

Assignment 5

Introduction

In this assignment, you will run Spark jobs on Amazon Web Service's (AWS) Elastic MapReduce (EMR) service, a cluster computing facility in AWS cloud. It was named for MapReduce as it was the original 'feature' but the service has continually expanded to cover emerging workload/apps/frameworks. Spark was added in EMR 4 and has since tracked the evolution of Spark; you can transfer the lessons you learned on the local cluster directly over to AWS.

Some advantages of an EMR cluster over a local cluster:

- › It can read directly from S3 object storage, a widely-used AWS service.
- › Many organizations already store their data on AWS, so running the analytics there as well makes sense.
- › It is optimized for a cloud environment.
- › It integrates with other AWS services (metrics, machine learning, queuing, ...)
- › It scales on demand—pay for what you need, when you use it.
- › It features specialized hardware that can accelerate tasks such as machine learning.

Some disadvantages:

- › It has a complex console for interactive use.
- › You have to turn clusters off when not in use if you do not want to pay for idle capacity.

Although we will focus on Amazon's service, similar tools exist on other major cloud vendors:

- › **Microsoft Azure HDInsight** (<https://azure.microsoft.com/en-us/services/hdinsight/#overview>) ,
- › **Google Dataproc** (<https://cloud.google.com/dataproc>) , and
- › **Alibaba E-MapReduce** (<https://www.alibabacloud.com/product/emapreduce>) .

In addition, Databricks, the company founded by the creator of Spark, offers a managed Spark service that **runs on all the above vendors** (<https://databricks.com/company/partners/cloud-partners>) .

One important point that we will return to several times in this assignment: When an EMR cluster (or indeed any cloud computing resource) is running, you are charged for every minute, whether or not it is performing any productive work for you in your application. When you are done working with a cluster, either to take a break during the assignment or when you are completely done, **terminate the cluster to stop the charges**. We will describe how to do that in a later section.

Prerequisite 1: Code that you will need from Assignment 4

Before starting this assignment, you will need the following programs from Assignment 4:

1. `relative_score_bcast.py`
2. `weather_etl.py`

Both should be correct, of course. If you have doubts, test them on the local cluster before running them on EMR.

Prerequisite 2: Tunnelling Spark History for the local cluster

In Step 3 of the Assignment, you will look up the Spark History for an application run on the local cluster. In order to do this you will need to configure tunnelling for this feature.

If you have not already done so, add the following line to your entry in `~/.ssh/config` for the local cluster:

```
LocalForward 18080 controller.local:18080
```

Preparing your S3 bucket

Most EMR applications take their input from **Amazon S3** (<https://aws.amazon.com/s3/>), the object storage service for AWS. S3 can inexpensively store very large datasets and is often used for **data lakes on AWS** (<https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/>). Analytics tools such as **Amazon Redshift** (<https://aws.amazon.com/redshift/>), **Amazon EMR Spark** (<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark.html>), and **Snowflake** (<https://docs.snowflake.com/en/user-guide/data-load-overview.html#supported-file-locations>) can all take input from S3. We will use EMR Spark in this assignment.

The top-level unit of S3 is a bucket, which is located in an **AWS Region** (https://aws.amazon.com/about-aws/global-infrastructure/regions_az/). Unlike many other AWS services, an S3 bucket is not further localized by Availability Zone within a Region.

There is one further complication: S3 bucket names are *globally unique*. And when Amazon says "global", they mean it. If you successfully create a bucket named `my-oh-so-marvellous-bucket-yes-yes`, no one else can use that bucket name. Not any other user. Not in any region. So long as you have that bucket name, no one else can use it *anywhere in the world*. (There are some specialized AWS "partitions", such as the one for the US Government, that have their own namespace but virtually all applications run in the general partition that we will use.)

For this assignment, we will write `BUCKET` anywhere you need to place your bucket name. For example, if we ask you to submit the parameters,

```
--conf spark.yarn.maxAppAttempts=1 s3://BUCKET/relative_score_bcast.py s3://BUCKET/reddit-1 s3://BUCKET/outp
```

and your bucket name is `my-bouquet`, you would write

```
--conf spark.yarn.maxAppAttempts=1 s3://my-bouquet/relative_score_bcast.py s3://my-bouquet/reddit-1 s3://my-
```

The best practice for naming your AWS resources is to adopt a naming convention to document the purpose, type and scope of each resource. AWS (and cloud services) also has a tag feature that offers a programmatic approach but we will not be using it here. See [here](https://github.com/osodevops/aws-enterprise-naming-tagging-standard) (<https://github.com/osodevops/aws-enterprise-naming-tagging-standard>) for a starting point on naming conventions.

For this course, start with the bucket name of `c732-SFU-ID-a5`.

Create an S3 bucket for your course work

To create your bucket, do the following steps:

1. Sign on to your IAM id for AWS.
2. Go to the S3 Console:
 - a. Drop down the menu labelled `Services` (in the top left corner of the AWS landing page).
 - b. Locate the `Storage` category in the menu and click on `S3` within that category.
3. Click `Create bucket` (middle right), producing a dialog box.
4. In the dialog box, enter:
 - a. Bucket name (Text): `BUCKET` (Start with the name `c732-SFU-ID-a5` and add qualifiers as required to uniqueify your name.)
 - b. AWS Region (Dropdown): `US West (Oregon) us-west-2`
 - c. Accept defaults for all other items
 - d. Scroll to the bottom of the page and click `Create bucket`.
5. A status bar will appear top-of-page. If it is green with `Successfully created . . .`, close the dialog by clicking the `X`.

Your EMR Spark jobs will get their source code and input data from your new bucket, as well as send output there.

Upload your code to the S3 bucket

The first step is to upload your two Spark programs from Assignment 4 to your new S3 bucket.

1. Go to the **S3 top-level Console** (<https://s3.console.aws.amazon.com/s3/home?region=us-west-2>).
2. You should see `BUCKET` in the list of Buckets.

3. Click `BUCKET`.
4. Click Upload (upper left) to go to the Upload dialog.
5. Click Add files.
6. Using your operating system's selection tool, locate and select `relative_score_bcast.py` on your computer.
7. Use the same process to select `weather_etl.py` for upload.
8. The S3 Upload dialog should now list the two Python programs in its Files and folders section and `s3://BUCKET` in its Destination section.
9. Click Upload.
10. A status bar will appear top-of-page. If it is green with Upload succeeded, click Close.

Upload the `reddit-1` dataset to the S3 bucket

You will also need to get a copy of the `reddit-1` dataset on your own machine:

1. From the list of files at ggbaker.ca/732-datasets/ (<https://ggbaker.ca/732-datasets/>), select `reddit-1.zip` and download it.
2. Unzip the file, creating a directory named `reddit-1`.

Now upload that directory to S3:

1. From the `BUCKET` bucket, click Upload.
2. Click Add folder.
3. Using your operating system's selection tool, select the `reddit-1` directory.
4. Confirm that you want to upload 10 files.
5. The S3 Upload dialog should now list all 10 partitions of `reddit-1` in its Files and folders section and `s3://BUCKET` in its Destination section.
6. Click Upload.
7. A blue status bar will appear, turning green on success. Click Close.

Create an output folder in the bucket

The last preparatory step is to create a folder for output from your Spark runs:

1. From the `BUCKET` bucket, click Create folder.
2. Enter the name `output`.
3. Click Create folder.
4. Once again you should see a green status bar. Close it.

Verify the bucket contents

Verify that your `BUCKET` bucket contains the following:

1. `relative_score_bcast.py`
2. `weather_etl.py`
3. A `reddit-1` folder with 10 partitions inside it.
4. An `output` folder.

Starting an EMR cluster

Now that you have your code and input data ready, you can run your first Spark job on EMR.

As mentioned in the introduction, the consoles for AWS services tend to be complex and busy. Most of the information they display is irrelevant to your purposes. The key here is to focus on the few items that matter to you and ignoring the rest. This is a useful professional skill for working with AWS specifically and cloud systems more generally.

To start your first cluster:

1. Sign on to your IAM id for AWS.
2. Go to the EMR Console:
 - a. Enter `emr` in the search bar at the top of the page
 - b. Click on the item labelled `EMR`.
3. In the upper right corner, select `Oregon` from the dropdown list of AWS Regions.
4. In the sidebar menu, section `EMR` on `EC2`, click on `Clusters`.
5. Click `Create cluster`.

For this step of the assignment, we're going to create a small, simple cluster, using the `Quick Options` dialog.

1. `General configuration` section:
 - `Cluster name (Text)`: `c732-emr-2x-m5.2x1` (Most other AWS resources including EMR clusters are only scoped within your account and need not be globally unique.)
 - `Logging (Checkbox)`: `Checked`
2. `Software configuration` section:
 - `Release (Dropdown)`: choose the most recent EMR release
 - `Applications (Radio button)`: `Spark`
3. `Cluster configuration` section:
 - `Instance type (Dropdown)`: `m5.2xlarge` (for `Primary` and `Core`)
 - `Delete the Task instance group`.
4. `Cluster scaling and provisioning option` section:
 - `Instance(s) of core nodes`: `2`
5. `Security configuration and EC2 key pair` section:
 - `Amazon EC2 key pair for SSH`: you do not need to specify a key pair. (If you have created a key pair in your AWS account previously, you do not need to use one at this time.)
6. `Identity and Access Management (IAM) roles` section:
 - `Service role`: either create a new service role with default settings or if you have it, select `EMR_DefaultRole`.
 - `Instance profile`: create a new instance profile and select `S3 bucket access All S3 buckets in this account with read and write access`
7. Click `Create cluster`.

It can take 5–15 minutes to create a cluster, depending upon the current AWS load. Near the top of the status screen, you will see a green `Starting` status. Wait until the status changes to `Running`.

Terminating the cluster if you want to take a break

This assignment can take considerable time to complete. If you want to break and resume later, you will want to terminate the cluster so that you do not accumulate charges while you are away.

To terminate the cluster and stop charges, click `Terminate` at the top of the cluster window.

Starting a clone of a terminated cluster

When you want to restart work, instead of creating a new cluster from scratch, you can start a clone:

1. Open the list of clusters by clicking on `Clusters` in the sidebar, under the `EMR` on `EC2` header.
2. In the list of clusters, click the box to the left of the one you would like to clone.

3. Press `Clone`.
4. When asked, "Would you like to include steps?", check `No` and press `Clone`.
5. After a pause, you will see a page summarizing the cluster.
6. The cluster is specified by a sequence of four pages, of which you are currently seeing the last. If you want to modify a parameter of the cluster, you can press `Previous` and `Next` to locate the page on which to modify it. Then press `Next` to return to the fourth page.
7. Once you are satisfied with the cluster specification, press `Create cluster`.

The new cluster will take several minutes to start. When its status becomes `Waiting` you can resume the assignment where you left off.

Section 1: Submitting a first Spark application

Once the status line reads `Waiting` in green, you can submit Spark applications. On EMR, each application is called a "Step". For interactive use, you start up a cluster, submit steps one at a time, then terminate the cluster when you are done. In batch use, you would instead predefine the list of steps and they would begin to run as soon as the cluster was ready, after which it would terminate.

1. Select the `Steps` tab.
2. Click `Add step`.
3. In the `Add step` dialog:
 - Step type (Dropdown): `Spark Application`
 - Name (Text): `Relative average Reddit-1`
 - Deploy mode (Dropdown): `Client`
 - Application location (Text): `s3://BUCKET/relative_score_bcast.py`
 - Spark-submit options (Text): `--conf spark.yarn.maxAppAttempts=1`
 - Arguments (Text): `s3://BUCKET/reddit-1 s3://BUCKET/output/reddit-1-1` **Note:** The text box is narrow and will wrap your arguments at spaces and hyphens. The arguments will nonetheless be correctly formatted when submitted to Spark.
 - Action on failure (Dropdown): `Continue [Default]`
4. Click `Add step` (Lower right)

Wait for change of Status

EMR steps proceed through three Status values: From `Pending` to `Running` and concluding in either `Completed` (in gray) or `Failed` (in red):

Filter: All steps <input type="text" value="Filter steps ..."/> 3 steps (all loaded)							
	ID	Name	Status	Start time (UTC-7) ▼	Elapsed time	Log files	
	s-2US5L1BC73BC8	Relative average Reddit-1	Completed	2021-08-19 09:53 (UTC-7)	28 seconds	View logs	

Add step
Clone step
Cancel step

View Jobs in the Application History Tab

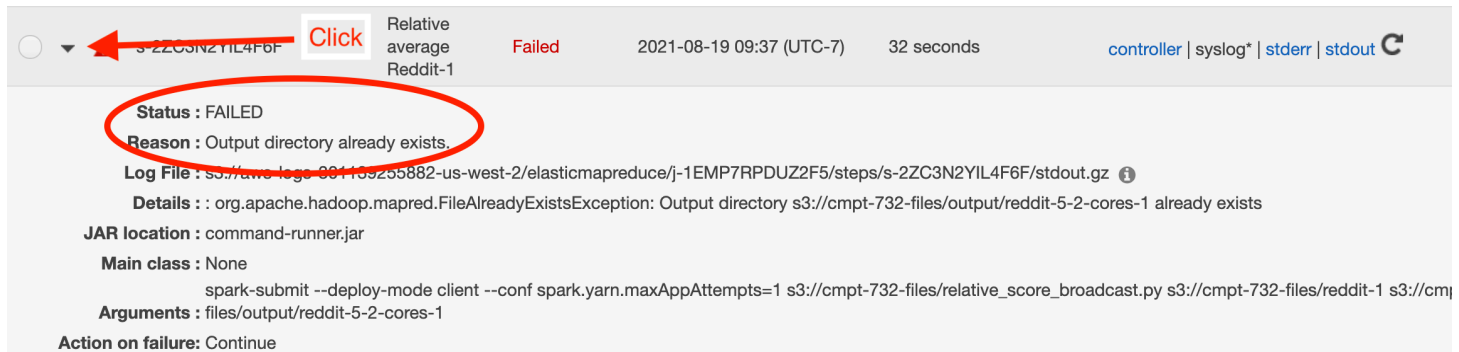
Filter: All steps <input type="text" value="Filter steps ..."/> 5 steps (all loaded)							
	ID	Name	Status	Start time (UTC-7) ▼	Elapsed time	Log files	
	s-1F92OL1CO7N2S	Relative average reddit-5 deploy client	Failed	2021-08-18 10:22 (UTC-7)	26 seconds	No logs created yet	

Your step will start with a `Pending` Status. This display does not automatically refresh, so you will have to press the the "Refresh" icon (clockwise circle) at top of step list, and wait for Status to change to `Running` and eventually either of the final status values.

Diagnosing a Failed Status

If your step failed, it was most likely due to one of two reasons. You can diagnose these case with a few operations on the console.

1. **The output folder already exists.** An annoying feature of Spark is that its jobs can fail in a write stage if you specify an output directory that already exists. Click the leftmost triangle on the step to open up the details:




Relative average
Reddit-1

Failed

2021-08-19 09:37 (UTC-7)

32 seconds

[controller](#) | [syslog*](#) | [stderr](#) | [stdout](#) 

Status : FAILED

Reason : Output directory already exists.

Log File : [s3://aws-logs-301139255882-us-west-2/elasticmapreduce/j-1EMP7RPDUZ2F5/steps/s-2ZC3N2YIL4F6F/stdout.gz](#) ⓘ

Details : org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory s3://cmpt-732-files/output/reddit-5-2-cores-1 already exists

JAR location : command-runner.jar

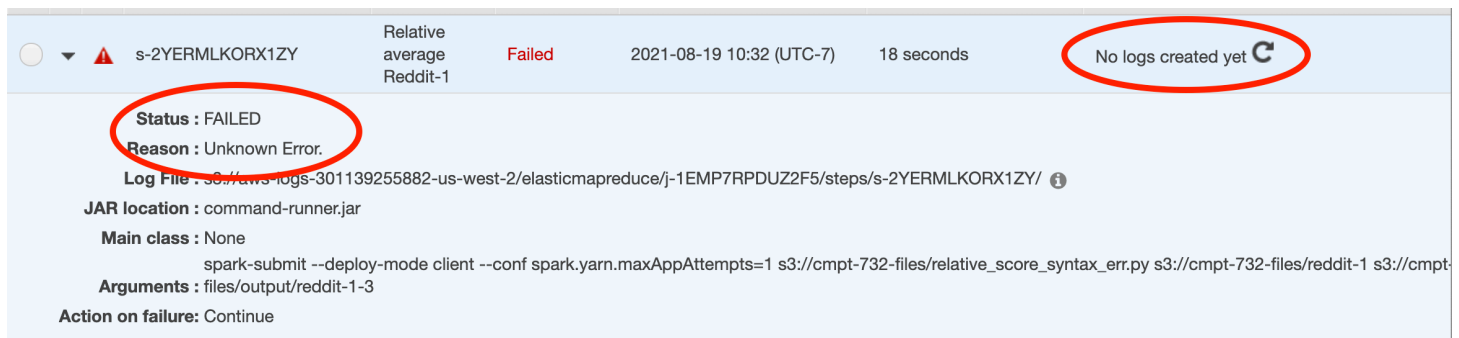
Main class : None

Arguments : files/output/reddit-5-2-cores-1

Action on failure: Continue

In this case, the Reason section will state that the output directory already exists.

2. **There is an error in the Python code.** If you have a Python error, the Step details will have a reason section reading `Unknown error` and you will have to look at the `stdout` log for details. It can take several minutes for that log to become available from the EMR Console. Initially, you will see something like this (note the "No logs created yet" in the upper right):




Relative average
Reddit-1

Failed

2021-08-19 10:32 (UTC-7)

18 seconds

No logs created yet 

Status : FAILED

Reason : Unknown Error.

Log File : [s3://aws-logs-301139255882-us-west-2/elasticmapreduce/j-1EMP7RPDUZ2F5/steps/s-2YERMLKORX1ZY/](#) ⓘ

JAR location : command-runner.jar

Main class : None

Arguments : files/output/reddit-1-3

Action on failure: Continue

(The "No logs created yet" message is misleading. The logs have been created on the cluster but not yet transferred to the management console.) Occasionally press the "refresh" icon (the clockwise arrow) until the logs become available:

After last step completes: Cluster waits

[Add step](#) [Clone step](#) [Cancel step](#)

[View Jobs in the Application History Tab](#)

Filter: All steps Filter steps ...		4 steps (all loaded) 					
	ID	Name	Status	Start time (UTC-7)	Elapsed time	Log files 	
	s-3M44MFM6J98Z4	Relative average reddit-5 deploy client	Failed	2021-08-18 10:13 (UTC-7)	32 seconds	controller syslog* stderr stdout 	

Click `stdout` to see the output, which should indicate the type of error and its line number.

Viewing the results of a success

If the step ended with a "Completed" status, it will have produced output partitions in the S3 directory you specified, `output/reddit-1-1`. In the S3 console, navigate to that directory. Download the `part-00000` partition and verify that it contains the values you expect from Assignment 4.

We won't be using the output files any more, so you can delete them in the S3 console when you're done reviewing them. But leave the `output` directory, which we'll continue to need.

Section 2: Pushing the filter closer to the data

In this section, we will see how pushing some of the computation closer to the data can dramatically reduce the amount of data that we have to process. S3 is a more sophisticated data store than HDFS used on the local cluster and can do important parts of the computation for us before the data is sent to Spark.

A fascinating feature of EMR's adaptation of Spark is that if the query planner is informed that input is coming from S3, the planner will automatically push as much filtering as possible out to S3. The only change you have to make in your code is to specify S3 as the input source.

Loading the input data to S3

This section will read a different dataset, `weather-1`. As you did in the first section, download the dataset to your machine from ggbaker.ca/732-datasets/ (<https://ggbaker.ca/732-datasets/>) and upload it to S3 in your bucket.

Running the original application without S3 filtering

Add a new step with to your cluster with the following parameters:

- › Step type (Dropdown): Spark Application
- › Name (Text): Weather, no S3 filtering
- › Deploy mode (Dropdown): Client
- › Spark-submit options (Text): `--conf spark.yarn.maxAppAttempts=1`
- › Application location (Text): `s3://BUCKET/weather_etl.py`
- › Arguments (Text): `s3://BUCKET/weather-1 s3://BUCKET/output/weather-1`
- › Action on failure (Dropdown): Continue [Default]

The step should run quickly.

Viewing the read size in the Spark history

EMR saves a step's Spark history for a week after it has completed. You can review that history during that week, even after the cluster has been terminated. There is a tremendous amount of detail in a Spark history, which you can use to optimize your applications. We'll dip into those details a few times in this assignment. In this section, we'll just look at the size of the input data.

1. Select the `Summary` tab in the cluster page.
2. Click the `Spark history server` link, located under the `Application user interfaces` header. It may take a minute or two for the history to load into a new browser tab.
3. Look for the application named `weather ETL` (or whatever you named the application in your call to `SparkSession.builder.appName()`). Click on the link in the `App ID` column. Note: In some cases, you may have to go to the `Show incomplete application` page to see your application, even though your application has in fact completed.
4. Click on the `Stages` item in the top menu bar.
5. Click on the lowest "Completed stage", which should have a Stage ID of 0 and be labelled something like `json at NativeMethodAccessorImpl.java:N` (where N is a line number)

Near the top left of the screen, you will see counts of input size, number of input records, output size, and number of output records. Record these values. [?]

Rerunning the application with S3 filtering

Spark on EMR can make use of **S3 Select** (<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark-s3select.html>) to limit the amount of data that must be transferred from S3 into Spark. To see it work, make the following two changes to your copy of `weather_etl.py`:

1. Change the name specified in `SparkSession.builder.appName` to `weather ETL S3 select`.
2. Change the data load call in the main routine to


```
spark.read.format("s3selectCSV").schema(observation_schema).options(compression='gzip').load(inputs).
```

 where `observation_schema` is the variable containing the input schema and `inputs` is the variable containing the input file directory.

Save the revised program as `weather_etl_s3_select.py` and upload it to your bucket in S3.

Run a new step with the same parameters as above, except:

- › Name (Text): `Weather`, with S3 filtering
- › Application location (Text): `s3://BUCKET/weather_etl_s3_select.py`
- › Output directory `s3://BUCKET/output/weather-2`

This step should run in about the same time as the first. However, we will see that it read a much smaller amount of data from S3. Look up this application's history in the Spark history server (recall that it is named `weather ETL S3 select`) and record its input and output counts as you did for the version without S3 filtering. [?]

Terminate this cluster

Terminate this cluster. In the next section, we will use a different configuration.

Section 3: Exploiting the elasticity of cloud services

In a cloud service, you pay for what you use when you use it. You can take advantage of this to expand your capacity temporarily when you need more. This property is called *elasticity*.

In this final section, you will use EMR to temporarily obtain some extra capacity to complete a job more quickly than you could achieve it on the standard cluster used in the course.

For comparison, we're going to run the same job on the local cluster first. Sign on to the local cluster and run the following job:

```
spark-submit --conf spark.dynamicAllocation.enabled=false --num-executors=4 --executor-cores=1 relative_score_broadcast.py
```

Bring up the Spark History at `localhost:18080` (<http://localhost:18080>) (you will need to have an active SSH connection and have met Prerequisite 2 above). Locate your application on the list and click on its App ID link. From the Spark Jobs page, note the `Total Uptime` (top left) and record the durations of each job listed in the table. [?]

Running an on-demand cluster from EMR

Imagine that the run you just made on the local cluster was the largest available to you. For example, the cluster might be shared by multiple teams, with your team allocated no more than this. What can you do if you need results faster than you can get them from your allocated local resources? You scale up by running an on-demand cluster from a cloud vendor such as AWS.

To do this, start a clone of the last cluster (as usual, not copying any steps from it) but this time change the hardware specification (on page "Step 2: Hardware" of the cloning sequence) of Node type `Core`:

- › Amend the name of the clone to `c732-emr-4x-m6gd.x1` (changed this from an earlier version using `m7`, which is not currently available in this region)
- › Set Instance type of the "Core" Core Instance Group to `m6gd.xlarge` (Note the `d`)
- › Set Instance count to 4

We are requesting 4 instances, each with 4 cores, rather than the 4 instances with 1 core that we were allocated on the local cluster.

Once you have made this change, advance to the fourth page ("Step 4") and press `Create cluster`. You will see that these more powerful instances also start up far more quickly than the weaker `m5.2xlarge` instances we used on our earlier EMR clusters.

When the cluster is ready, submit a Spark job running `relative_score_broadcast.py` reading input from `s3://sfu-cmpt-732-redshift/reddit-5/` (this is a publicly-accessible S3 bucket). Remember to specify a brand-new output directory so that your application does not fail on its write.

When the job has completed, open up its Spark history and select the Jobs page. Record the `Total Uptime`, as well as the time of each job when running on this cluster. [?]

Once you have run the job, **terminate the cluster**. These more powerful instances are more expensive and you do not want to run them for longer than necessary.

Questions

Use a word processor of your choice to generate `answers.pdf` and fill in the following:

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.
2. For Section 2:
 - a. What fraction of the input file was prefiltered by S3 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?
3. For Section 3: Look up the hourly costs of the `m6gd.xlarge` instance on the [EC2 On-Demand Pricing \(https://aws.amazon.com/ec2/pricing/on-demand/\)](https://aws.amazon.com/ec2/pricing/on-demand/) 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?

Submission

Submit your files to the CourSys activity [Assignment 5](#).

Updated Thu Oct. 12 2023, 13:04 by sbergner.