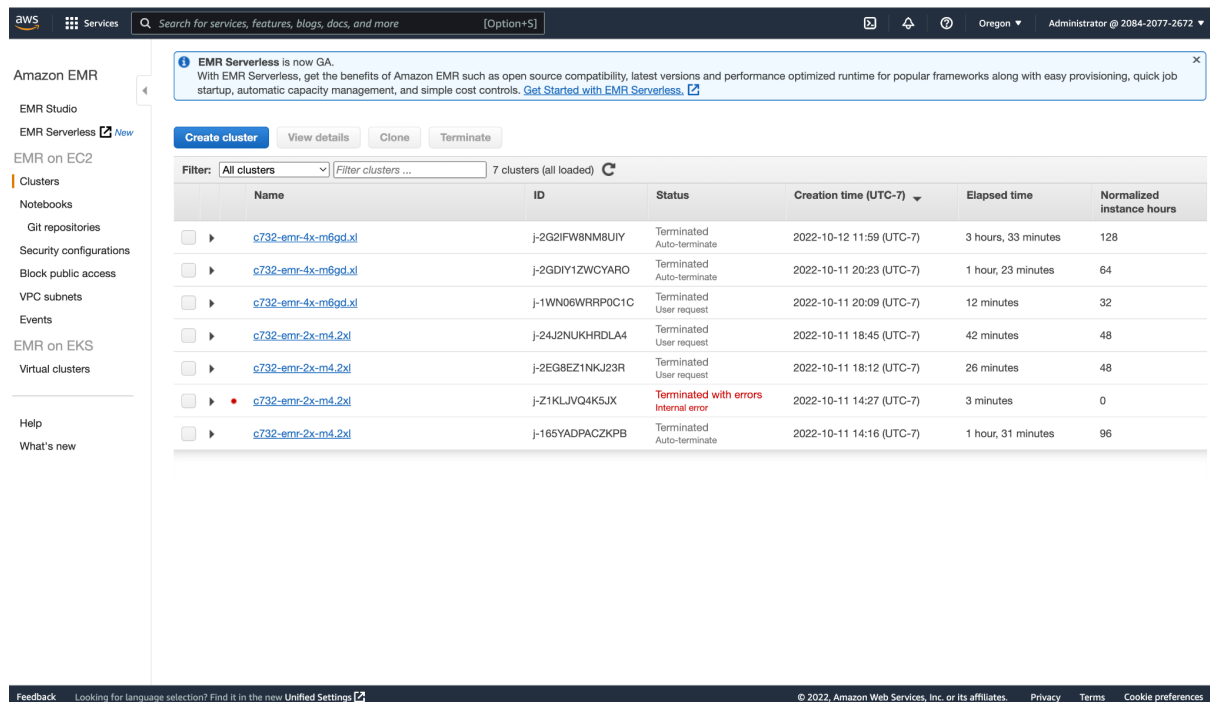


Question 1

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.



Amazon EMR

EMR Studio

EMR Serverless [New](#)

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on EKS

Virtual clusters

Help

What's new

EMR Serverless is now GA. With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime for popular frameworks along with easy provisioning, quick job startup, automatic capacity management, and simple cost controls. [Get Started with EMR Serverless.](#)

Create cluster View details Clone Terminate

Filter: All clusters Filter clusters ... 7 clusters (all loaded)

	Name	ID	Status	Creation time (UTC-7)	Elapsed time	Normalized instance hours
<input type="checkbox"/>	c732-emr-4x-m6gd.xl	j-2G2IFW8NM8UIY	Terminated Auto-terminate	2022-10-12 11:59 (UTC-7)	3 hours, 33 minutes	128
<input type="checkbox"/>	c732-emr-4x-m6gd.xl	j-2GDIY1ZWCYARO	Terminated Auto-terminate	2022-10-11 20:23 (UTC-7)	1 hour, 23 minutes	64
<input type="checkbox"/>	c732-emr-4x-m6gd.xl	j-1WN06WRRP0C1C	Terminated User request	2022-10-11 20:09 (UTC-7)	12 minutes	32
<input type="checkbox"/>	c732-emr-2x-m4.2xl	j-24J2NUKHRDLA4	Terminated User request	2022-10-11 18:45 (UTC-7)	42 minutes	48
<input type="checkbox"/>	c732-emr-2x-m4.2xl	j-2EG8EZ1NKJ23R	Terminated User request	2022-10-11 18:12 (UTC-7)	26 minutes	48
<input type="checkbox"/>	c732-emr-2x-m4.2xl	j-Z1KLJVQ4K5JX	Terminated with errors Internal error	2022-10-11 14:27 (UTC-7)	3 minutes	0
<input type="checkbox"/>	c732-emr-2x-m4.2xl	j-165YADPACZKPB	Terminated Auto-terminate	2022-10-11 14:16 (UTC-7)	1 hour, 31 minutes	96

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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.

3.1.2-amzn-0

Jobs

Stages

Storage

Environment

Executors

SQL

weather ETL S3 select application UI

Details for Stage 0 (Attempt 0)

Resource Profile Id: 0

Total Time Across All Tasks: 12 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

DAG Visualization

Show Additional Metrics

Event Timeline

Summary Metrics for 4 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	3 s	3 s	3 s	3 s	3 s
GC Time	0.2 s	0.2 s	0.3 s	0.3 s	0.3 s
Input Size / Records	23.8 KiB / 790	23.9 KiB / 794	24.5 KiB / 814	25.5 KiB / 847	25.5 KiB / 847
Output Size / Records	6.7 KiB / 790	6.7 KiB / 794	6.8 KiB / 814	7.1 KiB / 847	7.1 KiB / 847

Showing 1 to 4 of 4 entries

Aggregated Metrics by Executor

Tasks (4)

Show 20 entries

Search:

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Input Size / Records	Output Size / Records	Errors
0	0	0	SUCCESS	RACK_LOCAL	4	ip-172-31-20-23.us-west-2.compute.internal	stderr stdout	2022-10-11 19:14:07	3 s	0.2 s	25.5 KiB / 847	7.1 KiB / 847	
1	1	0	SUCCESS	RACK_LOCAL	3	ip-172-31-20-23.us-west-2.compute.internal	stderr stdout	2022-10-11 19:14:07	3 s	0.2 s	23.8 KiB / 790	6.7 KiB / 790	
2	2	0	SUCCESS	RACK_LOCAL	2	ip-172-31-27-189.us-west-2.compute.internal	stderr stdout	2022-10-11 19:04:11	3 s	0.3 s	674.4 KiB / 794	6.7 KiB / 794	

part-00000 (1)

显示全部

3.1.2-amzn-0

Jobs

Stages

Storage

Environment

Executors

SQL

weather_etl application UI

Details for Stage 0 (Attempt 0)

Resource Profile Id: 0

Total Time Across All Tasks: 13 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

DAG Visualization

Show Additional Metrics

Event Timeline

Summary Metrics for 4 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	3 s	3 s	3 s	3 s	3 s
GC Time	0.2 s	0.2 s	0.3 s	0.3 s	0.3 s
Input Size / Records	674 KiB / 790	674.4 KiB / 794	674.9 KiB / 814	675.2 KiB / 847	675.2 KiB / 847
Output Size / Records	6.7 KiB / 790	6.7 KiB / 794	6.8 KiB / 814	7.1 KiB / 847	7.1 KiB / 847

Showing 1 to 4 of 4 entries

Aggregated Metrics by Executor

Tasks (4)

Show 20 entries

Search:

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Input Size / Records	Output Size / Records	Errors
0	0	0	SUCCESS	RACK_LOCAL	2	ip-172-31-27-189.us-west-2.compute.internal	stderr stdout	2022-10-11 19:04:11	3 s	0.3 s	675.2 KiB / 847	7.1 KiB / 847	
1	1	0	SUCCESS	RACK_LOCAL	1	ip-172-31-27-189.us-west-2.compute.internal	stderr stdout	2022-10-11 19:04:11	3 s	0.2 s	674.9 KiB / 790	6.7 KiB / 790	
2	2	0	SUCCESS	RACK_LOCAL	2	ip-172-31-27-189.us-west-2.compute.internal	stderr stdout	2022-10-11 19:04:11	3 s	0.3 s	674.4 KiB / 814	6.8 KiB / 814	

part-00000 (1)

显示全部

(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.

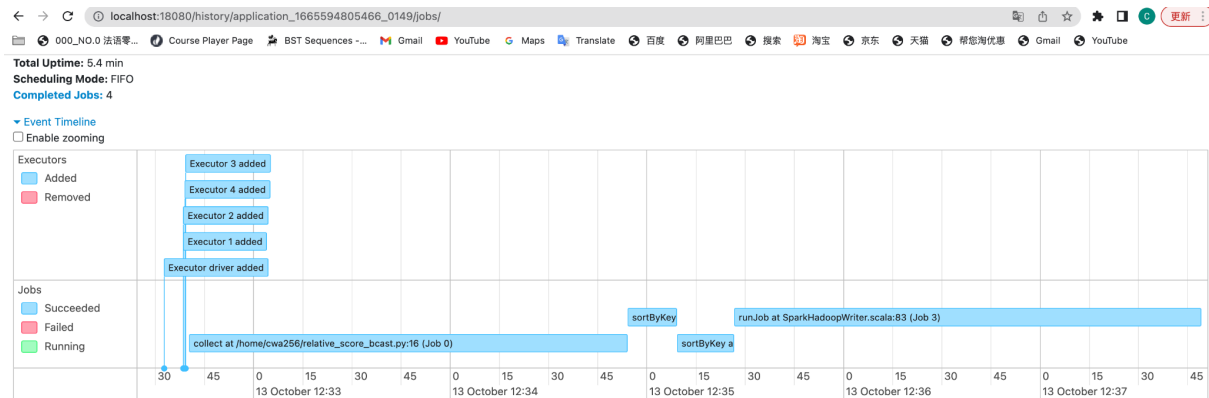
When comparing the time, we can see that in the local cluster, runJob at SparkHadoopWriter costs the most time. And the runJob at SparkHadoopWriter itself needs the most time.

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: 11 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

← → ↻

localhost:18080/history/application_1665594805466_0149/jobs/?id=3

🔍 📄 ⚙️ ⭐ 🔒 🔄 (更新)

000_NO.0 法语零...

Course Player Page

BST Sequences - ...

Gmail

YouTube

Maps

Translate

百度

阿里巴巴

搜索

淘宝

京东

天猫

帮您淘优惠

Gmail

YouTube

Spark 3.3.0

Jobs

Stages

Storage

Environment

Executors

example code application UI

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	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	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

← → ↻

localhost:18080/history/application_1665594805466_0149/jobs/job/?id=0

🔍 📄 ⚙️ ⭐ 🔒 🔄 (更新)

000_NO.0 法语零...

Course Player Page

BST Sequences - ...

Gmail

YouTube

Maps

Translate

百度

阿里巴巴

搜索

淘宝

京东

天猫

帮您淘优惠

Gmail

YouTube

Spark 3.3.0

Jobs

Stages

Storage

Environment

Executors

example code application UI

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


3.1.2-amzn-0

[Jobs](#)
[Stages](#)
[Storage](#)
[Environment](#)
[Executors](#)

[example code](#)
[application UI](#)

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

▼ Completed Jobs (4)

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

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


3.1.2-*amzn*-0

[Jobs](#)
[Stages](#)
[Storage](#)
[Environment](#)
[Executors](#)

[example code](#)
[application U](#)

▼ Completed Stages (2)

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

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



[Jobs](#)
[Stages](#)
[Storage](#)
[Environment](#)
[Executors](#)
[example code application U](#)

▼ Completed Stages (2)

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

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.