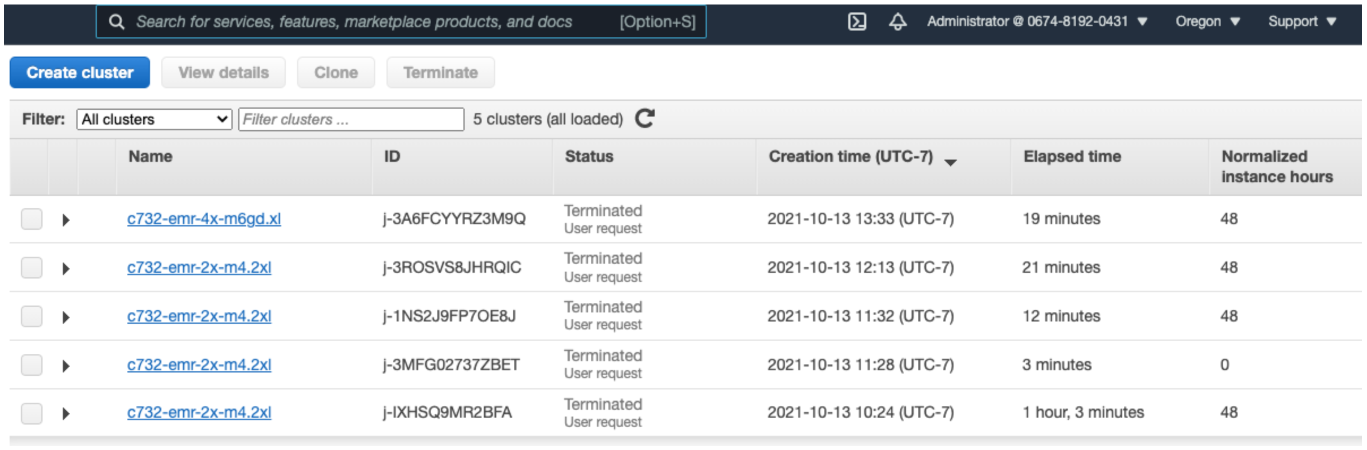
Assignment 5

Wanyi Su
Student # 301445656

Q1.



Name	ID	Status	Creation time (UTC-7)	Elapsed time	Normalized instance hours
c732-emr-4x-m6gd.xl	j-3A6FCYYRZ3M9Q	Terminated User request	2021-10-13 13:33 (UTC-7)	19 minutes	48
c732-emr-2x-m4.2xl	j-3ROSVS8JHRQIC	Terminated User request	2021-10-13 12:13 (UTC-7)	21 minutes	48
c732-emr-2x-m4.2xl	j-1NS2J9FP7OE8J	Terminated User request	2021-10-13 11:32 (UTC-7)	12 minutes	48
c732-emr-2x-m4.2xl	j-3MFG02737ZBET	Terminated User request	2021-10-13 11:28 (UTC-7)	3 minutes	0
c732-emr-2x-m4.2xl	j-IXHSQ9MR2BFA	Terminated User request	2021-10-13 10:24 (UTC-7)	1 hour, 3 minutes	48

Q2.

a).

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

As the screenshots show below, input size is 97.7 KiB after S3 select prefiltering compared to 2.6MiB before it was added to the code.

Fraction that after filtering: 97.7 KiB / 2.6 MiB = 3.67%
The fraction being filtering away: 1- 3.67% = 96.33%
Since S3 Select can execute SQL queries directly on data stored in Amazon S3 bucket and retrieve the subset of data for our need, input file was prefiltered by S3 when reading the csv file before it was sent to Spark.

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?

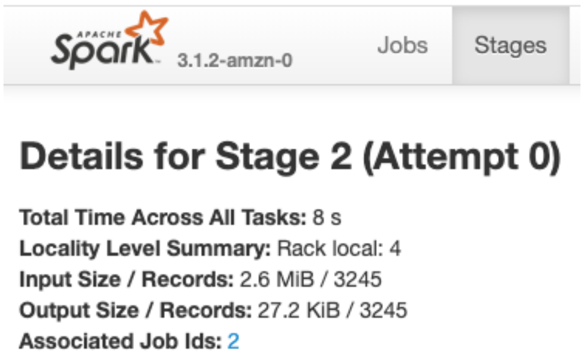
As the screenshots show below, input size is 97.7 KiB after S3 select prefiltering compared to 2.6MiB before it was added to the code. The amount of input data is greatly decreased.

S3 reads the data, performs the **filter** (weather.filter((weather['qflag'].isNull()) & (weather['station'].startswith('CA')) & (weather['observation'] == 'TMAX'))) and **select** function (weather.select('station', 'date', (weather['value']/10).alias('tmax'))) in code for input, and then send the filtered data to Spark. Among this **select** clause, the "weather['value']/10" computation part is done by Spark and send back to S3.
According to our code, these parts of input are prefiltered (which means included in input sent to Spark):

1. field 'qflag' (quality flag) that is null
2. field 'station' which starts with 'CA'
3. field 'observation' that is 'TMAX'

Spark implements calculation of dividing the temperature by 10 and writes the final output to the output file.

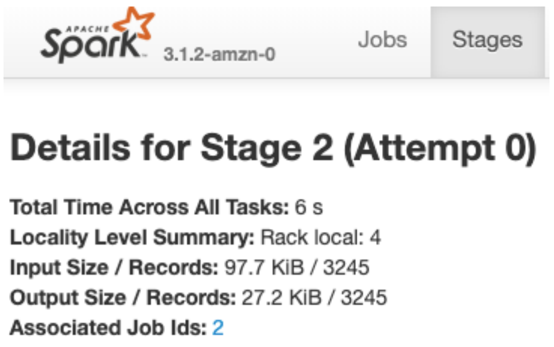
Before S3 filtering:



Details for Stage 2 (Attempt 0)

Total Time Across All Tasks: 8 s
Locality Level Summary: Rack local: 4
Input Size / Records: 2.6 MiB / 3245
Output Size / Records: 27.2 KiB / 3245
Associated Job Ids: 2

After S3 filter:



Details for Stage 2 (Attempt 0)

Total Time Across All Tasks: 6 s
Locality Level Summary: Rack local: 4
Input Size / Records: 97.7 KiB / 3245
Output Size / Records: 27.2 KiB / 3245
Associated Job Ids: 2

Q3.

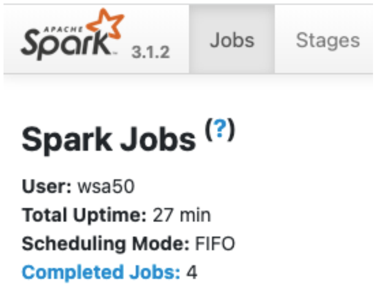
a).

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

We spent the most of time on IO-bound. On local cluster, the time is about 14.7 mins (8.9min+5.8min) in input reading and output writing stages (job 0 and 3 here). (see screenshots below)
On local cluster, SortBy process in writing process costs the most time. While on EMR cluster, reduceByKey in reading stage costs the most time. In general, read/write uses the most time.
The compute bound costs a time of 11.9 mins (6.2m+5.7m), which is lower than IO-bound time.
The situation is similar running on EMR cluster.

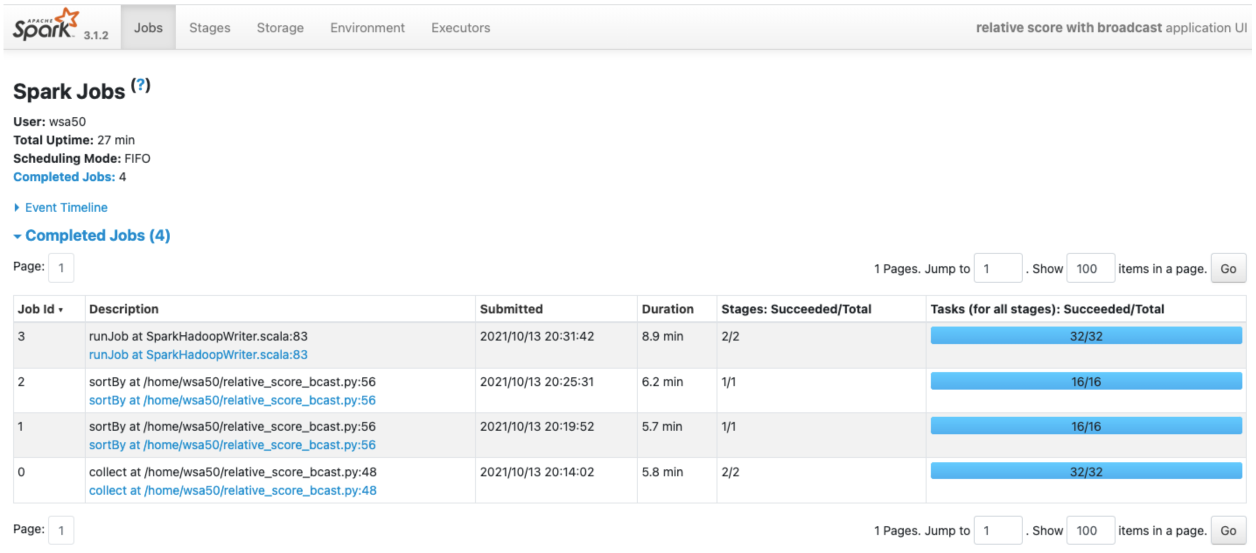
I/O bound: a condition where the time spending to complete a computation is determined primarily by the time waiting for input/output operations to be completed.
Compute bound (CPU bound): the time for it to complete a task is determined principally by the speed of the central processor.

On local cluster:



Spark Jobs (?)

User: wsa50
Total Uptime: 27 min
Scheduling Mode: FIFO
Completed Jobs: 4



Spark Jobs (?)

User: wsa50
Total Uptime: 27 min
Scheduling Mode: FIFO
Completed Jobs: 4

Event Timeline

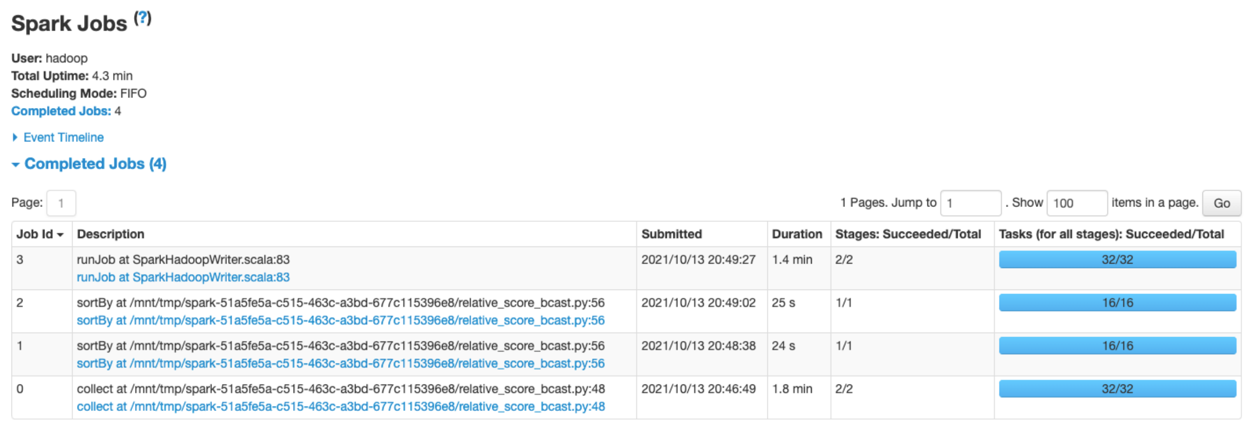
Completed Jobs (4)

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	2021/10/13 20:31:42	8.9 min	2/2	32/32
2	sortBy at /home/wsa50/relative_score_bcast.py:56 sortBy at /home/wsa50/relative_score_bcast.py:56	2021/10/13 20:25:31	6.2 min	1/1	16/16
1	sortBy at /home/wsa50/relative_score_bcast.py:56 sortBy at /home/wsa50/relative_score_bcast.py:56	2021/10/13 20:19:52	5.7 min	1/1	16/16
0	collect at /home/wsa50/relative_score_bcast.py:48 collect at /home/wsa50/relative_score_bcast.py:48	2021/10/13 20:14:02	5.8 min	2/2	32/32

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

On EMR cluster:


Spark Jobs (?)

User: hadoop
Total Uptime: 4.3 min
Scheduling Mode: FIFO
Completed Jobs: 4

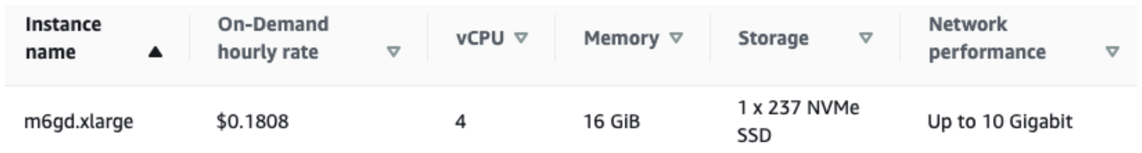
▸ Event Timeline
▾ Completed Jobs (4)

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	2021/10/13 20:49:27	1.4 min	2/2	32/32
2	sortBy at /mnt/tmp/spark-51a5fe5a-c515-463c-a3bd-677c115396e8/relative_score_bcast.py:56 sortBy at /mnt/tmp/spark-51a5fe5a-c515-463c-a3bd-677c115396e8/relative_score_bcast.py:56	2021/10/13 20:49:02	25 s	1/1	16/16
1	sortBy at /mnt/tmp/spark-51a5fe5a-c515-463c-a3bd-677c115396e8/relative_score_bcast.py:56 sortBy at /mnt/tmp/spark-51a5fe5a-c515-463c-a3bd-677c115396e8/relative_score_bcast.py:56	2021/10/13 20:48:38	24 s	1/1	16/16
0	collect at /mnt/tmp/spark-51a5fe5a-c515-463c-a3bd-677c115396e8/relative_score_bcast.py:48 collect at /mnt/tmp/spark-51a5fe5a-c515-463c-a3bd-677c115396e8/relative_score_bcast.py:48	2021/10/13 20:46:49	1.8 min	2/2	32/32

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?



Instance name ▲	On-Demand hourly rate ▽	vCPU ▽	Memory ▽	Storage ▽	Network performance ▽
m6gd.xlarge	$0.1808	4	16 GiB	1 x 237 NVMe SSD	Up to 10 Gigabit

Since the dataset is 10 times larger than reddit-5 and we spend 4.3min running it on EMR cluster, the cost can be 4.3 / 60 * 0.1808 * 4 * 10 = $0.5183

Since now we have 16 instances and each with vCPU of 4, we can repartition the RDD into 64 partitions or more (select a reasonable range, for example, 64 to 128 partitions) to facilitate the running process.