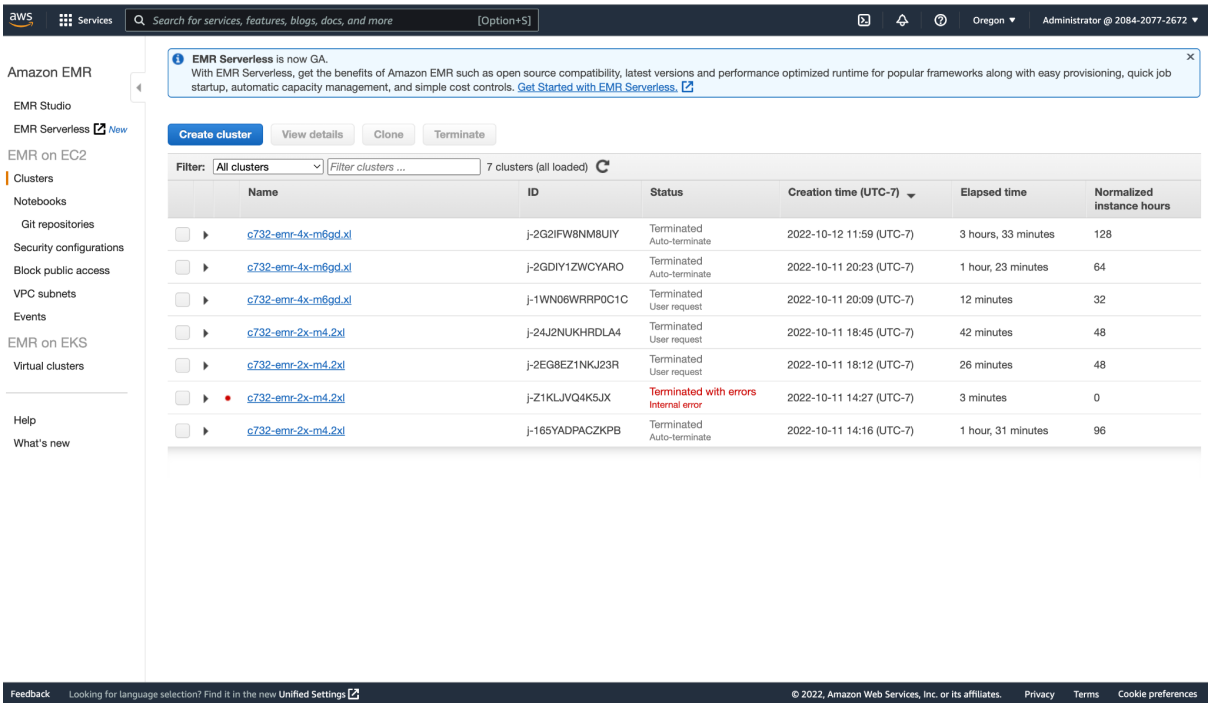
Question 1

Take a screen shot of your list of EMR clusters (if more than one page, only the page with the most recent), showing that all have `Terminated` status.

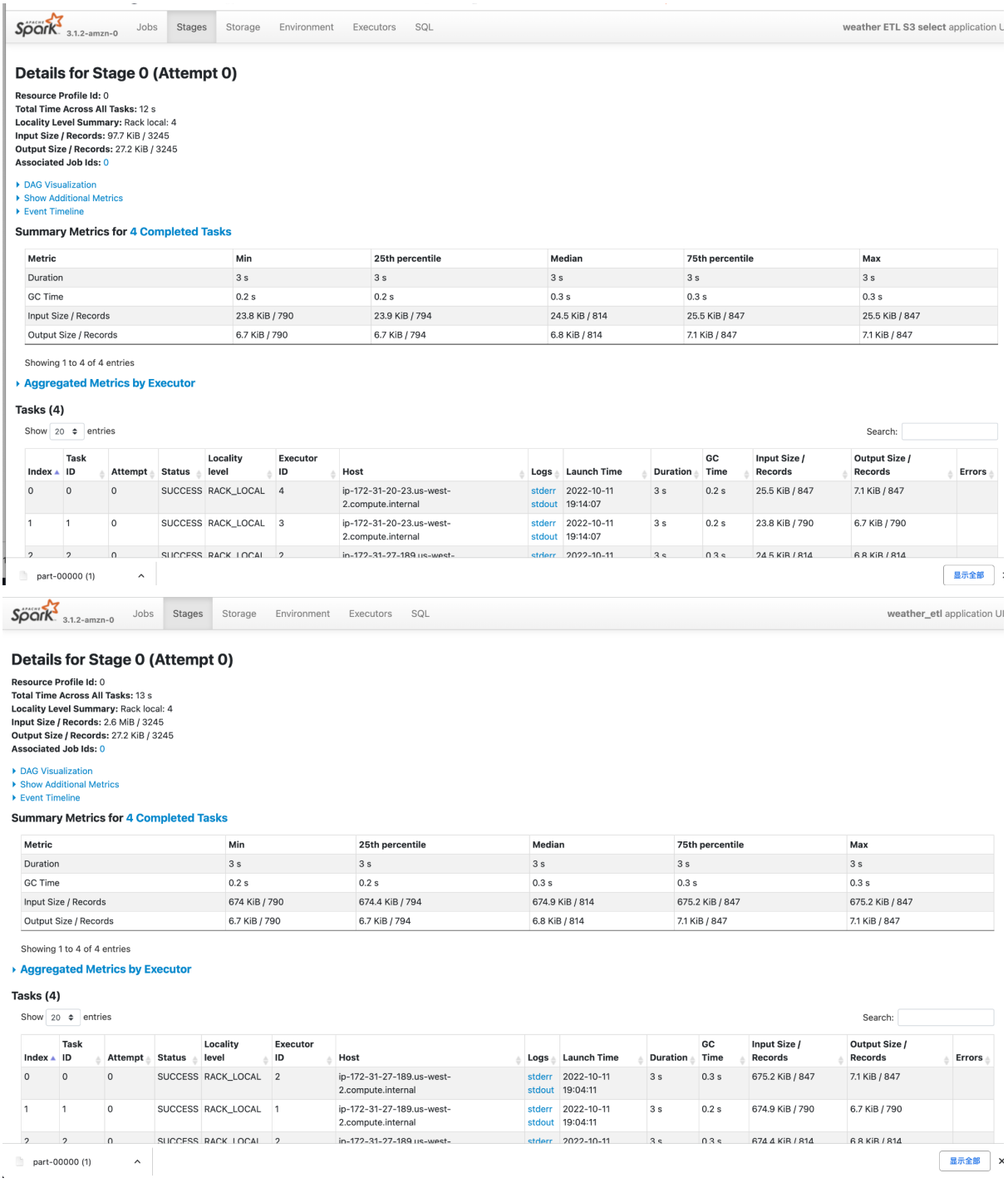
Here's my all clusters that are all terminated.



Question 2

(a) What fraction of the input file was prefiltered by S3 before it was sent to Spark?

We can see that the input size is 97.7 KB and 2.6 MB.
Thus, the fraction is 97.7 / 2.6 * 1024 = 97.7 / 2662.4 = 0.036696213942308 = 3.6696213942308%.
The fraction that input is filtered out is 1 - 0.036696213942308 = 0.963303786057692 = 96.3303786057692%.
Below are the screenshots of the input file size with and without S3 filtering.



(b) Comparing the different input numbers for the regular version versus the prefiltered one, what operations were performed by S3 and which ones performed in Spark?

The S3 is filtering the unnecessary data in each column. What is not used in the weather_etl.py code will be filtered out, such as “sflag” and “obstime”. Only the useful data will be saved, such as “qflag” and “value” and etc. The total number of columns is not eliminated, but the data each column contains is eliminated.
The Spark is doing what the weather_etl.py wants. It does most of the IO. It reads inputs and filter the conditions we need (in assignment 4) and creates the new column and returns the three columns we need and generates the output.

Question 3

a. Reviewing the job times in the Spark history, which operations took the most time? Is the application IO-bound or compute-bound?

The runJob at SparkHadoopWriter took the most time.

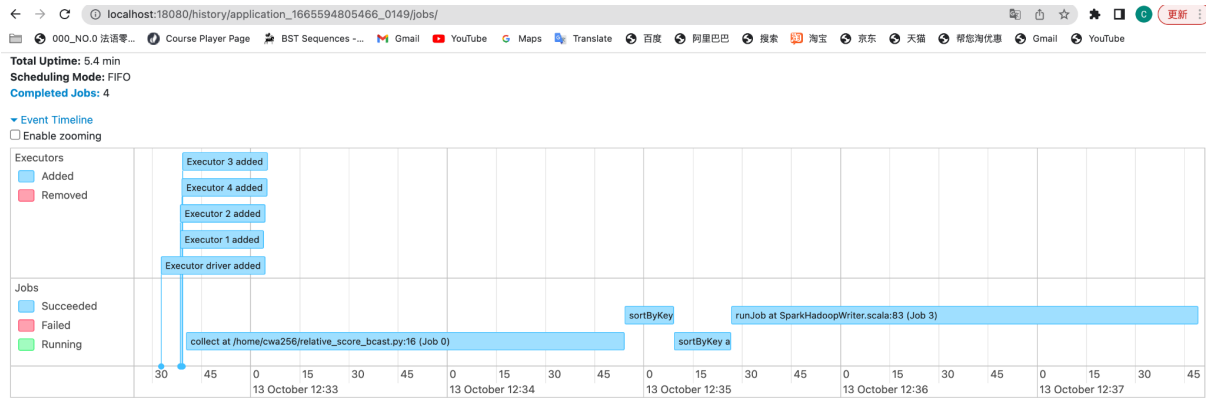
When comparing the time, we can see that in the local cluster, runJob at SparkHadoopWriter costs the most time. And the runJob at SparkHadoopWriter itself needs the most time.

When comparing the time, we can see that in the aws cluster, collect at /home/cwa256/relative_score_bcast.py:16 costs the most time. And the reduceByKey at /home/cwa256/relative_score_bcast.py:14 inside needs the most time. However, the reduceByKey also includes a certain time for the input, which leads us to guess that the runJob at SparkHadoopWriter still costs more because the time difference between those is small.

Thus, the application is IO-bound.

Below are screenshots of the local cluster and the aws cluster.

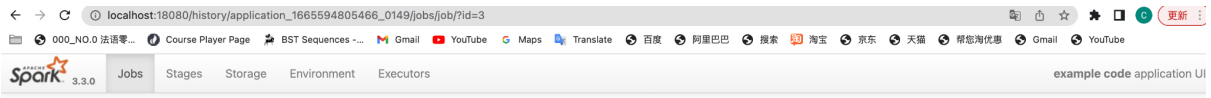
These are local clusters time.



Completed Jobs (4)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83	2022/10/13 12:35:26	2.4 min	2/2	16/16
2	sortByKey at /home/cwa256/relative_score_bcast.py:23 sortByKey at /home/cwa256/relative_score_bcast.py:23	2022/10/13 12:35:09	17 s	1/1	8/8
1	sortByKey at /home/cwa256/relative_score_bcast.py:23 sortByKey at /home/cwa256/relative_score_bcast.py:23	2022/10/13 12:34:54	15 s	1/1	8/8
0	collect at /home/cwa256/relative_score_bcast.py:16 collect at /home/cwa256/relative_score_bcast.py:16	2022/10/13 12:32:40	2.2 min	2/2	16/16



Details for Job 3

Status: SUCCEEDED
Submitted: 2022/10/13 12:35:26
Duration: 2.4 min
Completed Stages: 2

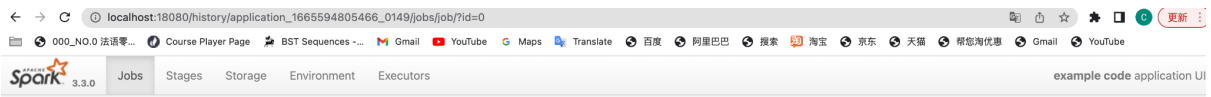
Event Timeline
DAG Visualization

Completed Stages (2)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	runJob at SparkHadoopWriter.scala:83 +details	2022/10/13 12:36:02	1.8 min	8/8		1578.0 MiB	564.4 MiB	
4	sortByKey at /home/cwa256/relative_score_bcast.py:23 +details	2022/10/13 12:35:26	36 s	8/8	649.0 MiB			564.4 MiB

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go



Details for Job 0

Status: SUCCEEDED
Submitted: 2022/10/13 12:32:40
Duration: 2.2 min
Completed Stages: 2

Event Timeline
DAG Visualization

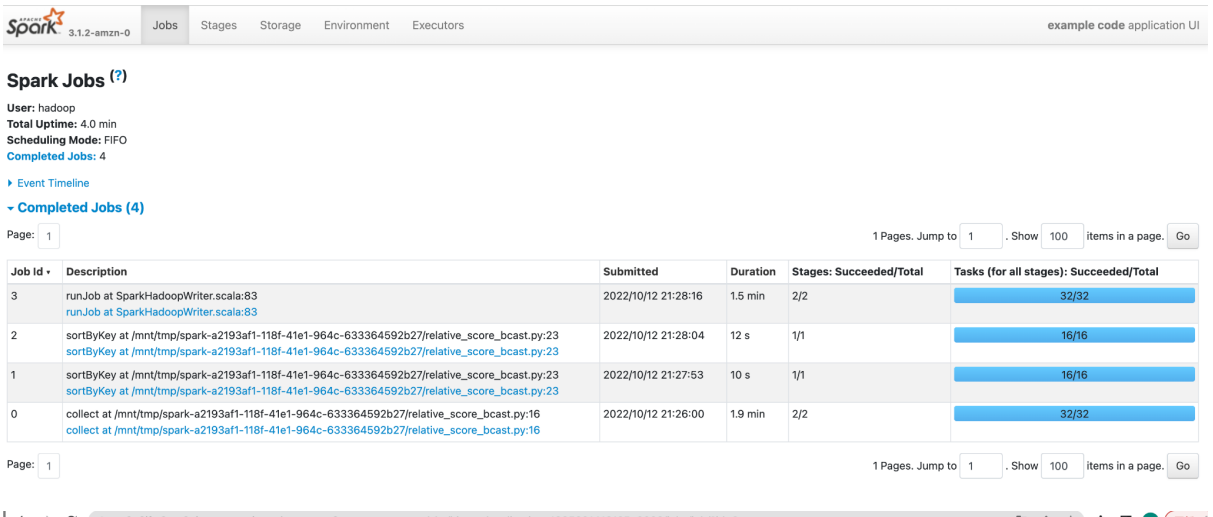
Completed Stages (2)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	collect at /home/cwa256/relative_score_bcast.py:16 +details	2022/10/13 12:34:53	0.5 s	8/8			3.6 KiB	
0	reduceByKey at /home/cwa256/relative_score_bcast.py:14 +details	2022/10/13 12:32:40	2.2 min	8/8	9.0 GiB			3.6 KiB

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

These are aws cluster job times.



Spark 3.1.2-amzn-0 | Jobs | Stages | Storage | Environment | Executors | example code application UI

Spark Jobs (?)

User: hadoop
Total Uptime: 4.0 min
Scheduling Mode: FIFO
Completed Jobs: 4

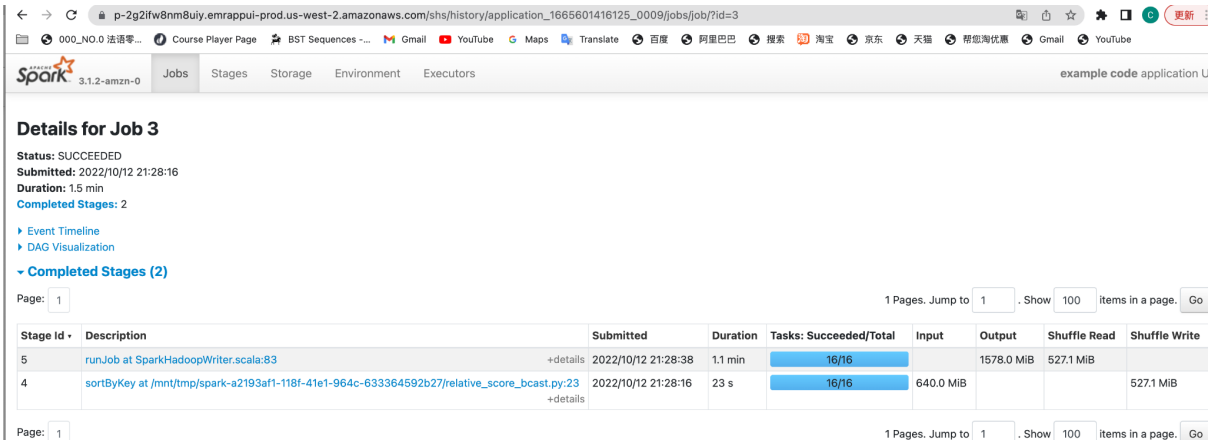
▸ Event Timeline

▾ **Completed Jobs (4)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83	2022/10/12 21:28:16	1.5 min	2/2	32/32
2	sortByKey at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:23 sortByKey at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:23	2022/10/12 21:28:04	12 s	1/1	16/16
1	sortByKey at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:23 sortByKey at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:23	2022/10/12 21:27:53	10 s	1/1	16/16
0	collect at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:16 collect at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:16	2022/10/12 21:26:00	1.9 min	2/2	32/32

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go



p-2g2ifw8nm8uiy.emrappui-prod.us-west-2.amazonaws.com/shs/history/application_1665601416125_0009/jobs/job/?id=3

Spark 3.1.2-amzn-0 | Jobs | Stages | Storage | Environment | Executors | example code application U

Details for Job 3

Status: SUCCEEDED
Submitted: 2022/10/12 21:28:16
Duration: 1.5 min
Completed Stages: 2

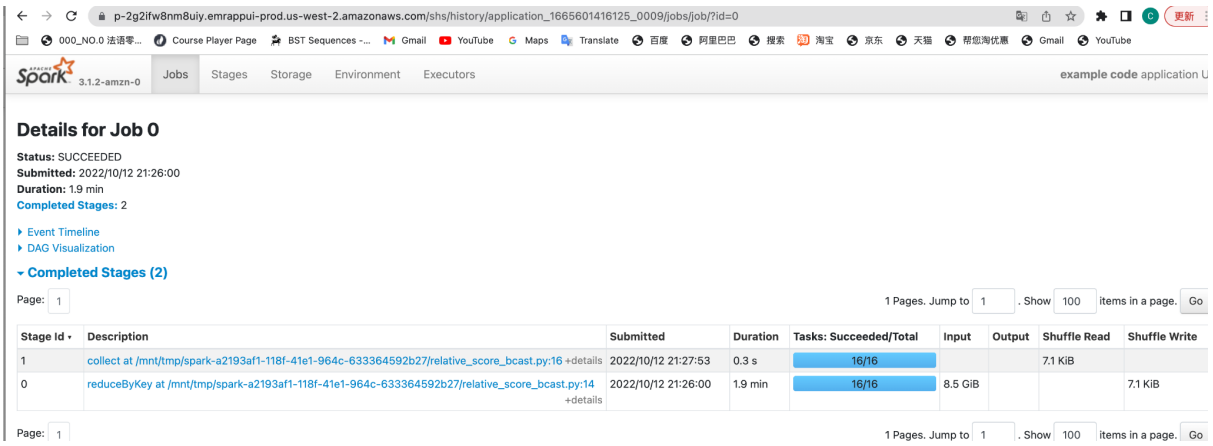
▸ Event Timeline
▸ DAG Visualization

▾ **Completed Stages (2)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	runJob at SparkHadoopWriter.scala:83 +details	2022/10/12 21:28:38	1.1 min	16/16		1578.0 MiB	527.1 MiB	
4	sortByKey at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:23 +details	2022/10/12 21:28:16	23 s	16/16	640.0 MiB			527.1 MiB

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go



p-2g2ifw8nm8uiy.emrappui-prod.us-west-2.amazonaws.com/shs/history/application_1665601416125_0009/jobs/job/?id=0

Spark 3.1.2-amzn-0 | Jobs | Stages | Storage | Environment | Executors | example code application U

Details for Job 0

Status: SUCCEEDED
Submitted: 2022/10/12 21:26:00
Duration: 1.9 min
Completed Stages: 2

▸ Event Timeline
▸ DAG Visualization

▾ **Completed Stages (2)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	collect at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:16 +details	2022/10/12 21:27:53	0.3 s	16/16			7.1 KiB	
0	reduceByKey at /mnt/tmp/spark-a2193af1-118f-41e1-964c-633364592b27/relative_score_bcast.py:14 +details	2022/10/12 21:26:00	1.9 min	16/16	8.5 GiB			7.1 KiB

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

b. Look up the hourly costs of the `m6gd.xlarge` instance on the EC2 On-Demand Pricing page. Estimate the cost of processing a dataset ten times as large as `reddit-5` using just those 4 instances. If you wanted instead to process this larger dataset making full use of 16 instances, how would it have to be organized?

We can use the on-demand hourly rate multiplied by the time multiplied by ten. Then we use the mapreduce hourly rate multiplied by the time multiplied by ten. Then we multiply it by four (instances) and add them up. This is before tax.
For EC2:
$0.1808 * (4 / 60 * 10) = $0.120533
$0.120533 * 4 = $0.482132
For EMR:
$0.0452 * (4 / 60 * 10) = $0.030133
$0.030133 * 4 = $0.120532
In total:
$0.482132 + $0.120532 = $0.602664

To process this larger dataset making full use of 16 instances, we will repartition.
Since we know that the reddit-5 has 8 files, we will repartition it with 16's multiplication times. In this case, I will choose 160 as the repartition time which is in good range. In this way, we can make sure the instances are fully used efficiently.
Also, the broadcast object may need to be reevaluated because it has risk when exceeding its limit size.