do all the configuration to point to ebs volume /vol0

Comparison between Shared memory/Hadoop/Spark Sort

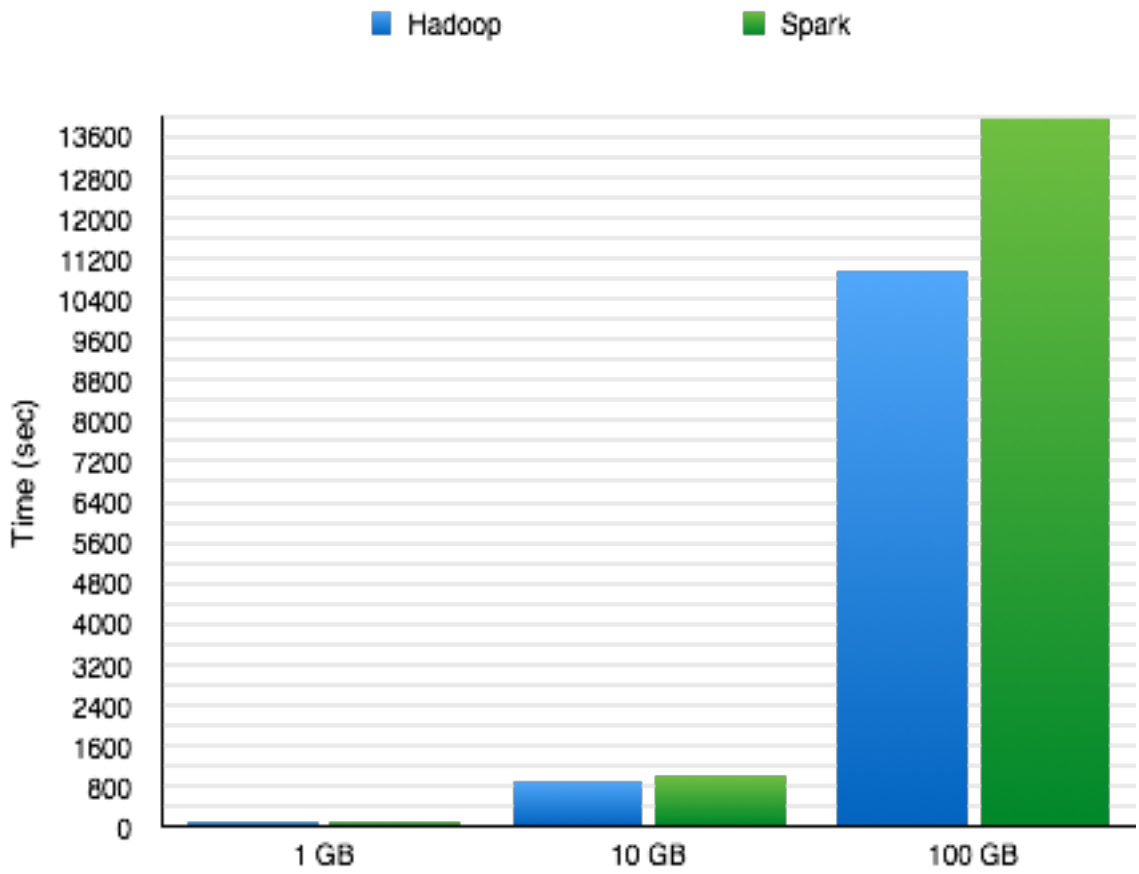
	Shared Memory	Hadoop	Spark
1 GB sort time (sec)	51.96	93	70



Graph Explanation

From above graph we can see that time took for shared-memory sort is atleast 20% less than spark and hadoop instance reason is because of better utilization of threads. Where as spark and hadoop are designed to solve problems on distributed system and they are not very much efficient on single node.

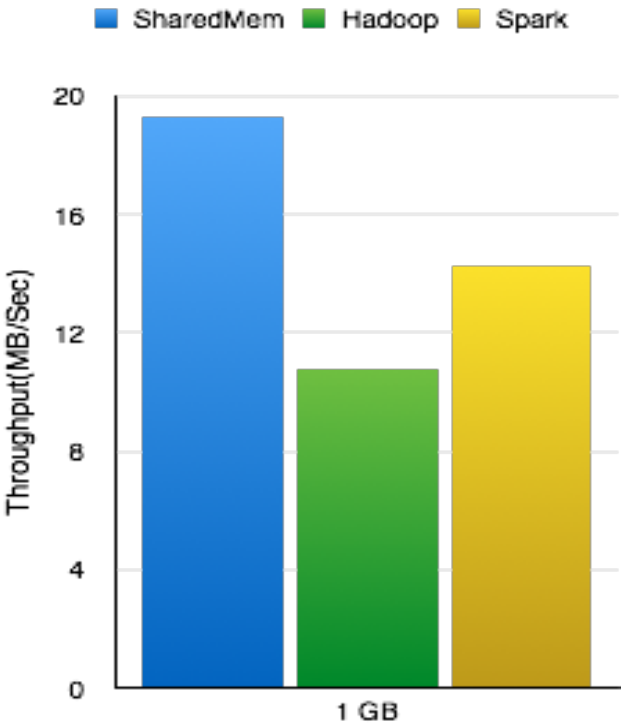
	Hadoop	Spark
1 GB sort time (sec)	93	70
10 GB sort time (sec)	906	1020
100 GB sort time (sec)	10920	13932



Graph Explanation

As from above graph we can see for single node the time taken by spark and hadoop are nearly equal but for dataset of 100 GB hadoop time is much lesser than spark. One reason I can think of for this is because spark keeps lots of intermediary data for if we need to do another processing on that intermediary data where as hadoop does not, so might be due to that we didn't see less time for spark. However if we need to run again some operation on intermediary data we have start new map-reduce task in hadoop but it not true for spark,We can use intermediary data in spark that will show performance inprovment.

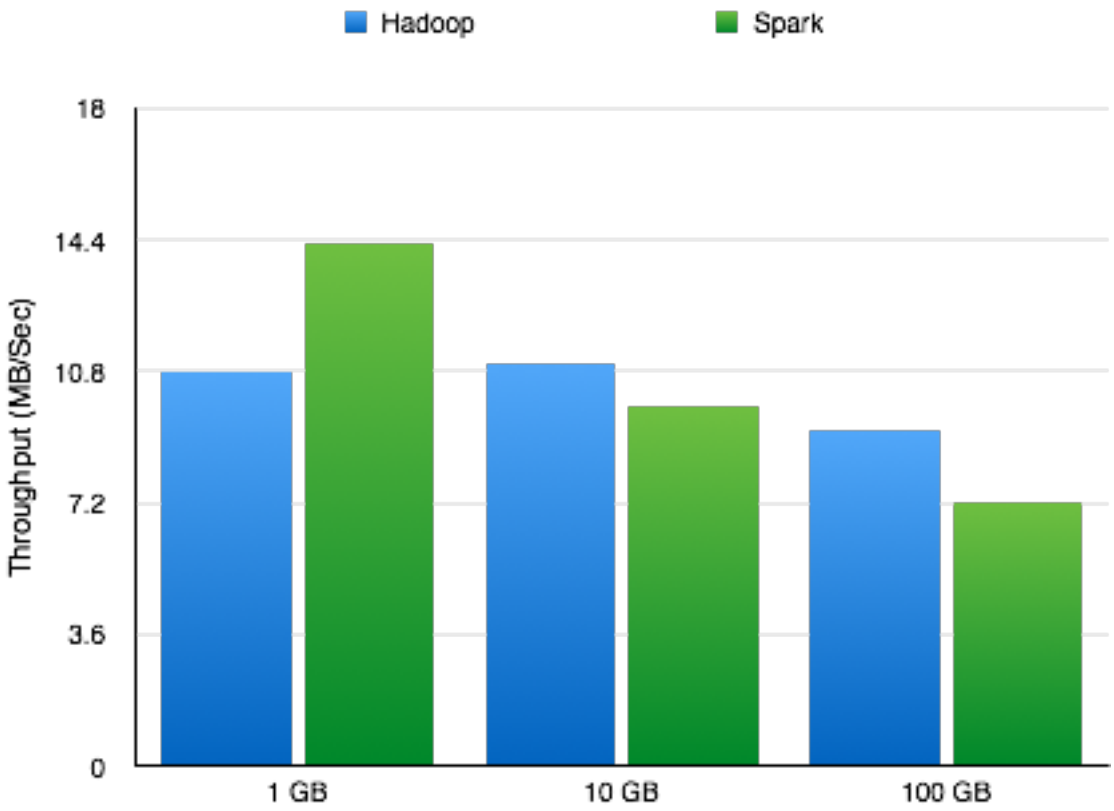
	Shared Memory	Hadoop	Spark
1GB dataset Throughput (MB/sec)	19.24	10.75	14.28



Graph Explanation

On above graph for single node shared-memory outperforms spark and hadoop for throughput because shared-memory program is well optimized to run on single node with careful selection of number of thread and max queue size. On other hand Hadoop and spark are used for distributed system. Running shared memory program on distributed system will lack performance due to network bandwidth and latency also lot of modification will be required to run my code in distributed system.

	Hadoop	Spark
1 GB dataset Throughput (MB/sec)	10.75	14.28
10 GB dataset Throughput (MB/sec)	11.03	9.8
100 GB dataset Throughput (MB/sec)	9.15	7.17



Graph Explanation

Above graph show the throughput comparison between hadoop and spark on different data set. Spark through put for 1 GB data set is more than hadoop whereas not true for other data set of 10GB and 100GB. Spark is designed to use intermediary data for applications like machine learning and analytics and it gives better performance on that kind of applications. Spark is built on by customizing hadoop there are cases where hadoop can outperform spark like above.

PA-2 Question and Answers

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

Answer: General architecture of hadoop consist of master node and server node running on heterogeneous computers. The master node consists of a Job Tracker, Task Tracker, NameNode, and DataNode. A slave node acts as both a DataNode and TaskTracker. It is possible to have data-only worker nodes and compute-only worker nodes. In normal working task workloads on the slave nodes should be isolated from the masters. Slaves nodes are frequently decommissioned for maintenance.

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?

Answer: In hadoop configuration every port no has its own responsibility and used for different purpose

For example: ports 8025,8035 configured in yarn-site.xml are used for resource management and task management. Where are 9000 is used for hdfs.

On single computer two process are not allowed to share the same port , if we configure same port then there will be collisions and application will might not run.

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

Answer: We can change the number of mapper and reducer in mapred-site.xml configuration

```
<property>
    <name>mapred.tasktracker.map.tasks.maximum</name>
   <value>2</value>
</property>
```

On changing property value we can change maximum map tasks per node normally it should be same as no of cores.

```
<property>
  <name>mapred.tasktracker.reduce.tasks.maximum</name>
   <value>2</value>
<property>
```

On changing property value we can change maximum task tasks per node normally it should be same as no of cores.

What conclusions can you draw?

Based on above experiment what software stack to use and when to use it mostly depends on what kind of application we are building on that infrastructure. It also depends on what kind of data we are going to process and do we need to perform task on those data again as well as what is the amount of data application will process.

For smaller task, less data (in few GB) :: Write my own code (as we can change logic on our need , no need to setup cluster, simple batch job or code of 100 lines works perfect)

For large data set(in TB), processing of intermediary data is less needed:: Hadoop

For large data set (TB), processing intensive, requires intermediary data:: Spark

To choose between hadoop and spark is mostly based on application need.

Which seems to be best at 1 node scale?

Shared-memory approach seems to me as best for one node scale. As we need to write less code and no need to setup cluster or store file in HDFS.

How about 16 nodes?

For 16 nodes, I would favor hadoop in some cases and some cases spark . hadoop configuration we need to do by manualy written script that takes time where as spark does all those work by its python script.

Another thing we need to keep in mind is about what kind of data ans processing our application needs.

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

I would prefer spark in case of 100 nodes because they have easy to use API in java ,scala and python. If any application needs 100 nodes there is a probability that they keeps on scaling and their infrastructure should be utilized by heterogeneous application. Therefore Spark is better option according to me for 100 nodes cluster.

How about 1000 node scales?

For 1000 nodes scales I would prefer to have both spark and hadoop cluster in infrastructure as both uses HDFS file system. So we can expand any cluster on need basis or on usage. Both spark and hadoop have their own advantage.In hadoop lots of research have been done which makes it easy to customize for needs, Where as Spark is also very good option now days because it can process graphs and use the existing machine-learning libraries.

Compare your results with those from the Sort Benchmark [9], specifically the winners in 2013 and 2014 who used Hadoop and Spark. Also, what can you learn from the CloudSort benchmark, a report can be found at [10].

Sorting data of TBs requires intensive computing, I/O and network task. It also need various hit-trial with new technique to get that benchmark. They are sorting data of 100TB on 330 EC2 instance at $4.51 per TB. So on average per computer they are sorting 100TB/330 = 0.30 TB. Where as in our case we are sorting 1 TB on single instance i.e. 3 times more than they are sorting on their single compute node. They have another factor to consider also of network which is not true for our shared-memory (as we are sorting on single node).

In terms of price and money their result are far more good than my code because my code took around several hours to sort data given the fact that data was 3 times more on single node. They have used various new techniques to run faster where as in my code I have used very basic technique like producer- consumer and priority sorting.