# Ex.2

## 1.

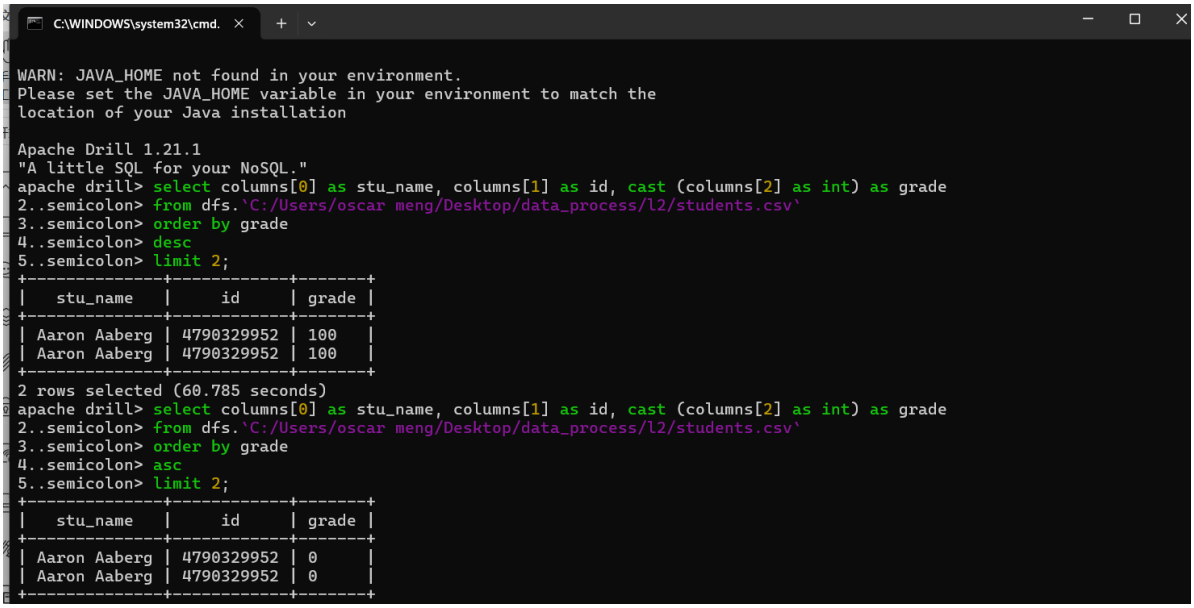
The 5Gb file is too big to upload it on canvas, you can refer to the screen shot to check the size of this file, and the python file that generates the 5Gb csvfile is also uploaded on canvas.

## 2.

### a)

| stu_name | id | score |
|---|---|---|
| Aaron Aaberg | 4790329952 | 0 |

The screen shot of the result:

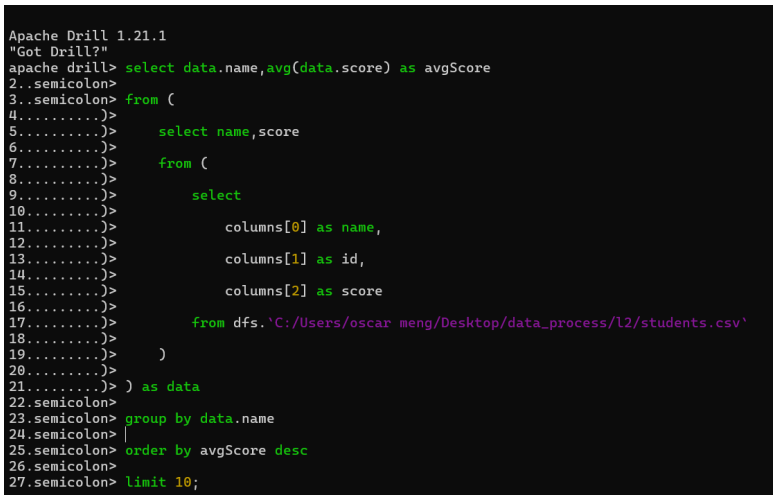


```
C:\WINDOWS\system32\cmd.

WARN: JAVA_HOME not found in your environment.
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation

Apache Drill 1.21.1
"A little SQL for your NoSQL."
apache drill> select columns[0] as stu_name, columns[1] as id, cast (columns[2] as int) as grade
2..semicolon> from dfs.`C:/Users/oscar meng/Desktop/data_process/l2/students.csv`
3..semicolon> order by grade
4..semicolon> desc
5..semicolon> limit 2;
+--------------+------------+-------+
|   stu_name   |     id     | grade |
+--------------+------------+-------+
| Aaron Aaberg | 4790329952 | 100   |
| Aaron Aaberg | 4790329952 | 100   |
+--------------+------------+-------+
2 rows selected (60.785 seconds)
apache drill> select columns[0] as stu_name, columns[1] as id, cast (columns[2] as int) as grade
2..semicolon> from dfs.`C:/Users/oscar meng/Desktop/data_process/l2/students.csv`
3..semicolon> order by grade
4..semicolon> asc
5..semicolon> limit 2;
+--------------+------------+-------+
|   stu_name   |     id     | grade |
+--------------+------------+-------+
| Aaron Aaberg | 4790329952 | 0     |
| Aaron Aaberg | 4790329952 | 0     |
+--------------+------------+-------+
```

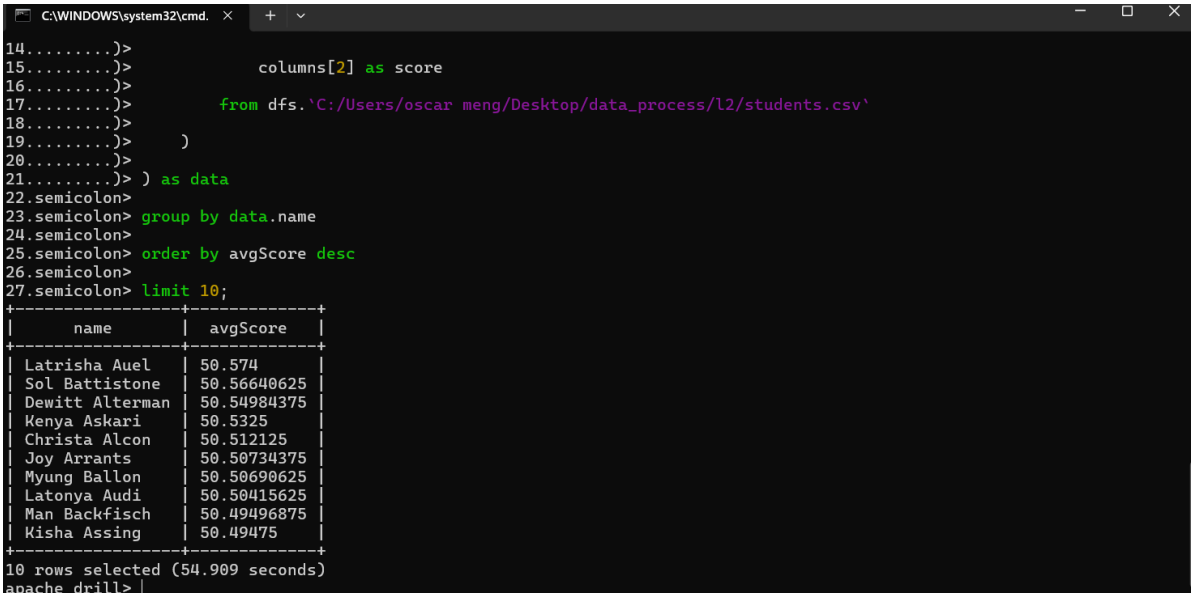
### b)

| name | avgScore |
|---|---|
| Latrisha Auel | 50.574 |

The screen shot of the result:



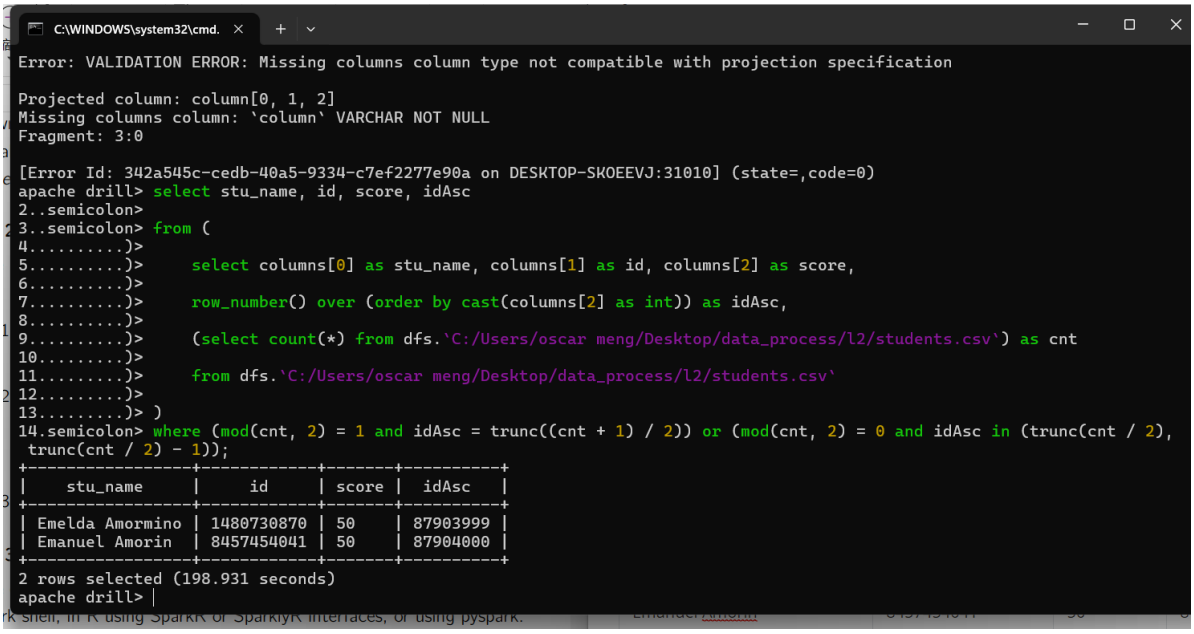
```
Apache Drill 1.21.1
"Got Drill?"
apache drill> select data.name,avg(data.score) as avgScore
2..semicolon>
3..semicolon> from (
4.........)>
5.........)>     select name,score
6.........)>
7.........)>     from (
8.........)>
9.........)>         select
10........)>
11........)>             columns[0] as name,
12........)>
13........)>             columns[1] as id,
14........)>
15........)>             columns[2] as score
16........)>
17........)>         from dfs.`C:/Users/oscar meng/Desktop/data_process/l2/students.csv`
18........)>
19........)>     )
20........)>
21........)> ) as data
22.semicolon>
23.semicolon> group by data.name
24.semicolon>
25.semicolon> order by avgScore desc
26.semicolon>
27.semicolon> limit 10;
```




### 3.

| stu_name | id | score | idAsc |
|---|---|---|---|
| Emanuel Amorin | 8457454041 | 50 | 87904000 |
| Emelda Amormino | 1480730870 | 50 | 87903999 |

The screen shot of the result:




# Ex.3

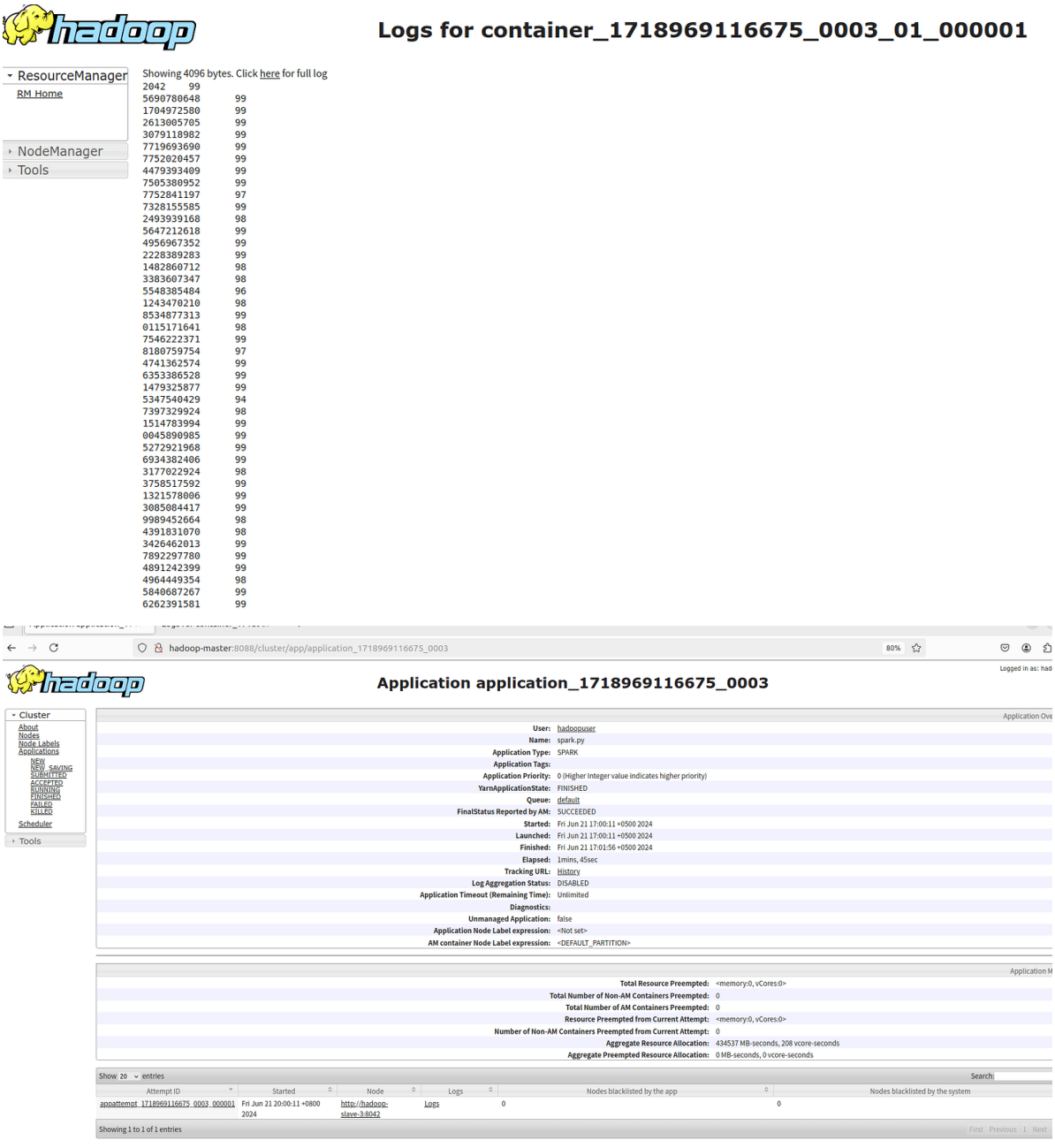
We prepare data of 7.76MB, 175.44MB, 7.4GB.

Here is our code:

```python
from pyspark import sql, SparkConf, SparkContext

spark = sql.session.SparkSession.builder.master("yarn").appName("l4e3").getOrCreate()
rdd = spark.read.text("hdfs://hadoop-master:9000/user/root/input/students_100.csv").rdd

pairs = rdd.flatMap(lambda r: [r[0].split(",")[1:3]])
max_grade = pairs.reduceByKey(lambda x, y: max([x, y]))

for entry in max_grade.collect():
    print("{}\t{}".format(entry[0], entry[1]))
```

The results of the maximum grade and the speed are as follows:

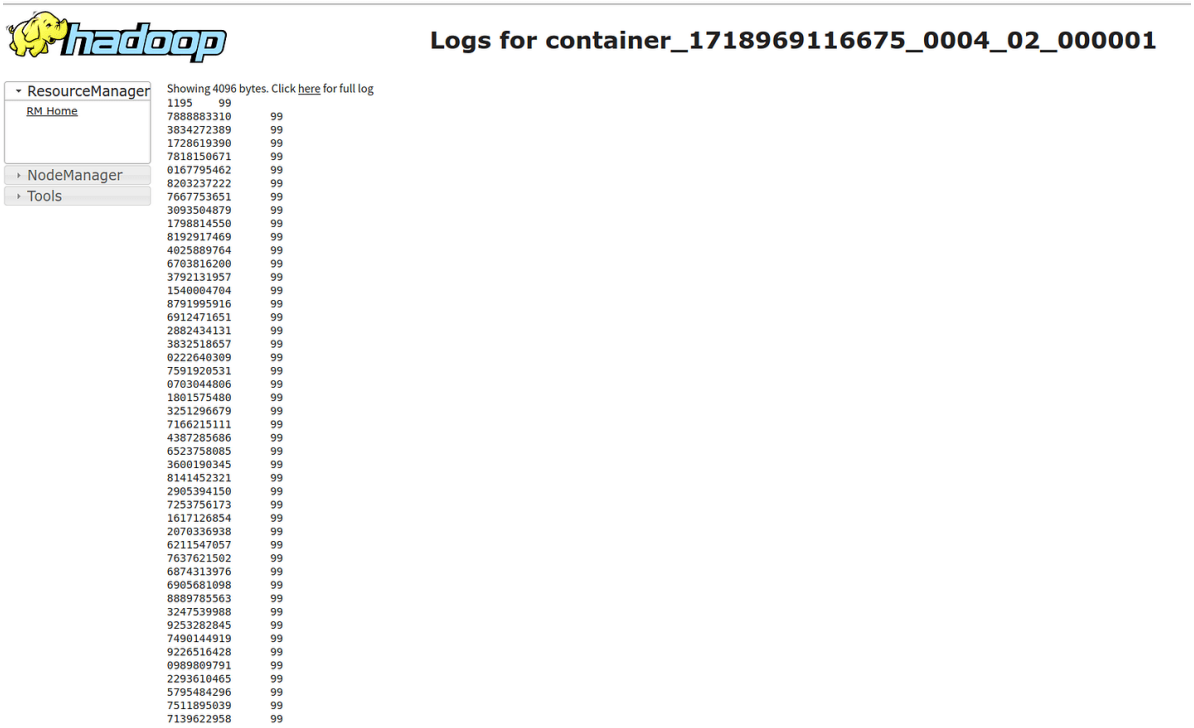
For 7.76MB file:







For 175.44MB file:

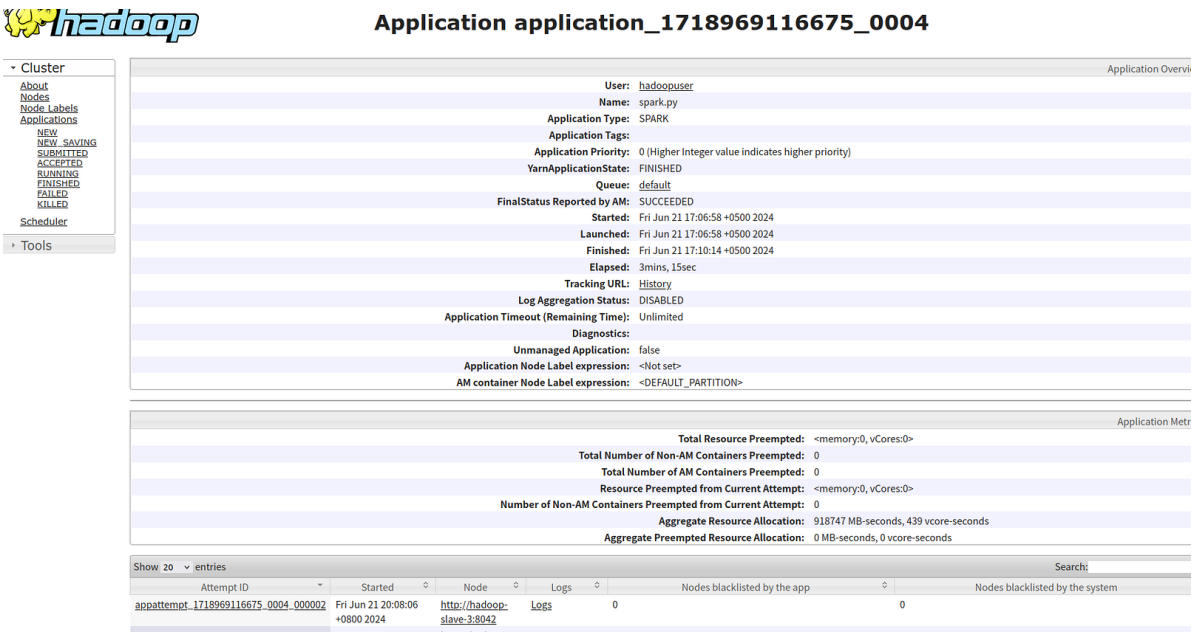


Logs for container_1718969116675_0004_02_000001

ResourceManager
RM Home
NodeManager
Tools

Showing 4096 bytes. Click here for full log

```
1195      99
7888883310      99
3834272389      99
1728619390      99
7818150671      99
0167795462      99
8203237222      99
7667753651      99
3093504879      99
1798814550      99
8192917469      99
4025889764      99
6703816200      99
3792131957      99
1540004704      99
8791995916      99
6912471651      99
2882434131      99
3832518657      99
0222640389      99
7591920531      99
0703044806      99
1801575480      99
3251296679      99
7166215111      99
4387285686      99
6523758085      99
3600190345      99
8141452321      99
2905394150      99
7253756173      99
1617126854      99
2070336938      99
6211547057      99
7637621502      99
6874313976      99
6905681098      99
8889785563      99
3247539988      99
9253282845      99
7490144919      99
9226516428      99
0989809791      99
2293610465      99
5795484296      99
7511895039      99
7139622958      99
```



Application application_1718969116675_0004

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Application Overvi

| | |
|---|---|
| User: | hadoopuser |
| Name: | spark.py |
| Application Type: | SPARK |
| Application Tags: | |
| Application Priority: | 0 (Higher Integer value indicates higher priority) |
| YarnApplicationState: | FINISHED |
| Queue: | default |
| FinalStatus Reported by AM: | SUCCEEDED |
| Started: | Fri Jun 21 17:06:58 +0500 2024 |
| Launched: | Fri Jun 21 17:06:58 +0500 2024 |
| Finished: | Fri Jun 21 17:10:14 +0500 2024 |
| Elapsed: | 3mins, 15sec |
| Tracking URL: | History |
| Log Aggregation Status: | DISABLED |
| Application Timeout (Remaining Time): | Unlimited |
| Diagnostics: | |
| Unmanaged Application: | false |
| Application Node Label expression: | <Not set> |
| AM container Node Label expression: | <DEFAULT_PARTITION> |

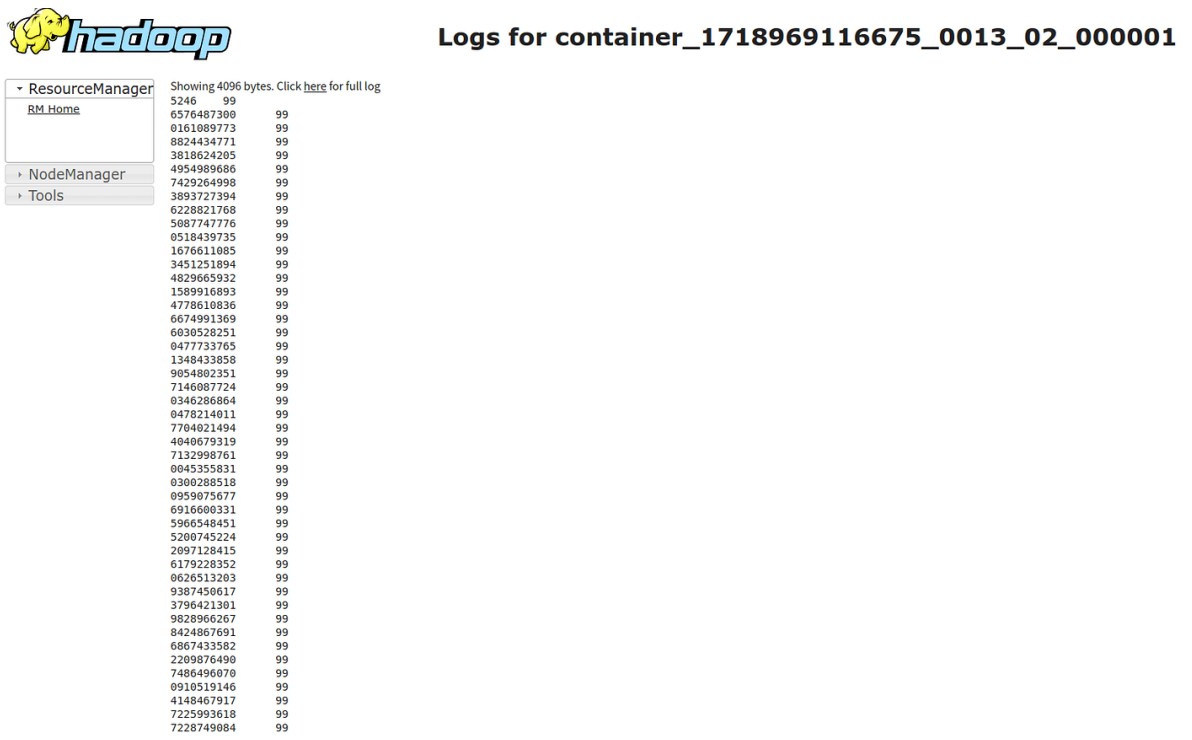
Application Metr

| | |
|---|---|
| Total Resource Preempted: | <memory:0, vCores:0> |
| Total Number of Non-AM Containers Preempted: | 0 |
| Total Number of AM Containers Preempted: | 0 |
| Resource Preempted from Current Attempt: | <memory:0, vCores:0> |
| Number of Non-AM Containers Preempted from Current Attempt: | 0 |
| Aggregate Resource Allocation: | 918747 MB-seconds, 439 vcore-seconds |
| Aggregate Preempted Resource Allocation: | 0 MB-seconds, 0 vcore-seconds |

Show 20 entries Search:

| Attempt ID | Started | Node | Logs | Nodes blacklisted by the app | Nodes blacklisted by the system |
|---|---|---|---|---|---|
| appattempt_1718969116675_0004_000002 | Fri Jun 21 20:08:06 +0800 2024 | http://hadoop-slave-3:8042 | Logs | 0 | 0 |

For 7.4GB file:

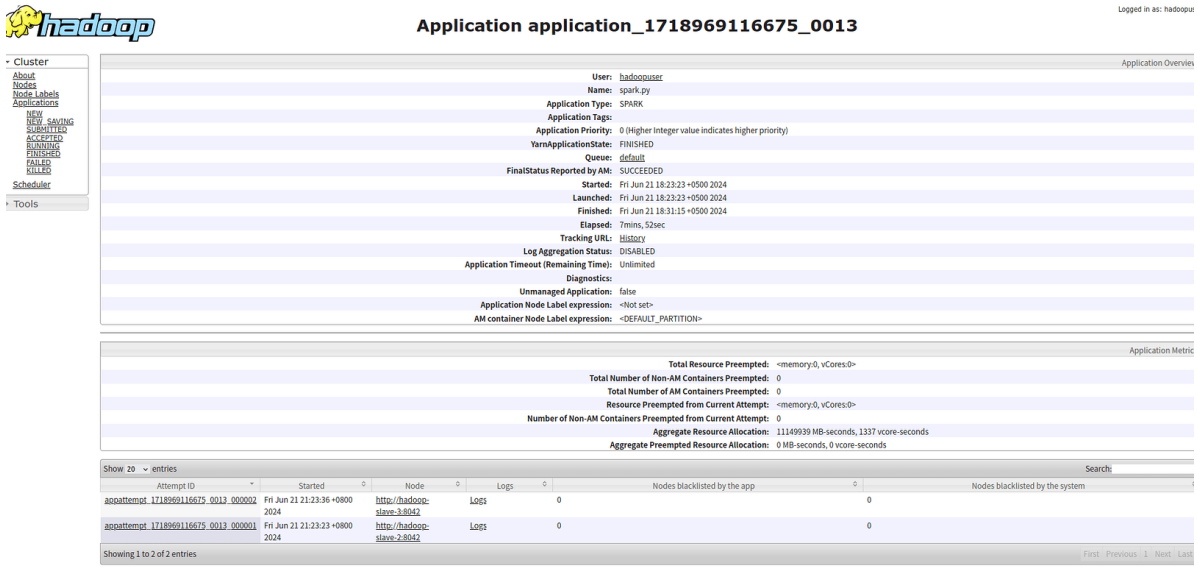


Logs for container_1718969116675_0013_02_000001

ResourceManager
RM Home
NodeManager
Tools

Showing 4096 bytes. Click here for full log

```
5246      99
6576487300      99
0161089773      99
8824434771      99
3818624205      99
4954989686      99
7429264998      99
3893727394      99
6228821768      99
5087747776      99
0518439735      99
1676611085      99
3451251894      99
4829665932      99
1589916893      99
4778610836      99
6674991369      99
6030528251      99
0477733765      99
1348433858      99
9054802351      99
7146087724      99
0346286864      99
0478214011      99
7704021494      99
4040679319      99
7132998761      99
0045355831      99
0300288518      99
0959075677      99
6916600331      99
5966548451      99
5200745224      99
2097128415      99
6179228352      99
0626513203      99
9387450617      99
3796421301      99
9828966267      99
8424867691      99
6867433582      99
2209876490      99
7486496070      99
0910519146      99
4148467917      99
7225993618      99
7228749084      99
```



Application application_1718969116675_0013

Logged in as: hadoopus

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Application Overvie

| | |
|---|---|
| User: | hadoopuser |
| Name: | spark.py |
| Application Type: | SPARK |
| Application Tags: | |
| Application Priority: | 0 (Higher Integer value indicates higher priority) |
| YarnApplicationState: | FINISHED |
| Queue: | default |
| FinalStatus Reported by AM: | SUCCEEDED |
| Started: | Fri Jun 21 18:23:23 +0500 2024 |
| Launched: | Fri Jun 21 18:23:23 +0500 2024 |
| Finished: | Fri Jun 21 18:31:15 +0500 2024 |
| Elapsed: | 7mins, 52sec |
| Tracking URL: | History |
| Log Aggregation Status: | DISABLED |
| Application Timeout (Remaining Time): | Unlimited |
| Diagnostics: | |
| Unmanaged Application: | false |
| Application Node Label expression: | <Not set> |
| AM container Node Label expression: | <DEFAULT_PARTITION> |

Application Metric

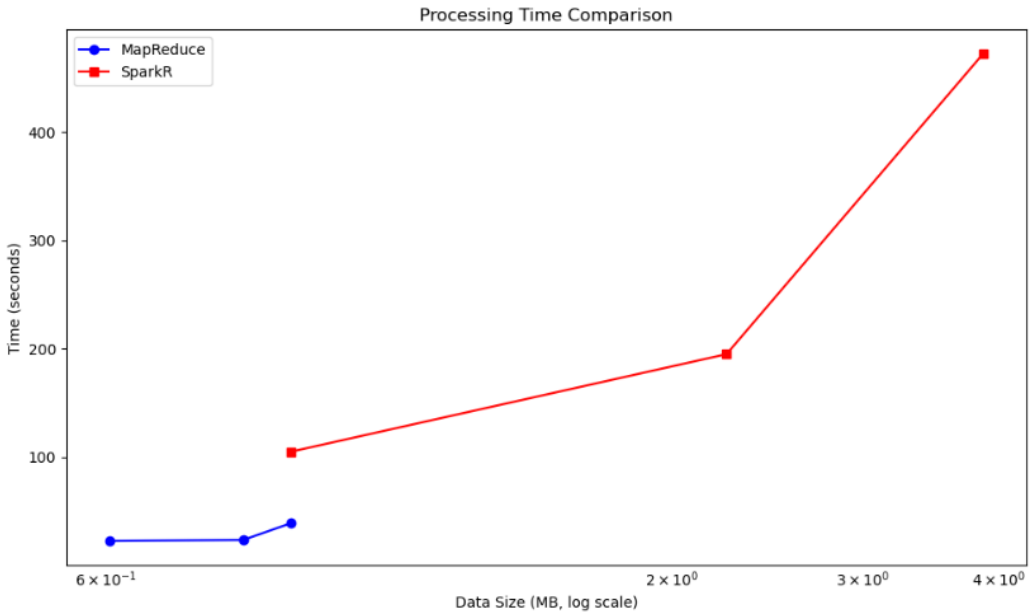
| | |
|---|---|
| Total Resource Preempted: | <memory:0, vCores:0> |
| Total Number of Non-AM Containers Preempted: | 0 |
| Total Number of AM Containers Preempted: | 0 |
| Resource Preempted from Current Attempt: | <memory:0, vCores:0> |
| Number of Non-AM Containers Preempted from Current Attempt: | 0 |
| Aggregate Resource Allocation: | 11149939 MB-seconds, 1337 vcore-seconds |
| Aggregate Preempted Resource Allocation: | 0 MB-seconds, 0 vcore-seconds |

Show 20 entries Search:

| Attempt ID | Started | Node | Logs | Nodes blacklisted by the app | Nodes blacklisted by the system |
|---|---|---|---|---|---|
| appattempt_1718969116675_0013_000002 | Fri Jun 21 21:23:36 +0800 2024 | http://hadoop-slave-3:8042 | Logs | 0 | 0 |
| appattempt_1718969116675_0013_000001 | Fri Jun 21 21:23:23 +0800 2024 | http://hadoop-slave-2:8042 | Logs | 0 | 0 |

Showing 1 to 2 of 2 entries First Previous 1 Next Last

Comparing the result between MapReduce and SparkR:



From the graph, we can see that for smaller files, MapReduce is faster than SparkR. For bigger files, we believe that MapReduce will be slower than SparkR. This is because SparkR is more efficient in handling large datasets due to its in-memory processing capabilities.

| fize size(MB) / time(s) | 4.03 | 6.36 | 7.76 | 175.44 | 7577.6 |
|---|---|---|---|---|---|
| MapReduce | 22.614 | 23.488 | 39.084 | None | None |
| SparkR | None | None | 105 | 195 | 472 |