**Q1. 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.**

**Graphical user interface, application

Description automatically generated**

**For Section 2:**

**Q2. What fraction of the input file was prefiltered by S3 before it was sent to Spark?**

The screenshot of the regular version is as below:

**Graphical user interface, table

Description automatically generated**

The screenshot of the prefiltered version is as below:

**Graphical user interface

Description automatically generated with medium confidence**

We can find that the input size of the regular version is 2.6MiB; while the input size of the prefiltered version is only 97.7KiB. Thus, (2.6 – 97.7/1024) = 2.50MiB data was prefiltered by S3 before it was sent to Spark, that about 96.3% of the original dataset.

**Q3. 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 reader, three filter functions, and the select function were performed by S3; and the column value computing operation were performed in Spark.

**For Section 3:**

**Q4. Reviewing the job times in the Spark history, which operations took the most time? Is the application IO-bound or compute-bound?**

The screenshot of running on SFU cluster:

**Graphical user interface, application

Description automatically generated**

**Table

Description automatically generated with medium confidence**

The screenshot of running on AWS:

**Graphical user interface, text, application

Description automatically generated**

Graphical user interface, application

Description automatically generated

As shown in the screenshots, the compute-bound (reduceByKey, sortBy) time in the SFU cluster is around (4.4+4.3+4.4+4.9) = 18 min, and the IO operation (writer) time is around 3.1min; in the AWS, the compute-bound time is around (1.8+(21+22+32)/60) = 2.9min, and the IO operation (writer) time is around 1.3min. Thus, seems the compute-bound took more time in this case. But on the other hand, within the reduceByKey stage, the IO operation and the compute operation can exist at the same time, and we cannot find the exact time-cost separately, so it’s hard to say whether the compute-bound or IO bound would take more time. Besides, the time cost also depends on the size of dataset, the compute-related code amount, etc., not just the cluster setting.

**Q5. 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?**

The screenshot of EC2 on-demand pricing page of m6gd.xlarge is as following:

Graphical user interface, application, table

Description automatically generated

Thus, we can estimate the cost of processing a dataset ten times as large as reddit-5 using 4 instances: 0.1808 \* 4 \* (4.5/60) \* 10 = 0.5424 dollars

If we use 16 instances, then we would have 64 cores to process data. According to our prior experiment results, we can get a better performance when the partitions are around two or three times of cores. Thus, we can organize the dataset into 128-192 files before inputting it, or we can use the repartition function to help reorganize the RDD.