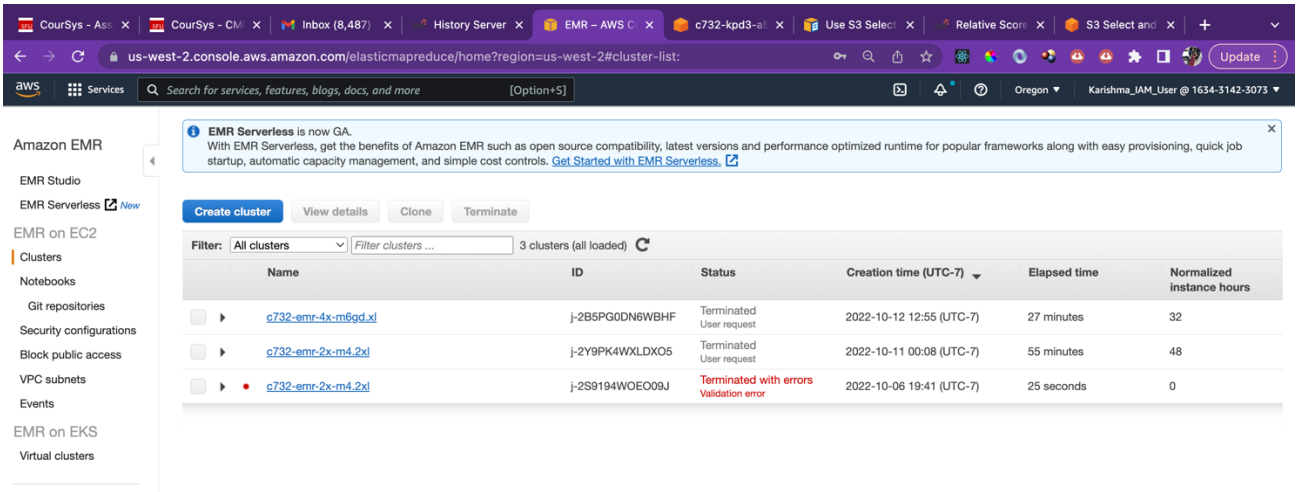
Q1) The terminated clusters are as shown below.



Q2) a) The input size without filtering = 2.6 MiB
The input size with S3 filtering = 97.7 KiB
The fraction of the input file was filtered and sent to Spark = 97.7/2662.4 = 0.0366

Thus the amount that was filtered out by S3Select = 96.34% and the amount that was sent to spark after filtering = 3.66%

Details for Stage 0 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 14 s
Locality Level Summary: Rack local: 4
Input Size / Records: 2.6 MiB / 3245
Output Size / Records: 27.2 KiB / 3245
Associated Job Ids: 0

Without S3 Select

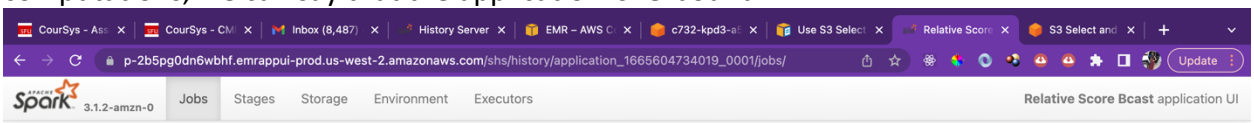
Details for Stage 0 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 10 s
Locality Level Summary: Rack local: 4
Input Size / Records: 97.7 KiB / 3245
Output Size / Records: 27.2 KiB / 3245
Associated Job Ids: 0

With S3 Select

b) The SQL operations such as filtering, where and select, etc are being performed through S3Select whereas the other operations are performed by Spark.

Q3) From the Jobs and DAGs below we can see that the maximal amount of time is taken through the collect job and the runJob, both involve reading from the input and writing into the output, respectively. Since most time is spent in IO operations compared to the rest of the computations, we can say that the application is IO bound.



Spark Jobs (?)

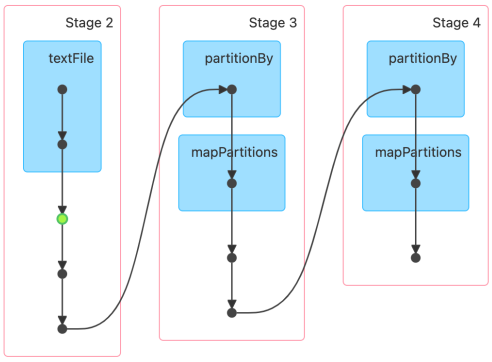
User: hadoop
Total Uptime: 9.5 min
Scheduling Mode: FIFO
Completed Jobs: 6

Event Timeline

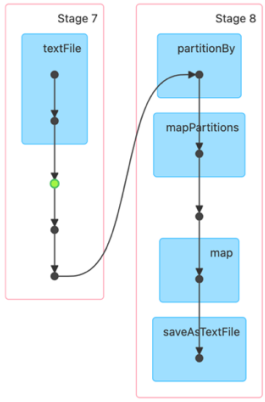
Completed Jobs (6)

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
5	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83	2022/10/12 20:14:32	1.9 min	2/2	32/32
4	sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:59 sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:59	2022/10/12 20:13:50	41 s	1/1	16/16
3	sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:59 sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:59	2022/10/12 20:13:11	39 s	1/1	16/16
2	collect at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:52 collect at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:52	2022/10/12 20:10:31	2.7 min	3/3	48/48
1	sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:48 sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:48	2022/10/12 20:09:56	35 s	1/1	16/16
0	sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:48 sortBy at /mnt/tmp/spark-5b07d63a-4184-433c-a192-45bc1483b0a8/relative_score_bcast.py:48	2022/10/12 20:07:13	2.7 min	1/1	16/16



Collect DAG



runJob DAG

b)

cost of m6gd.xlarge = $0.1808/hour of usage while using 4vCPUs

Since the cost is calculated depending on the time taken, a dataset 10 times the size of reddit-5 would take much more time to execute with the same 4 instances and hence the price would increase accordingly.

In order to process a very large dataset while making use of 16 instances, we could partition the dataset so that the instances can work parallelly with these chunks thus making better use of the available resources.