Parallel & Distributed Processing

AWS SQS/S3 with Java AWS SDK

The application consists of three scripts. The first script is a Hadoop data preprocessor. The other two are Spark scripts that aggregate the data to find the most viewed channels and the highest upload days

Hadoop Preprocessor

To get it running, launch dfs with

```
start-dfs.sh
```

Ensure that the Hadoop binary path is properly configured, refer to the official docs for its setup. Then compile the source code and compress it into a JAR file via,

```
javac -cp `hadoop classpath` CSVPreprocessor.java
```

```
jar cf CSVPreprocessor.jar  CSVPreprocessor*.class
```

Move the input csv into hdfs with the following. Rreplace the first argument after -put with a path to the input csv.

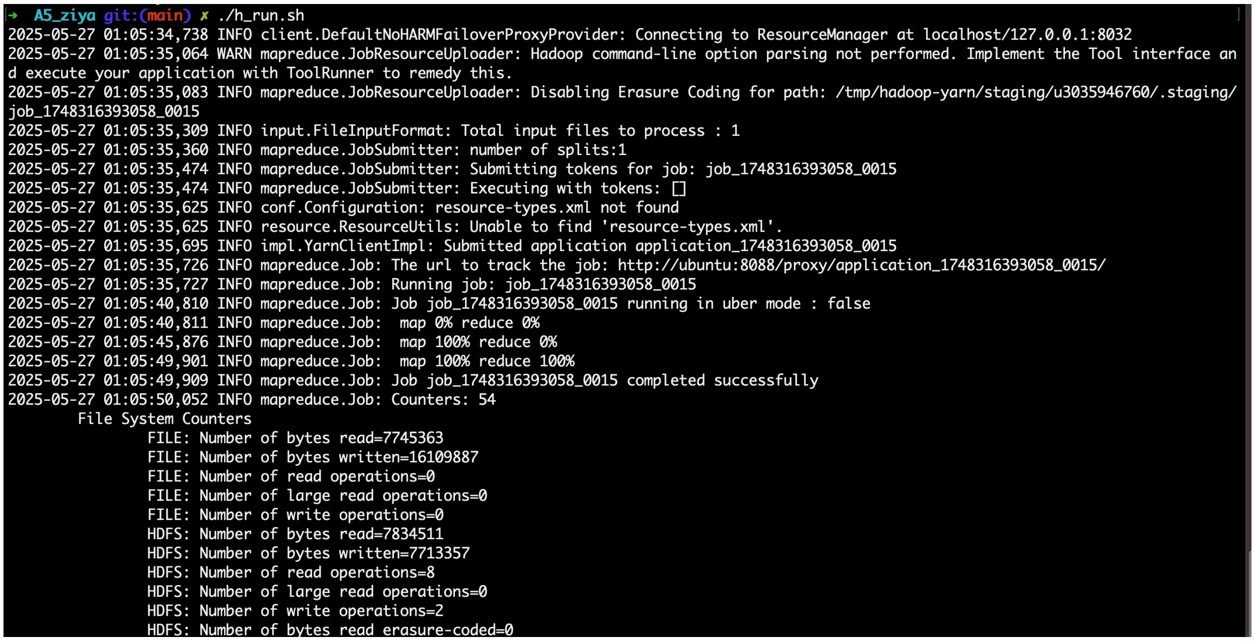
```
hdfs dfs -put /.../input_data.csv /user/your_hadoop_user/input_dir/
```

Run the Hadoop script with (we store the output to std_err in a txt file),

```
hadoop jar CSVPreprocessor.jar CSVPreprocessor /user/your_hadoop_user/
input_dir /user/your_hadoop_user/output_dir 2> hadoop_err.txt
```

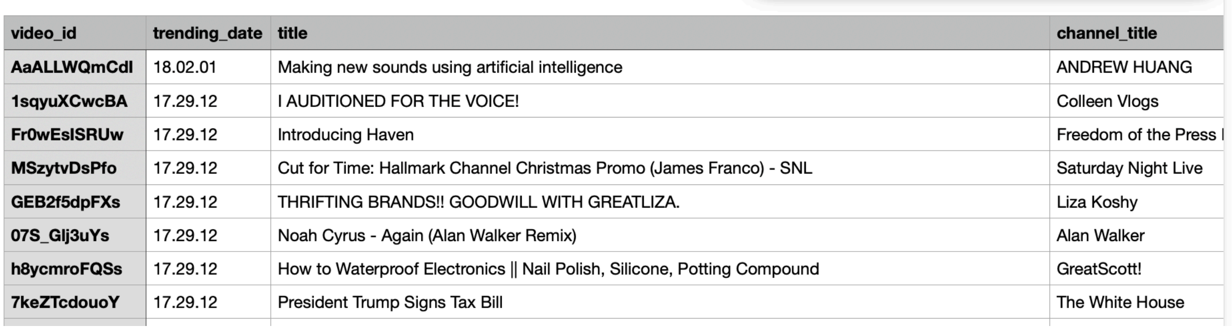
Part of the output has been displayed in Fig. 1 (I put all of the above commands in a single bash script called *h_run.sh*), the entire output has been placed in *hadoop_err.txt* , for some reason it outputs to std_err and not std_out for log messages. Spark outputs to both as we will see later.

The resultant csv from this is displayed in Fig 2.



```
→ A5_ziya git:(main) ✗ ./h_run.sh
2025-05-27 01:05:34,738 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at localhost/127.0.0.1:8032
2025-05-27 01:05:35,064 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface an
d execute your application with ToolRunner to remedy this.
2025-05-27 01:05:35,083 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/u3035946760/.staging/
job_1748316393058_0015
2025-05-27 01:05:35,309 INFO input.FileInputFormat: Total input files to process : 1
2025-05-27 01:05:35,360 INFO mapreduce.JobSubmitter: number of splits:1
2025-05-27 01:05:35,474 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1748316393058_0015
2025-05-27 01:05:35,474 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-05-27 01:05:35,625 INFO conf.Configuration: resource-types.xml not found
2025-05-27 01:05:35,625 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-05-27 01:05:35,695 INFO impl.YarnClientImpl: Submitted application application_1748316393058_0015
2025-05-27 01:05:35,726 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/proxy/application_1748316393058_0015/
2025-05-27 01:05:35,727 INFO mapreduce.Job: Running job: job_1748316393058_0015
2025-05-27 01:05:40,810 INFO mapreduce.Job: Job job_1748316393058_0015 running in uber mode : false
2025-05-27 01:05:40,811 INFO mapreduce.Job:  map 0% reduce 0%
2025-05-27 01:05:45,876 INFO mapreduce.Job:  map 100% reduce 0%
2025-05-27 01:05:49,901 INFO mapreduce.Job:  map 100% reduce 100%
2025-05-27 01:05:49,909 INFO mapreduce.Job: Job job_1748316393058_0015 completed successfully
2025-05-27 01:05:50,052 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=7745363
                FILE: Number of bytes written=16109887
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=7834511
                HDFS: Number of bytes written=7713357
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
```

Fig 1. The Hadoop preprocessor’s partial output (run without redirecting the output to a text file)



video_id	trending_date	title	channel_title
AaALLWQmCdI	18.02.01	Making new sounds using artificial intelligence	ANDREW HUANG
1sqyuXCwcBA	17.29.12	I AUDITIONED FOR THE VOICE!	Colleen Vlogs
Fr0wEsISRUw	17.29.12	Introducing Haven	Freedom of the Press
MSzytvDsPfo	17.29.12	Cut for Time: Hallmark Channel Christmas Promo (James Franco) - SNL	Saturday Night Live
GEB2f5dpFXs	17.29.12	THRIFTING BRANDS!! GOODWILL WITH GREATLIZA.	Liza Koshy
07S_Glj3uYs	17.29.12	Noah Cyrus - Again (Alan Walker Remix)	Alan Walker
h8ycmroFQSs	17.29.12	How to Waterproof Electronics || Nail Polish, Silicone, Potting Compound	GreatScott!
7keZTcdouoY	17.29.12	President Trump Signs Tax Bill	The White House

Fig 2. The csv output from the Hadoop program

Spark Aggregators

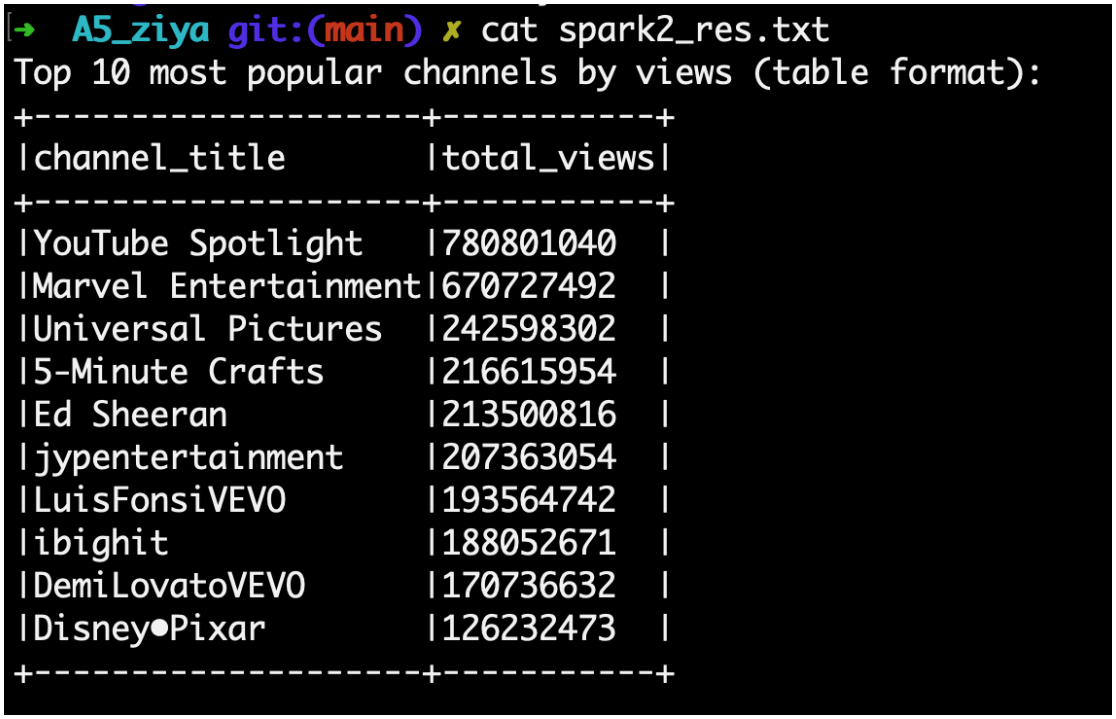
We have two scripts running this time. To get started, we first rename the output csv using hdfs. If you would like to modify these paths, make sure you also adjust them in the scripts.

```
hdfs dfs -mv /user/u3035946760/output/output_dir/part-r-00000
/user/u3035946760/input/USVideos/proc.csv
```

We compile the source files just like earlier (with Javac and jar). Then to run the scripts we use *spark-submit*, like this:

```
spark-submit --class PopularChannels --master local[*]
PopularChannels.jar 2> Spark2_err_2.txt 1> Spark2_res_2.txt
```

The result (output to std_out) is displayed in the figure below. The other information output by spark can be viewed in Spark2_err.txt

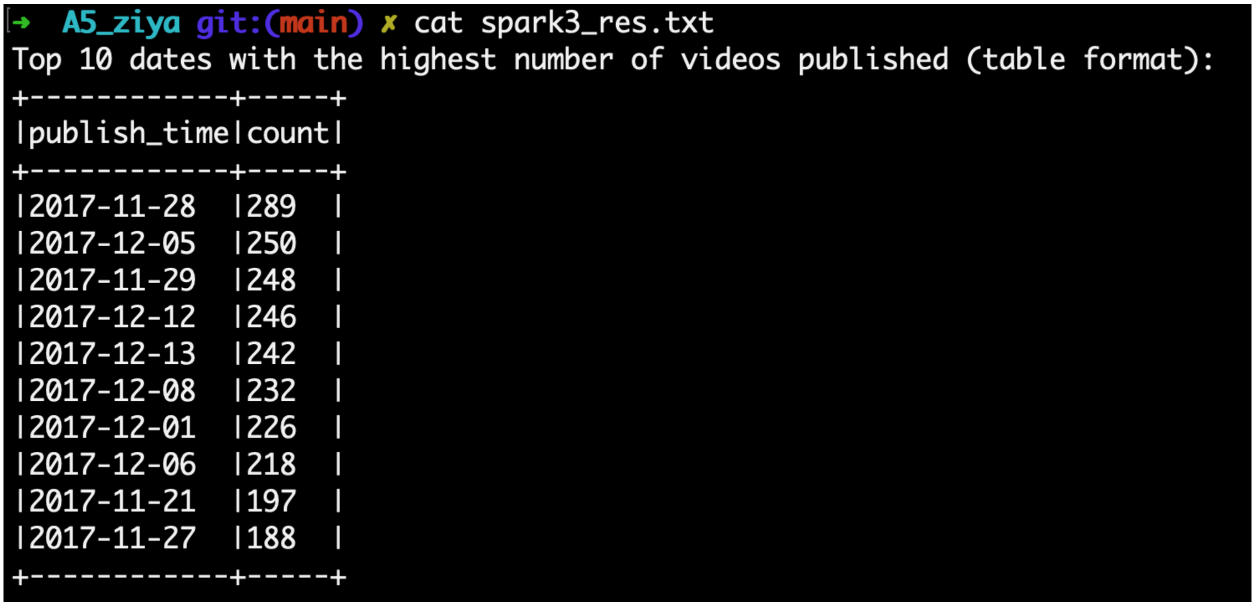


```
➜  A5_ziya git:(main) ✗ cat spark2_res.txt
Top 10 most popular channels by views (table format):
+--------------------+-----------+
|channel_title       |total_views|
+--------------------+-----------+
|YouTube Spotlight   |780801040  |
|Marvel Entertainment|670727492  |
|Universal Pictures  |242598302  |
|5-Minute Crafts     |216615954  |
|Ed Sheeran          |213500816  |
|jypentertainment    |207363054  |
|LuisFonsiVEVO       |193564742  |
|ibighit             |188052671  |
|DemiLovatoVEVO      |170736632  |
|Disney•Pixar        |126232473  |
+--------------------+-----------+
```

Fig 3. The most popular channels output by the Spark script

Likewise, doing the same for the third scripts (with the build process), we get the output shown in figure 4 after running:

```
spark-submit --class TrendingVideoDates --master local[*]
TrendingVideoDates.jar 2> Spark3_err_2.txt 1>
Spark3_res_2.txt
```



```
[→  A5_ziya git:(main) ✗ cat spark3_res.txt
Top 10 dates with the highest number of videos published (table format):
+------------+-----+
|publish_time|count|
+------------+-----+
|2017-11-28  |289  |
|2017-12-05  |250  |
|2017-11-29  |248  |
|2017-12-12  |246  |
|2017-12-13  |242  |
|2017-12-08  |232  |
|2017-12-01  |226  |
|2017-12-06  |218  |
|2017-11-21  |197  |
|2017-11-27  |188  |
+------------+-----+
```

Fig 4. The output for third Spark script