Cloud System Configuration:

Instance created with Node type: Skylake and Source: Ubuntu 18.04

Details for node 81b02796-a84a-413b-a207-67e8fd04cc77

Site: uc	**Cluster:** chameleon	**Platform Type:** x86_64	**# CPUs:** 2
# of Threads: 48	**RAM Size:** 192 GiB	**Node Type:** compute_skylake	**Wattmeter:** No
Version: 1e8bd9c674908ad73deccf432744960b62184fbd			
Bios			
Release Date: 05/21/2018	**Vendor:** 1.4	**Version:** Dell Inc.	
Chassis			
Manufacturer: Dell Inc.	**Name:** PowerEdge R740	**Serial:** 9YSY7M2	
GPU			
GPU: No			
Network Adapters			More
Operating System			
Kernel:	**Name:**	**Version:**	
Processor			
Cache L1d: 32768	**Cache L1i:** 32768	**Cache L2:** 1048576	**Cache L3:** 20185088
Clock Speed: 2600000000	**Instruction Set:** x86-64	**Model:** Intel Xeon	**Other Description:** Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
Vendor: Intel	**Version:** None		
Storage Devices			Less
Storage Device #0			
Device: sda	**Driver:** megaraid_sas	**Interface:** SATA	**Model:** MZ7KM240HMHQ0D3
Rev: GD53	**Size:** 240 GB	**Vendor:** Samsung	
Supported Job Types			
Best Effort: No	**Deploy:** Yes	**Virtual:** ivt	

Installed qemu/KVM virtualization or containerization tools and used it to deploy following VMs/containers with the following sizes:

1. 1 small instance (4-cores, 8gb ram, 50gb disk)
   Created another VM (for namenode & scheduler) of disk=15GB, core=6, ram=16GB.
2. 4 small instance (4-cores, 8gb ram, 40gb disk)
   Created another VM (for namenode & scheduler) of disk=20GB, core=6, ram=16GB.
3. 1 large instance (16-cores, 32gb ram, 130gb disk)
   Created another VM (for namenode & scheduler) of disk=30GB, core=6, ram=16GB.

Hadoop Configuration Setup on virtual cluster:

Download and extract the Hadoop framework on namenode.

$ wget http://www.gtlib.gatech.edu/pub/apache/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz

$ cp hadoop-3.2.1.tar.gz /exports/projects/.

$ cd /exports/projects

$ tar xzvf hadoop

$ cd hadoop-3.2.1

We need to edit /etc/profile.d/hadoop.sh on namenode as well as datanodes.

We need to edit etc/hadoop/hadoop-env.sh, etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml, etc/hadoop/mapred-site.xml on namenode.

/etc/profile.d/hadoop.sh :
export HADOOP_HOME=/exports/projects/hadoop-3.2.1/
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/

etc/hadoop/hadoop-env.sh
edit line 54
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/

core-site.xml

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ cat etc/hadoop/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://192.168.122.116:9000</value>
        </property>
        <property>
                <name>io.file.buffer.size</name>
                <value>131072</value>
        </property>
</configuration>
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$
```

hdfs-site.xml

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ cat etc/hadoop/hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/home/ubuntu/hdfs-metadata</value>
        </property>
        <property>
                <name>dfs.blocksize</name>
                <value>67108864</value>
        </property>
        <property>
                <name>dfs.namenode.handler.count</name>
                <value>100</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/home/ubuntu/hdfs-data</value>
        </property>
</configuration>
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$
```

yarn-site.xml

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ cat etc/hadoop/yarn-site.xml
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>192.168.122.116</value>
        </property>
        <property>
                <name>yarn.nodemanager.log-dirs</name>
                <value>/home/ubuntu/hdfs-logs</value>
        </property>
</configuration>
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$
```

mapred-site.xml

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ cat etc/hadoop/mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <property>
                <name>mapreduce.jobhistory.address</name>
                <value>192.168.122.116:10020</value>
        </property>
        <property>
                <name>mapreduce.jobhistory.webapp.address</name>
                <value>192.168.122.116:19888</value>
        </property>
        <property>
                <name>yarn.app.mapreduce.am.env</name>
                <value>HADOOP_MAPRED_HOME=/exports/projects/hadoop-3.2.1/</value>
        </property>
        <property>
                <name>mapreduce.map.env</name>
                <value>HADOOP_MAPRED_HOME=/exports/projects/hadoop-3.2.1/</value>
        </property>
        <property>
                <name>mapreduce.reduce.env</name>
                <value>HADOOP_MAPRED_HOME=/exports/projects/hadoop-3.2.1/</value>
        </property>
</configuration>
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$
```

Running Hadoop on virtual cluster:

bin/hadoop jar HadoopSort.jar HadoopSort /home/input/input1GB.txt /home/output 2

Success output:

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ bin/hadoop jar HadoopSort.jar HadoopSort /home/input/input1GB.txt /home/output 2
Hadoop Sort Program Started
2020-04-27 04:18:30,200 INFO client.RMProxy: Connecting to ResourceManager at /192.168.122.116:8032
2020-04-27 04:18:30,661 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ubuntu/.staging/job_1587947702316_0002
2020-04-27 04:18:30,762 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-04-27 04:18:30,888 INFO input.FileInputFormat: Total input files to process : 1
2020-04-27 04:18:30,929 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-04-27 04:18:31,048 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-04-27 04:18:31,068 INFO mapreduce.JobSubmitter: number of splits:15
2020-04-27 04:18:31,109 INFO Configuration.deprecation: mapred.textoutputformat.separator is deprecated. Instead, use mapreduce.output.textoutputformat.separator
2020-04-27 04:18:31,173 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2020-04-27 04:18:31,198 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1587947702316_0002
2020-04-27 04:18:31,198 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-04-27 04:18:31,338 INFO conf.Configuration: resource-types.xml not found
2020-04-27 04:18:31,338 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2020-04-27 04:18:31,403 INFO impl.YarnClientImpl: Submitted application application_1587947702316_0002
2020-04-27 04:18:31,433 INFO mapreduce.Job: The url to track the job: http://demo-namenode.local:8088/proxy/application_1587947702316_0002/
2020-04-27 04:18:31,433 INFO mapreduce.Job: Running job: job_1587947702316_0002
2020-04-27 04:18:36,565 INFO mapreduce.Job: Job job_1587947702316_0002 running in uber mode : false
2020-04-27 04:18:36,567 INFO mapreduce.Job:  map 0% reduce 0%
2020-04-27 04:18:43,690 INFO mapreduce.Job:  map 7% reduce 0%
2020-04-27 04:18:44,699 INFO mapreduce.Job:  map 13% reduce 0%
2020-04-27 04:18:45,710 INFO mapreduce.Job:  map 27% reduce 0%
2020-04-27 04:18:47,734 INFO mapreduce.Job:  map 60% reduce 0%
2020-04-27 04:18:48,743 INFO mapreduce.Job:  map 67% reduce 0%
2020-04-27 04:18:49,758 INFO mapreduce.Job:  map 73% reduce 0%
2020-04-27 04:18:50,768 INFO mapreduce.Job:  map 100% reduce 0%
2020-04-27 04:18:57,829 INFO mapreduce.Job:  map 100% reduce 50%
2020-04-27 04:18:58,839 INFO mapreduce.Job:  map 100% reduce 100%
2020-04-27 04:19:01,886 INFO mapreduce.Job: Job job_1587947702316_0002 completed successfully
2020-04-27 04:19:02,019 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=1020000024
                FILE: Number of bytes written=2043859544
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1001836748
                HDFS: Number of bytes written=1000000000
                HDFS: Number of read operations=55
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Launched map tasks=15
                Launched reduce tasks=2
                Data-local map tasks=15
                Total time spent by all maps in occupied slots (ms)=131059
```

```
                Total time spent by all reduce tasks (ms)=24119
                Total vcore-milliseconds taken by all map tasks=131059
                Total vcore-milliseconds taken by all reduce tasks=24119
                Total megabyte-milliseconds taken by all map tasks=134204416
                Total megabyte-milliseconds taken by all reduce tasks=24697856
        Map-Reduce Framework
                Map input records=10000000
                Map output records=10000000
                Map output bytes=1000000000
                Map output materialized bytes=1020000180
                Input split bytes=1740
                Combine input records=0
                Combine output records=0
                Reduce input groups=10000000
                Reduce shuffle bytes=1020000180
                Reduce input records=10000000
                Reduce output records=10000000
                Spilled Records=20000000
                Shuffled Maps =30
                Failed Shuffles=0
                Merged Map outputs=30
                GC time elapsed (ms)=3933
                CPU time spent (ms)=73940
                Physical memory (bytes) snapshot=8739938304
                Virtual memory (bytes) snapshot=44142948352
                Total committed heap usage (bytes)=7898923008
                Peak Map Physical memory (bytes)=591138816
                Peak Map Virtual memory (bytes)=2600534016
                Peak Reduce Physical memory (bytes)=789180416
                Peak Reduce Virtual memory (bytes)=2617581568
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1001835008
        File Output Format Counters
                Bytes Written=1000000000
Time taken to sort: 32466.0 milliseconds.
```

Output files:

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ bin/hadoop fs -ls /home/output
Found 3 items
-rw-r--r--   1 ubuntu supergroup          0 2020-04-27 04:18 /home/output/_SUCCESS
-rw-r--r--   1 ubuntu supergroup  505240900 2020-04-27 04:18 /home/output/part-r-00000
-rw-r--r--   1 ubuntu supergroup  494759100 2020-04-27 04:18 /home/output/part-r-00001
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$
```

Spark Configuration Setup on virtual cluster:

Download and extract the Spark framework on namenode.

$ wget https://apache.claz.org/spark/spark-3.0.0-preview2/spark-3.0.0-preview2-bin-hadoop2.7.tgz

$ cp spark-3.0.0-preview2-bin-hadoop3.2.tgz /exports/projects/.

$ cd /exports/projects

$ tar xvf spark-3.0.0-preview2-bin-hadoop3.2.tgz
$ cd spark-3.0.0-preview2-bin-hadoop3.2

We need to edit conf/spark-env.sh, conf/spark-defaults.conf on namenode.
We need to edit /etc/profile.d/hadoop.sh on all namenode and datanodes.

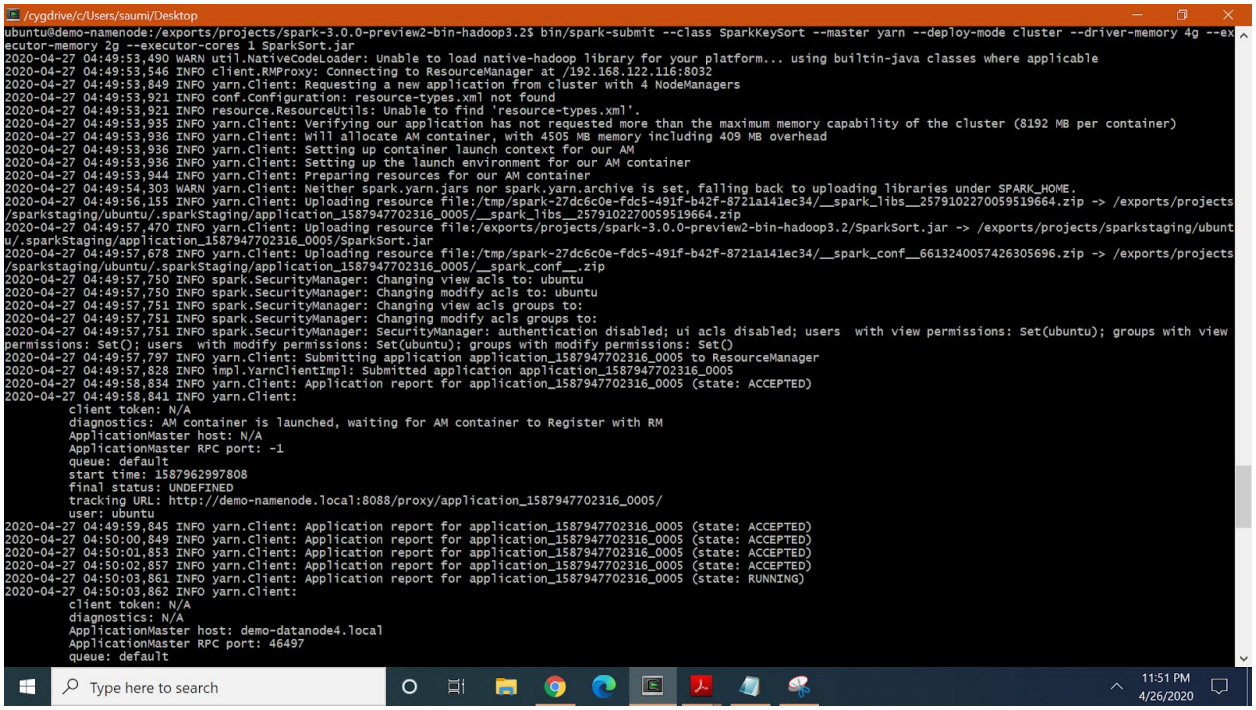
conf/spark-env.sh
HADOOP_CONF_DIR=/exports/projects/hadoop-3.2.1/etc/hadoop
YARN_CONF_DIR=/exports/projects/hadoop-3.2.1/etc/hadoop

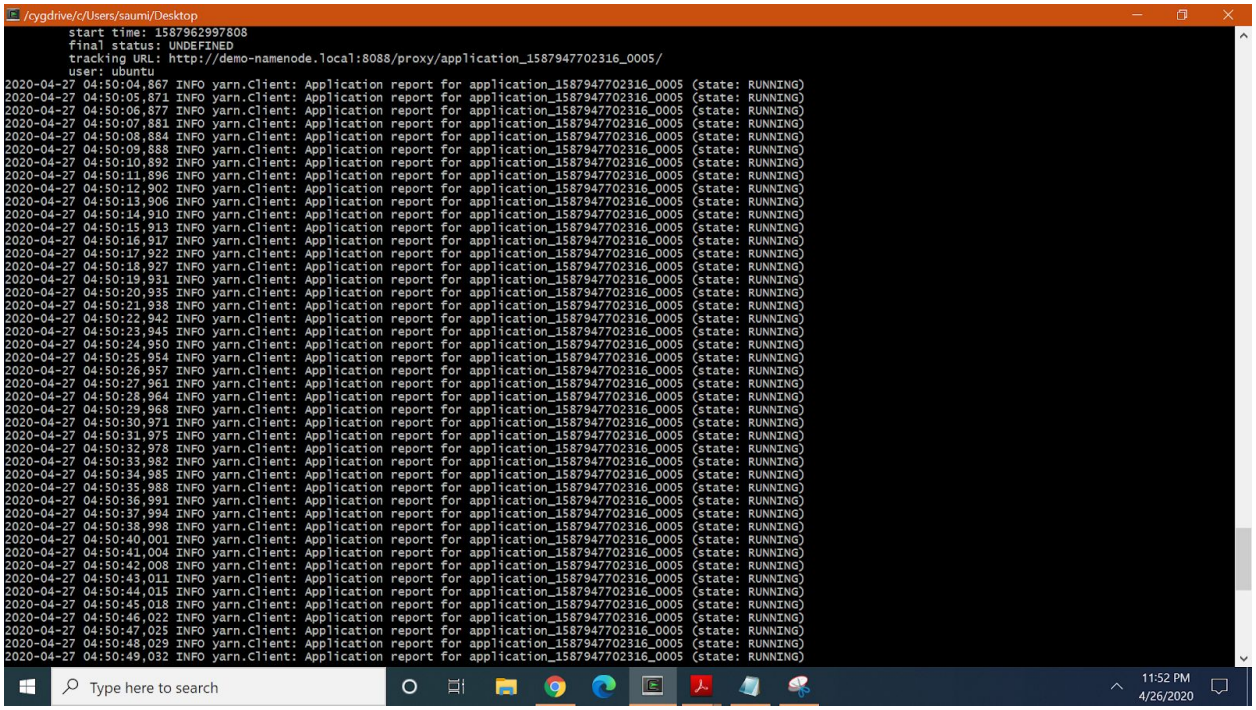
conf/spark-defaults.conf
$ mkdir /exports/projects/sparkstaging
$ cp conf/spark-defaults.conf.template conf/spark-defaults.conf
$ vim conf/spark-defaults.conf
spark.yarn.stagingDir /exports/projects/sparkstaging

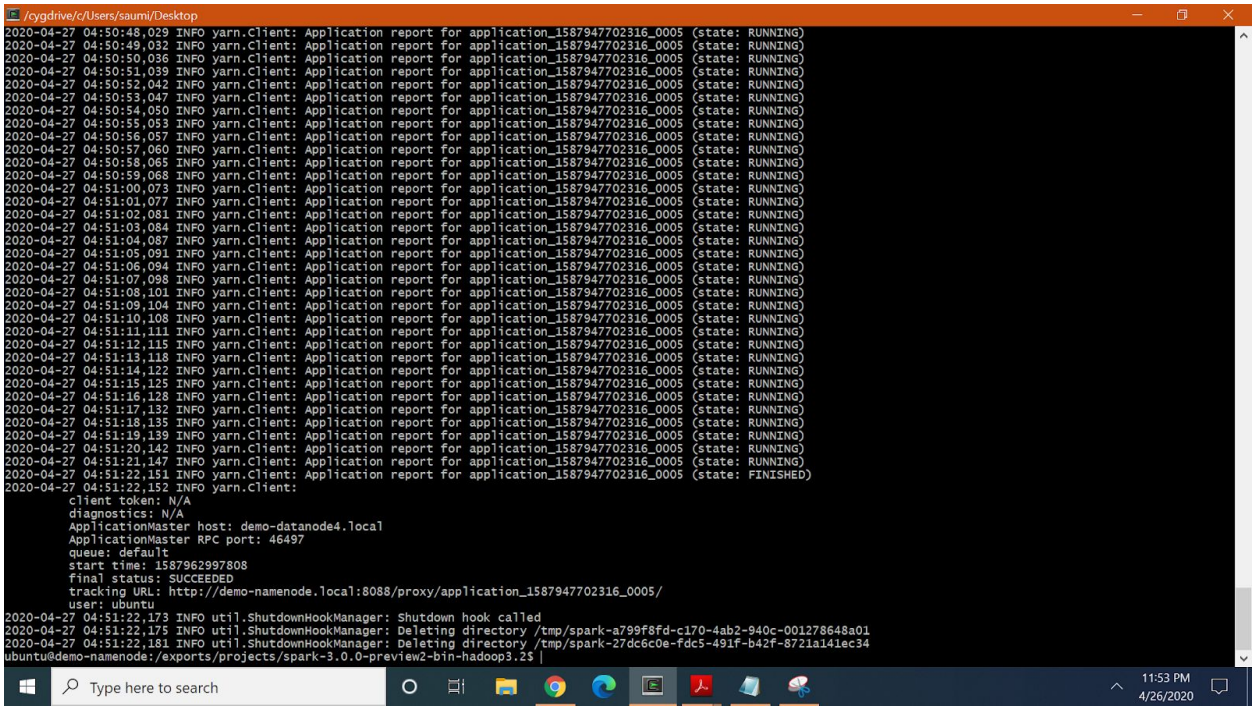
/etc/profile.d/hadoop.sh
export HADOOP_PREFIX=/exports/projects/hadoop-2.9.2/
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/
export SPARK_HOME=/exports/projects/spark-3.0.0-preview2-bin-hadoop3.2/

Running Spark on virtual cluster:
bin/spark-submit --class SparkKeySort --master yarn --deploy-mode cluster --driver-memory 4g --ex
ecutor-memory 2g --executor-cores 1 SparkSort.jar

Success output:


/cygdrive/c/Users/saumi/Desktop
ubuntu@demo-namenode:/exports/projects/spark-3.0.0-preview2-bin-hadoop3.2$ bin/spark-submit --class SparkKeySort --master yarn --deploy-mode cluster --driver-memory 4g --executor-memory 2g --executor-cores 1 SparkSort.jar
2020-04-27 04:49:53,490 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-04-27 04:49:53,546 INFO client.RMProxy: Connecting to ResourceManager at /192.168.122.116:8032
2020-04-27 04:49:53,849 INFO yarn.Client: Requesting a new application from cluster with 4 NodeManagers
2020-04-27 04:49:53,921 INFO conf.Configuration: resource-types.xml not found
2020-04-27 04:49:53,921 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2020-04-27 04:49:53,935 INFO yarn.Client: Verifying our application has not requested more than the maximum memory capability of the cluster (8192 MB per container)
2020-04-27 04:49:53,936 INFO yarn.Client: Will allocate AM container, with 4505 MB memory including 409 MB overhead
2020-04-27 04:49:53,936 INFO yarn.Client: Setting up container launch context for our AM
2020-04-27 04:49:53,936 INFO yarn.Client: Setting up the launch environment for our AM container
2020-04-27 04:49:53,944 INFO yarn.Client: Preparing resources for our AM container
2020-04-27 04:49:54,303 WARN yarn.Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
2020-04-27 04:49:56,155 INFO yarn.Client: Uploading resource file:/tmp/spark-27dc6c0e-fdc5-491f-b42f-8721a141ec34/__spark_libs__2579102270059519664.zip -> /exports/projects/sparkstaging/ubuntu/.sparkStaging/application_1587947702316_0005/__spark_libs__2579102270059519664.zip
2020-04-27 04:49:57,470 INFO yarn.Client: Uploading resource file:/exports/projects/spark-3.0.0-preview2-bin-hadoop3.2/SparkSort.jar -> /exports/projects/sparkstaging/ubuntu/.sparkStaging/application_1587947702316_0005/SparkSort.jar
2020-04-27 04:49:57,678 INFO yarn.Client: Uploading resource file:/tmp/spark-27dc6c0e-fdc5-491f-b42f-8721a141ec34/__spark_conf__6613240057426305696.zip -> /exports/projects/sparkstaging/ubuntu/.sparkStaging/application_1587947702316_0005/__spark_conf__.zip
2020-04-27 04:49:57,750 INFO spark.SecurityManager: Changing view acls to: ubuntu
2020-04-27 04:49:57,750 INFO spark.SecurityManager: Changing modify acls to: ubuntu
2020-04-27 04:49:57,751 INFO spark.SecurityManager: Changing view acls groups to:
2020-04-27 04:49:57,751 INFO spark.SecurityManager: Changing modify acls groups to:
2020-04-27 04:49:57,751 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(ubuntu); groups with view permissions: Set(); users with modify permissions: Set(ubuntu); groups with modify permissions: Set()
2020-04-27 04:49:57,797 INFO yarn.Client: Submitting application application_1587947702316_0005 to ResourceManager
2020-04-27 04:49:57,828 INFO impl.YarnClientImpl: Submitted application application_1587947702316_0005
2020-04-27 04:49:58,834 INFO yarn.Client: Application report for application_1587947702316_0005 (state: ACCEPTED)
2020-04-27 04:49:58,841 INFO yarn.Client:
client token: N/A
diagnostics: AM container is launched, waiting for AM container to Register with RM
ApplicationMaster host: N/A
ApplicationMaster RPC port: -1
queue: default
start time: 1587962997808
final status: UNDEFINED
tracking URL: http://demo-namenode.local:8088/proxy/application_1587947702316_0005/
user: ubuntu
2020-04-27 04:49:59,845 INFO yarn.Client: Application report for application_1587947702316_0005 (state: ACCEPTED)
2020-04-27 04:50:00,849 INFO yarn.Client: Application report for application_1587947702316_0005 (state: ACCEPTED)
2020-04-27 04:50:01,853 INFO yarn.Client: Application report for application_1587947702316_0005 (state: ACCEPTED)
2020-04-27 04:50:02,857 INFO yarn.Client: Application report for application_1587947702316_0005 (state: ACCEPTED)
2020-04-27 04:50:03,861 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:03,862 INFO yarn.Client:
client token: N/A
diagnostics: N/A
ApplicationMaster host: demo-datanode4.local
ApplicationMaster RPC port: 46497
queue: default
Type here to search
11:51 PM
4/26/2020


/cygdrive/c/Users/saumi/Desktop
start time: 1587962997808
final status: UNDEFINED
tracking URL: http://demo-namenode.local:8088/proxy/application_1587947702316_0005/
user: ubuntu
2020-04-27 04:50:04,867 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:05,871 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:06,877 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:07,881 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:08,884 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:09,888 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:10,892 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:11,896 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:12,902 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:13,906 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:14,910 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:15,913 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:16,917 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:17,922 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:18,927 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:19,931 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:20,935 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:21,938 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:22,942 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:23,945 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:24,950 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:25,954 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:26,957 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:27,961 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:28,964 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:29,968 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:30,971 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:31,975 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:32,978 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:33,982 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:34,985 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:35,988 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:36,991 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:37,994 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:38,998 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:40,001 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:41,004 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:42,008 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:43,011 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:44,015 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:45,018 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:46,022 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:47,025 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:48,029 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
2020-04-27 04:50:49,032 INFO yarn.Client: Application report for application_1587947702316_0005 (state: RUNNING)
Type here to search
11:52 PM
4/26/2020



Output files:

```
ubuntu@demo-namenode:/exports/projects/hadoop-3.2.1$ bin/hadoop fs -ls /home/output
Found 2 items
-rw-r--r--   1 ubuntu supergroup          0 2020-04-27 04:51 /home/output/_SUCCESS
-rw-r--r--   1 ubuntu supergroup 1000000000 2020-04-27 04:51 /home/output/part-00000
```

Performance evaluation of sort:

Experiment	Shared Memory	Linux	Hadoop Sort	Spark Sort
1 small.instance, 1GB dataset	18720 ms	11230 ms	43882 ms	115886 ms
1 small.instance, 4GB dataset	115204 ms	60970 ms	133347 ms	311105 ms
1 small.instance, 16GB dataset	672304 ms	268250 ms	503190 ms	1172810 ms
1 large.instance, 1GB dataset	12756 ms	8840 ms	41406 ms	109747 ms
1 large.instance, 4GB dataset	45222 ms	33220 ms	130943 ms	311952 ms
1 large.instance, 16GB dataset	223550 ms	178530 ms	469123 ms	1180000 ms
1 large.instance, 24GB dataset	799015 ms	287540 ms	697240 ms	1714749 ms
4 small.instances, 1GB dataset	NA	NA	26315 ms	83574 ms
4 small.instances, 4GB dataset	NA	NA	57416 ms	264049 ms
4 small.instances, 16GB dataset	NA	NA	195466 ms	968035 ms
4 small.instances, 24GB dataset	NA	NA	403100 ms	1271000 ms

Note: We were facing issues while running Hadoop and Spark sort programs for 32GB and 48GB. We were getting an error as " No space left on disk".
Therefore, we have tested all sort programs for a maximum of 24GB dataset as seen in the table above.

Question : Number of threads, mappers, reducers used in each experiment is given in the table below. Also, the table contains data read and data write values for each dataset for each experiment.

MySort

MySort	Sort Threads	IO Threads	Data Read (Bytes)	Data Write (Bytes)
1 small.instance, 1GB dataset	16	4	1000000000	1000000000
1 small.instance, 4GB dataset	16	4	4000000000	4000000000
1 small.instance, 16GB dataset (1 GB Chunk size)	8	4	43769682000	32194357000
1 large.instance, 1GB dataset	20	16	1002764000	974104000
1 large.instance, 4GB dataset	16	16	4000000000	4000000000
1 large.instance, 16GB dataset	32	16	16000000000	16000000000
1 large.instance, 24GB dataset (2 GB Chunk size)	16	10	46445683930	46847716110

LinuxSort:

Linux Sort	Threads	Data Read (Bytes)	Data write (Bytes)
1 small.instance, 1GB dataset	4	968884000	976564000
1 small.instance, 4GB dataset	12	6255124000	7810692360
1 small.instance, 16GB dataset	4	31239203120	3124648582
1 large.instance, 1GB dataset	24	965044000	946078650
1 large.instance, 4GB dataset	20	3892172000	3902884200
1 large.instance, 16GB dataset	22	24069752000	31246685980
1 large.instance, 24GB dataset	22	41983445820	46865010930

Hadoop Sort

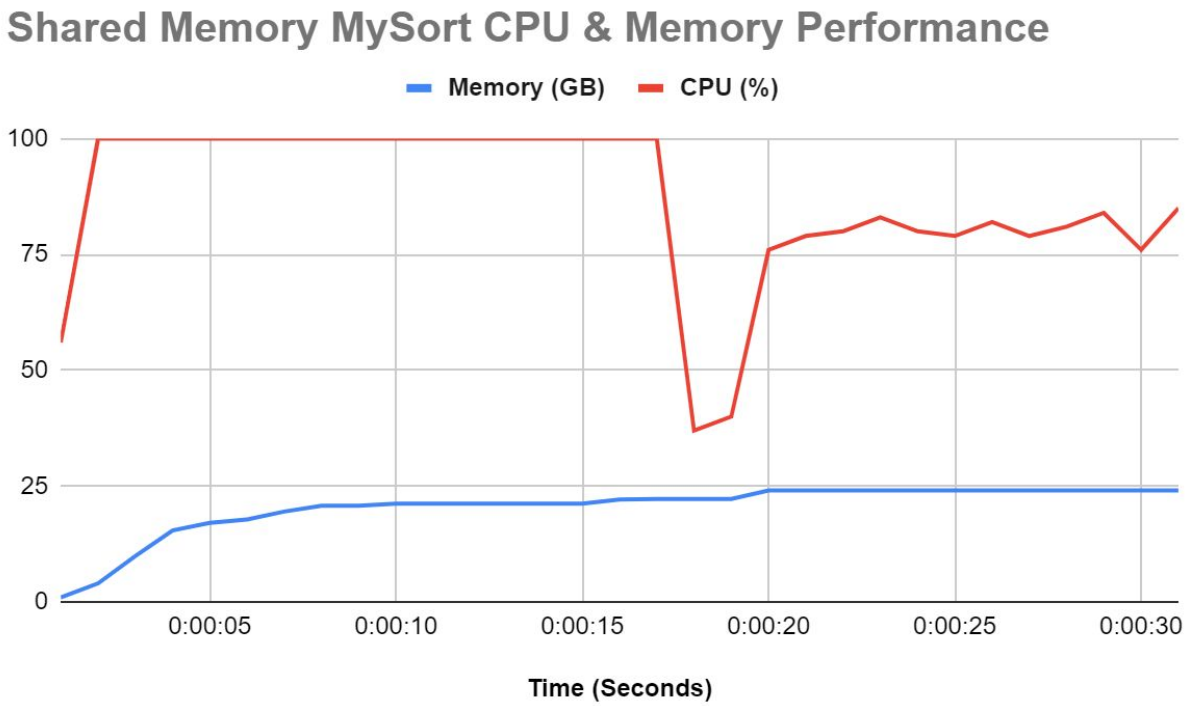
Hadoop	Data Read (Bytes)	Data Write (Bytes)	Number of reducers
1 small.instance, 1GB dataset	1001836688	1000000000	2
1 small.instance, 4GB dataset	4007739968	4000000000	2
1 small.instance, 16GB dataset	16031220170	16000000000	2
1 large.instance, 1GB dataset	1001836718	1000000000	2
1 large.instance, 4GB dataset	4007740088	4000000000	2
1 large.instance, 16GB dataset	16031220470	16000000000	2
1 large.instance, 24GB dataset	24046830652	24000000000	2
4 small.instances, 1GB dataset	1001836748	1000000000	4
4 small.instances, 4GB dataset	4007740208	4000000000	8
4 small.instances, 16GB dataset	16031223099	16000000000	8
4 small.instances, 24GB dataset	24046832800	24000000000	4

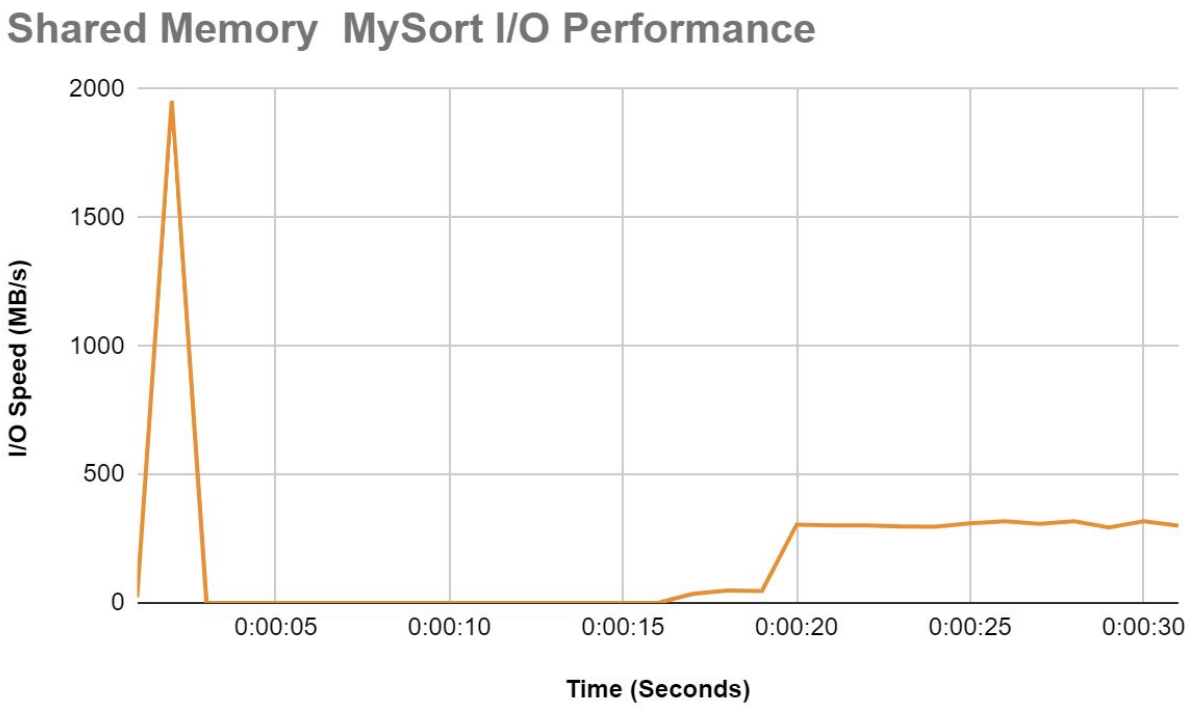
Spark Sort:

Spark	Data Read (Bytes)	Data Write (Bytes)
1 small.instance, 1GB dataset	1001836688	1000000000
1 small.instance, 4GB dataset	4007739968	4000000000
1 small.instance, 16GB dataset	16000000000	16000000000
1 large.instance, 1GB dataset	1000000000	1000000000
1 large.instance, 4GB dataset	4000000000	4000000000
1 large.instance, 16GB dataset	16000000000	16000000000
1 large.instance, 24GB dataset	24000000000	24000000000
4 small.instances, 1GB dataset	1001836748	1000000000
4 small.instances, 4GB dataset	4007740208	4000000000
4 small.instances, 16GB dataset	16031223099	16000000000
4 small.instances, 24GB dataset	24046832800	24000000000

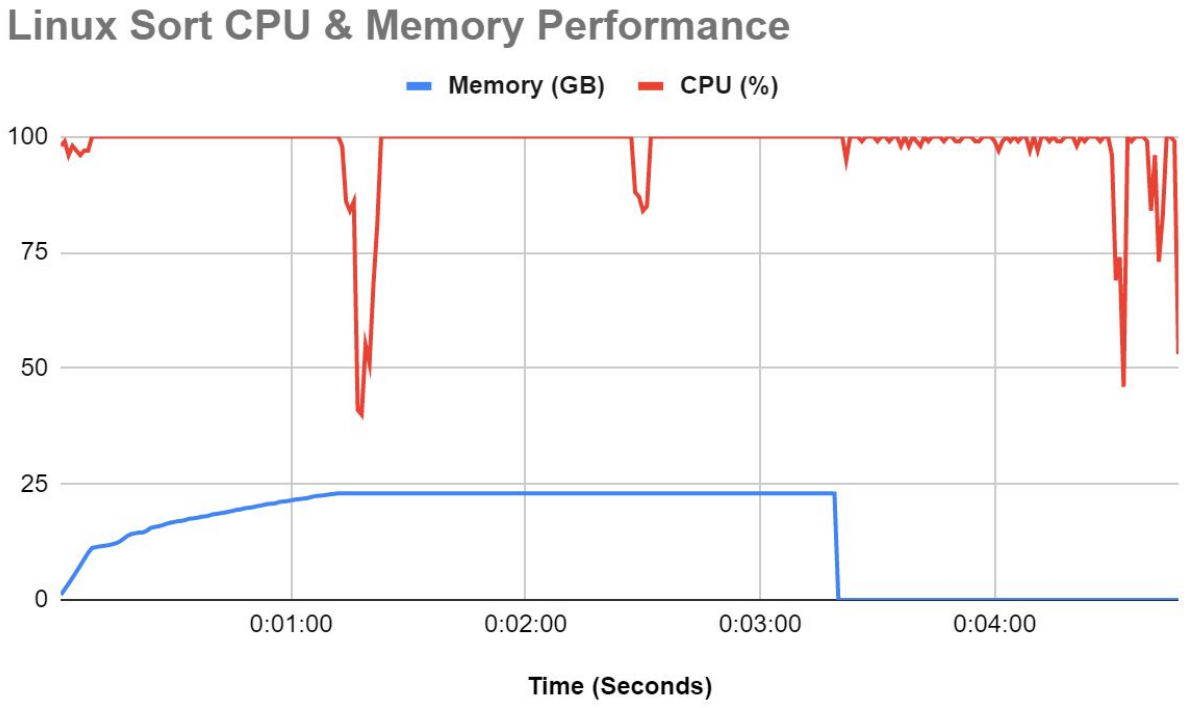
Plot for cpu utilization ,memory utilization and the disk I/O data :

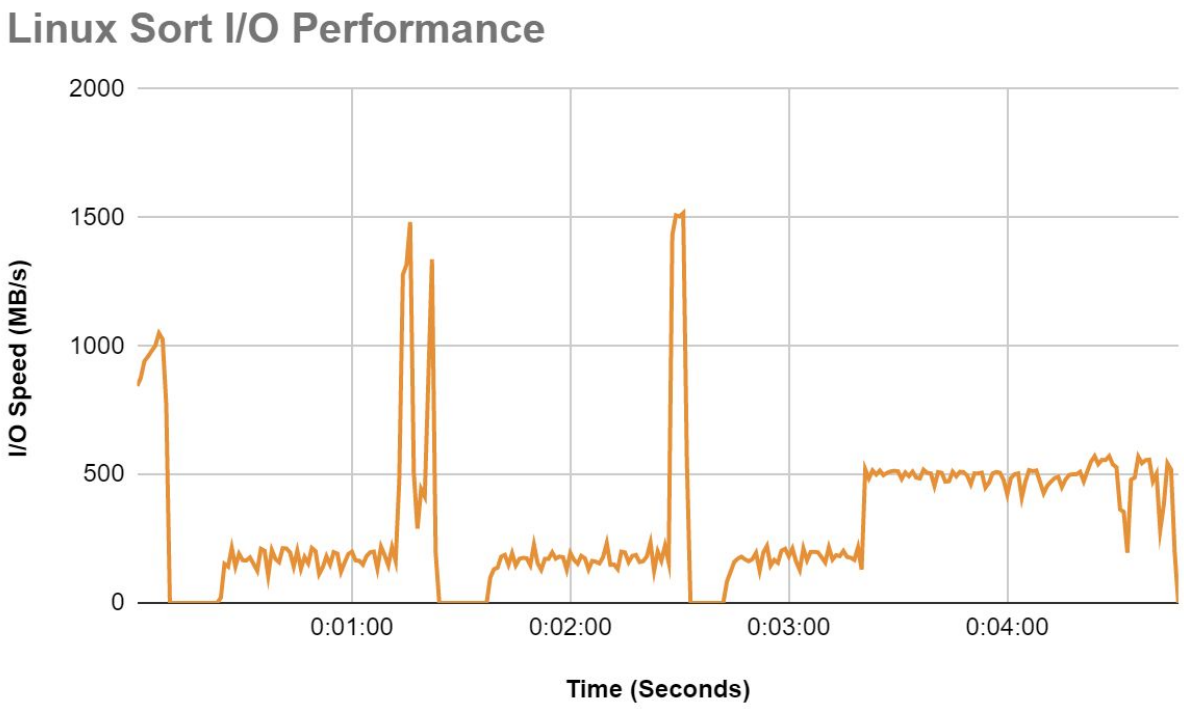
1 large.instance, 24GB dataset : MySort



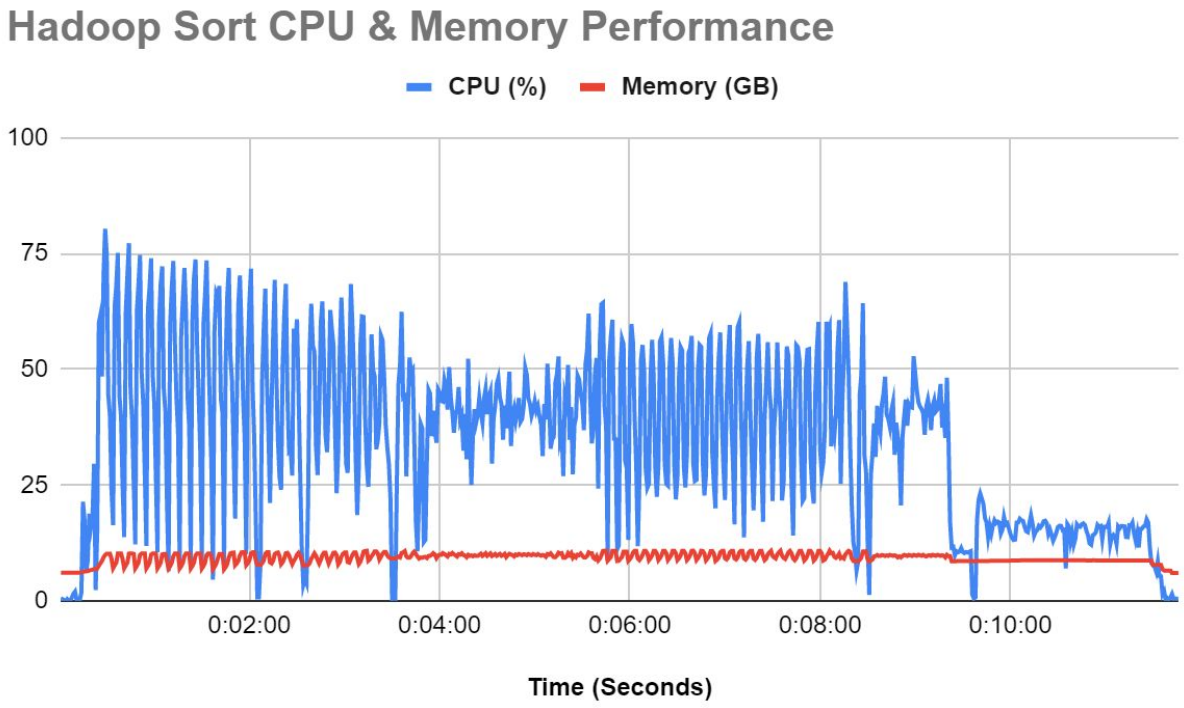


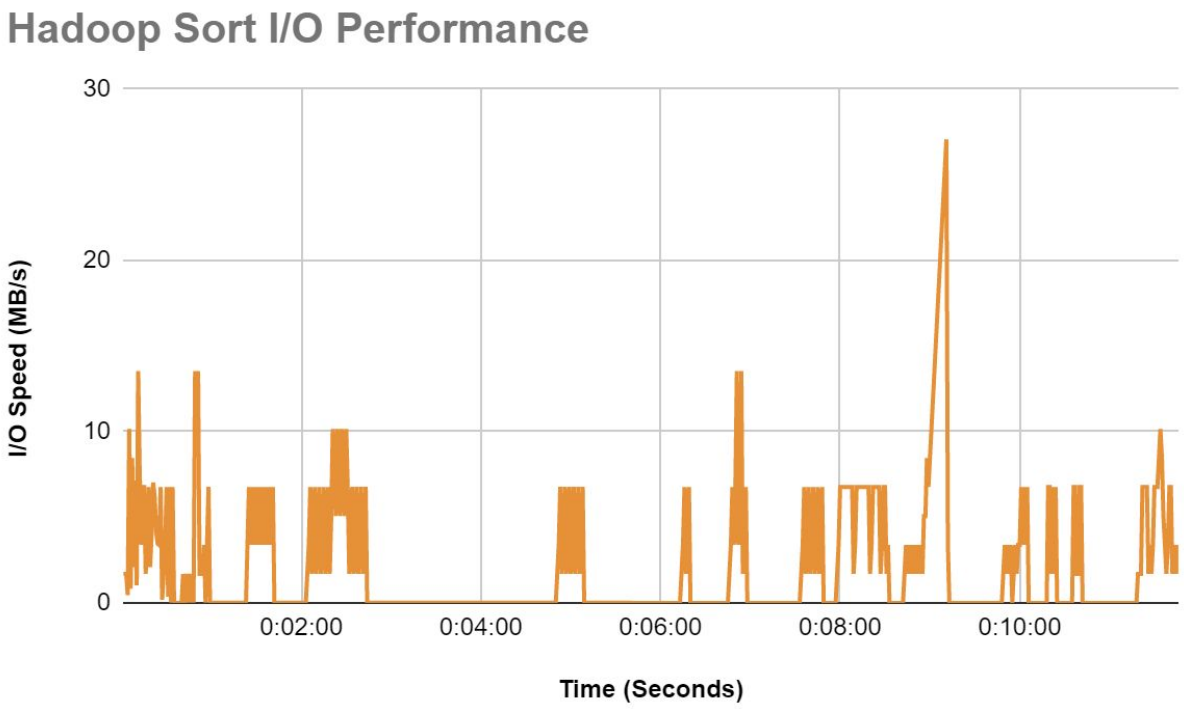
1 large.instance, 24GB dataset : Linux Sort



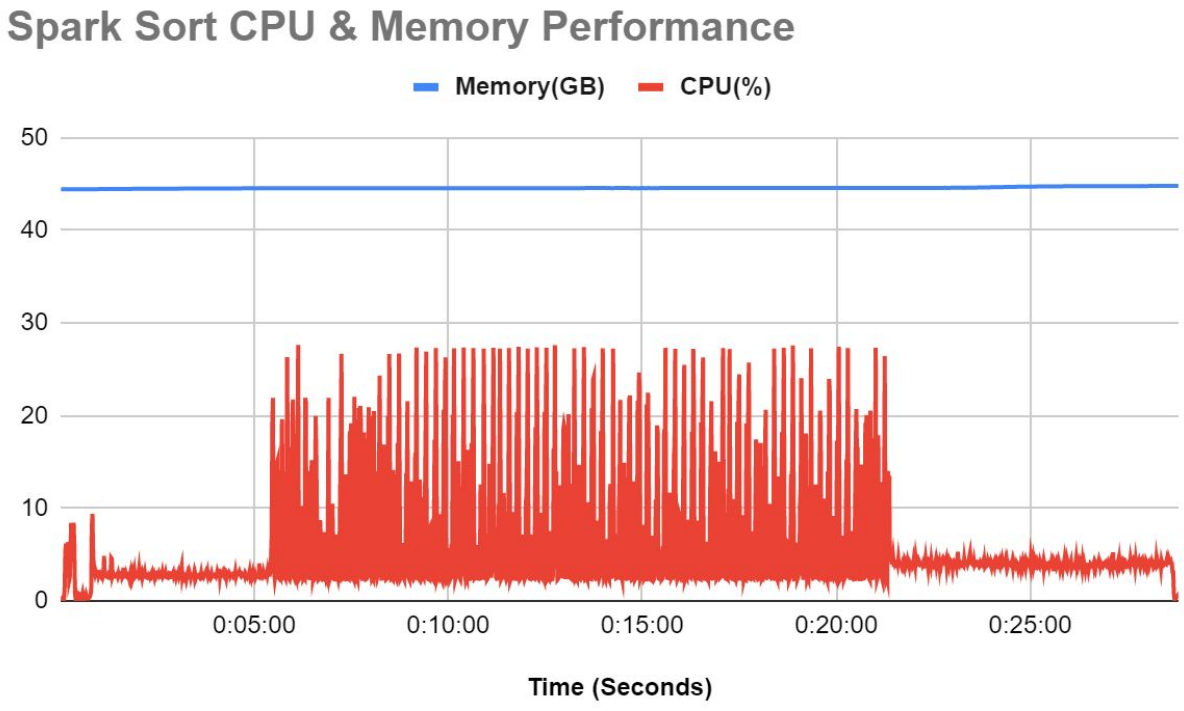


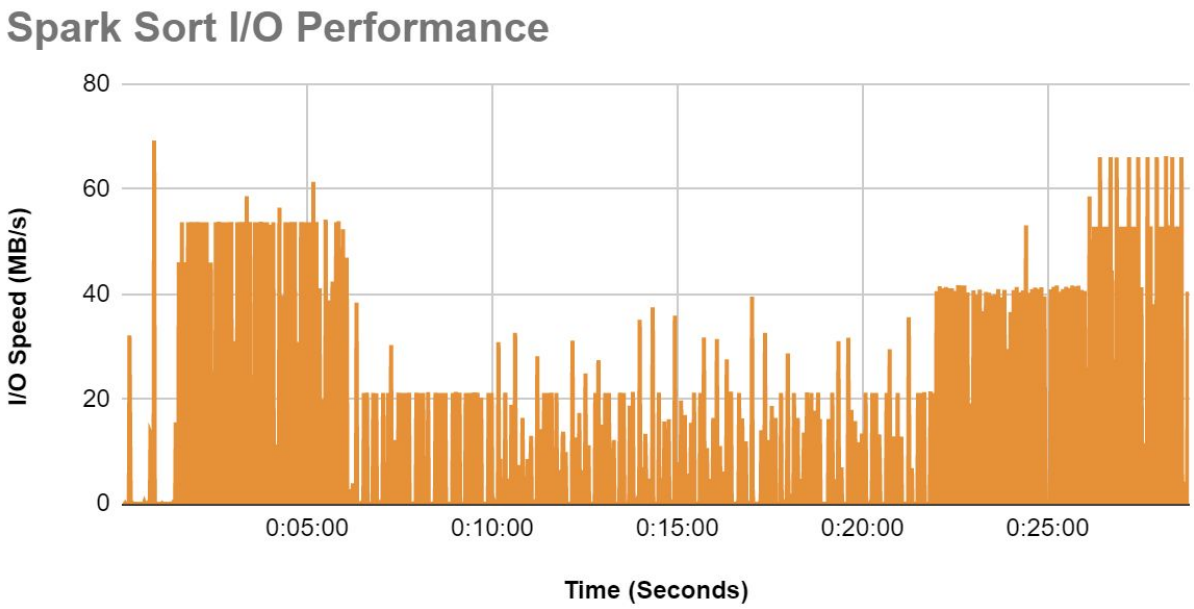
1 large.instance, 24GB dataset : Hadoop



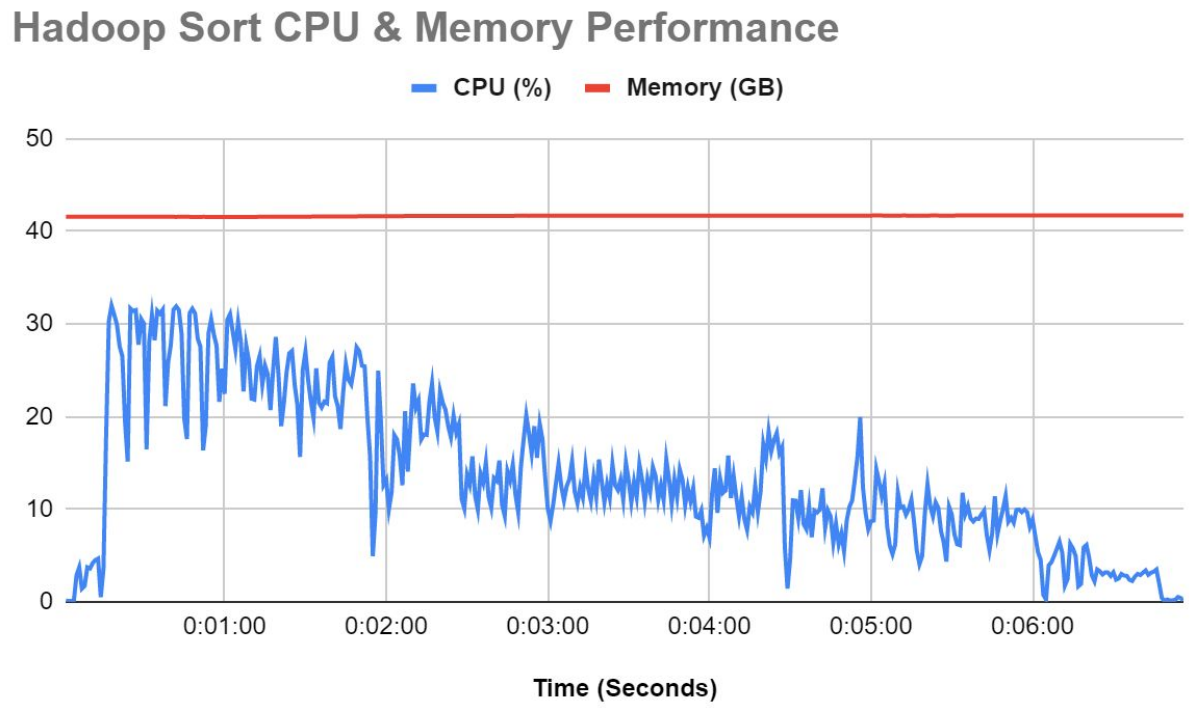


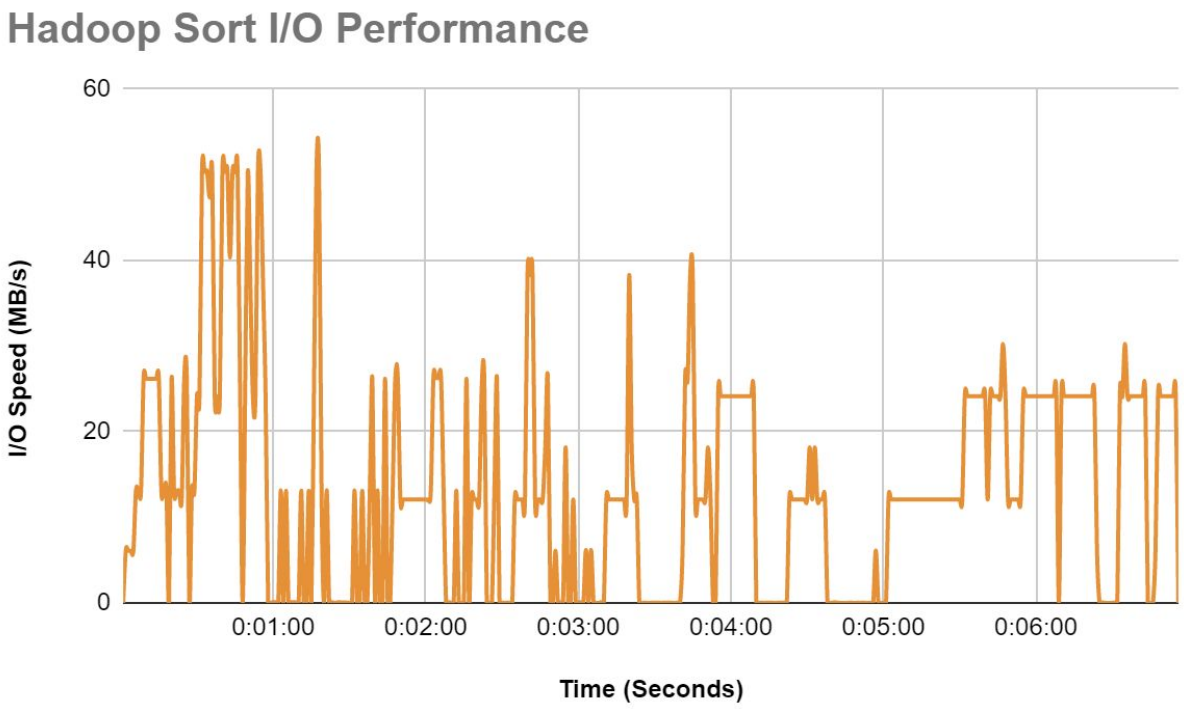
1 large.instance, 24GB dataset : Spark



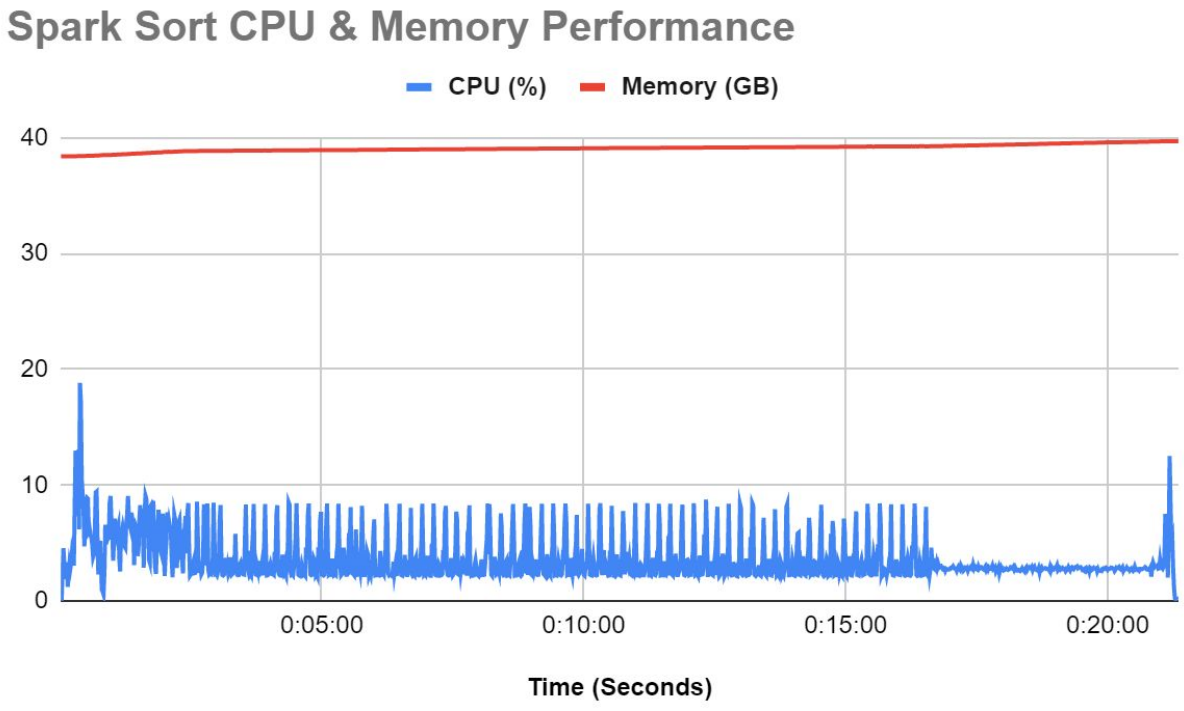


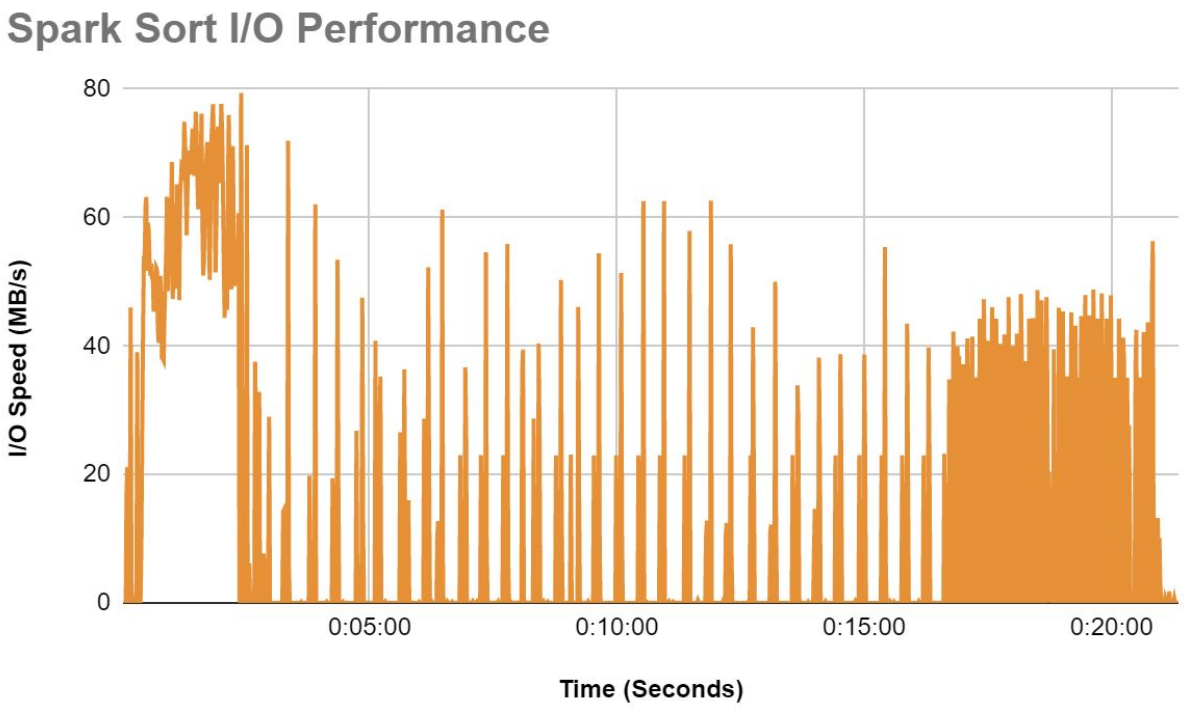
4 small.instances, 24GB dataset : Hadoop





4 small.instances, 24GB dataset: Spark





What conclusions can you draw?

- Among all 3 sort programs MySort, HadoopSort and SparkSort ; MySort is taking less time to sort.
- Hadoop Sort is clearly the best sorting technique when compared to Spark sort as per our experiment observations.

Ideally, Spark should give better results. But it depends on which system we are using and how many resources we have.

We used default configuration for Spark Configuration file.
vim conf/spark-defaults.conf
spark.yarn.stagingDir /exports/projects/sparkstaging

We used default parameters for Spark Submit Command while running spark program.
> --driver-memory 4g
> --executor-memory 2g
>--num-executors :2
>--executor-cores: 1

Note: Increasing the RAM and cores for experiment could have got better results for Spark.

Which seems to be best at 1 node scale (1 large.instance)?

- Among all the 4 different sorts, the best time has been noted by Linux sort
- To consider the second best, among Shared memory Sort, Hadoop Sort & Spark Sort, the best performance is given by the Shared Memory sort for datasets (1, 4 & 16GB) at 1 node scale. For 24GB, the time taken by Hadoop Sort is less than Shared Memory sort.

Is there a difference between 1 small.instance and 1 large.instance?

- Shared Memory shows a significant difference between 1 small instance and 1 large instance as the dataset size increases. Similar is the case with linux sort as well.
- But in the case of Hadoop Sort and Spark Sort, the time taken to sort for 1 large instance is less compared with 1 small instance. But the difference in time is not as huge as seen in Shared memory and Linux sort.

How about 4 nodes (4 small.instance)?

- At 4 node scale, the results seem to give less time for Hadoop Sort than Spark Sort for all datasets.

What speedup do you achieve with strong scaling between 1 to 4 nodes?

- Strong scaling for 1GB between 1small instance and 4 small instances -

1GB	1 small instance	4 small instance	Speedup Achieved
Hadoop	43882 ms	26315 ms	40%
Spark	115886 ms	83574 ms	27%

- 4GB

4GB	1 small instance	4 small instance	Speedup Achieved
Hadoop	133347 ms	57416 ms	56%
Spark	311105 ms	264049 ms	15%

- 16GB

16GB	1 small instance	4 small instance	Speedup Achieved
Hadoop	503190 ms	195466 ms	61%
Spark	1172810 ms	968035 ms	17.4%

What speedup do you achieve with weak scaling between 1 to 4 nodes?

- 1 small instance 1gb, 4 small instance 4gb

	4 small instance 4GB	1 small instance 1GB	Speedup Achieved
Hadoop	57416 ms	43882 ms	23.5%
Spark	264049 ms	115886 ms	56.1%

- 1 small instance 4gb, 4 small instance 16gb

	4 small instance 16GB	1 small instance 4GB	Speedup Achieved
Hadoop	195466 ms	133347 ms	31.7%
Spark	968035 ms	311105 ms	67.8%

- 1 small.instance, 1GB dataset, 1 large.instance, 4GB dataset - all

	1 large instance 4GB	1 small instance 1GB	Speedup Achieved
Shared Memory	45222 ms	18720 ms	58.6%
Linux	33220 ms	11230 ms	66.1%
Hadoop	130943 ms	43882 ms	66.4%
Spark	311952 ms	115886 ms	62.8%

- 1 small.instance, 4GB dataset, 1 large.instance, 16GB dataset -all

	1 large instance 16GB	1 small instance 4GB	Speedup Achieved
Shared Memory	223550 ms	115204 ms	48.4%
Linux	178530 ms	60970 ms	65.8%
Hadoop	469123 ms	133347 ms	71.5%
Spark	1180000 ms	311105 ms	73.6%

How many small.instances do you need with Hadoop to achieve the same level of performance as your shared memory sort?

- Based on the results obtained, for 1 small instance to 4 small instances, there is almost a 40% reduction in time taken for sorting 1GB data. Therefore, to achieve the same level of performance as shared memory sort, almost 8 small instances would be needed for 1GB.

- For 4GB data, the time taken by 4 small instances from 1 small instance shows a reduction of 56% which gives a much better result than the shared memory sort.

How about how many small.instances do you need with Spark to achieve the same level of performance as you did with your shared memory sort?

- Based on the results obtained, for 1 small instance to 4 small instances, there is almost 28% reduction in time taken for sorting 1GB data. Therefore, to achieve the same level of performance as shared memory sort, almost 20 small instances would be needed for 1GB.
- Similarly, for 4GB data, 15% reduction is seen for 4 small instances compared to 1 small instance. In this case too, to achieve the same level of performance as shared memory sort, almost 20 small instances would be needed.

Can you draw any conclusions on the performance of the bare-metal instance performance from HW5 compared to the performance of your sort on a large instance through virtualization?

- MySort program observations on With bare-metal instance

Experiment	Shared Memory
Baremetal instance, 1GB dataset	13117 ms
Baremetal instance, 4GB dataset	53823 ms
Baremetal instance, 16GB dataset	543786 ms

- MySort program observations on a large instance through virtualization

Experiment	Shared Memory
1 large.instance, 1GB dataset	12756 ms
1 large.instance, 4GB dataset	45222 ms
1 large.instance, 16GB dataset	223550 ms

- From above evaluations, we can see that MySort benchmark performed better on a large instance compared to bare-metal instance.

Can you predict which would be best if you had 100 small.instances?

- Spark would be best for 100 small instances.
- The reason for this is when compared with Hadoop MapReduce it operates in steps, the workflow for MapReduce is: read data from cluster, perform an operation, write results to the cluster, read updated data from the cluster, perform next operation, write next result to cluster.

- Spark on the other hand completes full data analytics operations in-memory and in near real-time. The workflow for Spark is: read data from the cluster, perform all of the requisite analytic operations, write results to the cluster.
- Spark stores the intermediate data so that we can directly use that data when we want at any time but the only constraint is that spark consumes a huge amount of memory.

How about 1000?

- It depends on which system we are using and how many resources we have. Because Spark takes large system resources.
- Though Spark is 100 times faster than Hadoop, it's not very friendly to intensive jobs. So still for intensive and large scale jobs, Hadoop would be the best choice for 1000 nodes.

Compare your results with those from the Sort Benchmark (http://sortbenchmark.org), specifically the winners in 2013 and 2014 who used Hadoop and Spark.

- In 2013, Hadoop was the winner with the sorting speed of 1.42 TB/min as it calculated 102.5 TB in 4,328 seconds with 2100 nodes of distributed systems.
- In 2014, Apache Spark was the winner with the sorting speed of 4.27 TB/min and it calculated 100 TB in 1,406 seconds with only 207 Amazon EC2 nodes.
- They are much faster than our shared memory performance. Because they have different processors and disks. Also, number nodes are higher for them.
- In our case, Hadoop Sort for 24GB with 4 small instance: sort time is :403100 ms
  So sorting speed is = 24GB/403100ms = 59.53 MB/sec
- In our case, Spark Sort for 24GB with 4 small instance: sort time is :1528000 ms
  So sorting speed is = 24GB/1528000ms = 15.7 MB/sec

Question : Also, what can you learn from the CloudSort benchmark, a report can be found at (http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf).

- CloudSort benchmark is something that makes use of resources available at public cloud to perform sorting operations on the data. CloudSort proposes using the cloud to define a benchmark that measures the efficiency of external sort from a total-cost of ownership perspective. It provides a platform where we can derive more innovations on the clouds support to IO intensive tasks.
- We can see Cloudsort has great advantages in external sort. Because it is more accessible to normal people, people don't have to have access to national labs or some institutions to do that. And the cloud platform like Amazon provides a reasonable and affordable price for us to do external sort on cloud. Moreover, we can easily compare our results with some sorting benchmark and try to improve our own performance.
- CloudSort benchmark is one of the best public cloud benchmark systems we have and it takes in consideration all the factors like cost, speed and efficiency for the calculation of benchmark.