Spark advanced report.

The algorithm that I apply to this problem is the following:

1) Read new data from kafka.
2) Partition it by date, hour, hash tag and user id.
3) Find intersecting partitions – partitions that are already present on the HDFS such that they are also present in the new kafka data – and read them from the HDFS to later be able to merge them with the new data.
4) Remove the partitions that we identified in the previous step from the HDFS.
5) Merge the offloaded HDFS data with the new data and write it back to the HDFS.

Here are the stages:

Completed Stages (6)

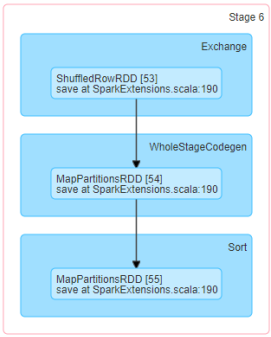
Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
6	save at SparkExtensions.scala:190	+details	2019/01/16 02:19:12	7 s	200/200		18.0 B	747.0 B	
5	save at SparkExtensions.scala:190	+details	2019/01/16 02:19:08	4 s	203/203	9.1 KB			747.0 B
3	count at DataFrameExtensions.scala:46	+details	2019/01/16 02:19:08	87 ms	1/1			177.0 B	
2	count at DataFrameExtensions.scala:46	+details	2019/01/16 02:19:06	2 s	3/3	18.0 B			177.0 B
1	collect at SparkExtensions.scala:173	+details	2019/01/16 02:18:59	6 s	200/200			369.0 B	
0	cache at TopicDataingester.scala:46	+details	2019/01/16 02:18:56	3 s	2/2				369.0 B

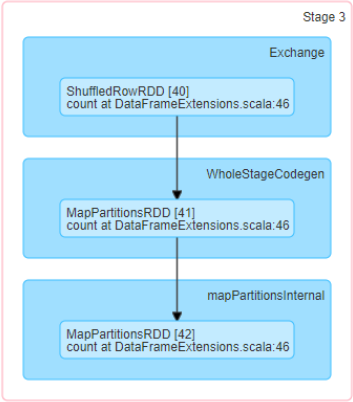
Next I provide screenshots of each stage's DAG.

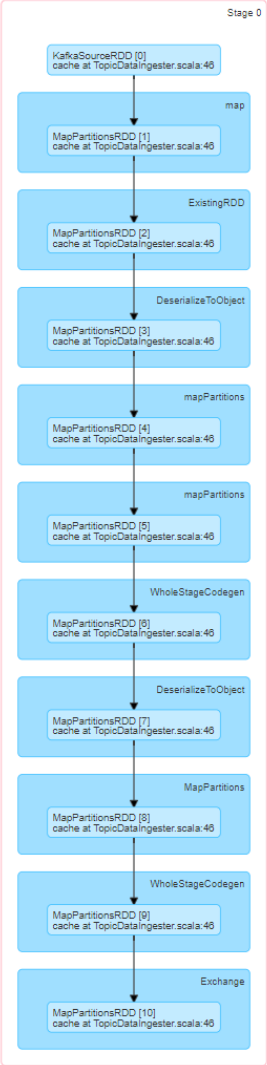
Details for Stage 6 (Attempt 0)

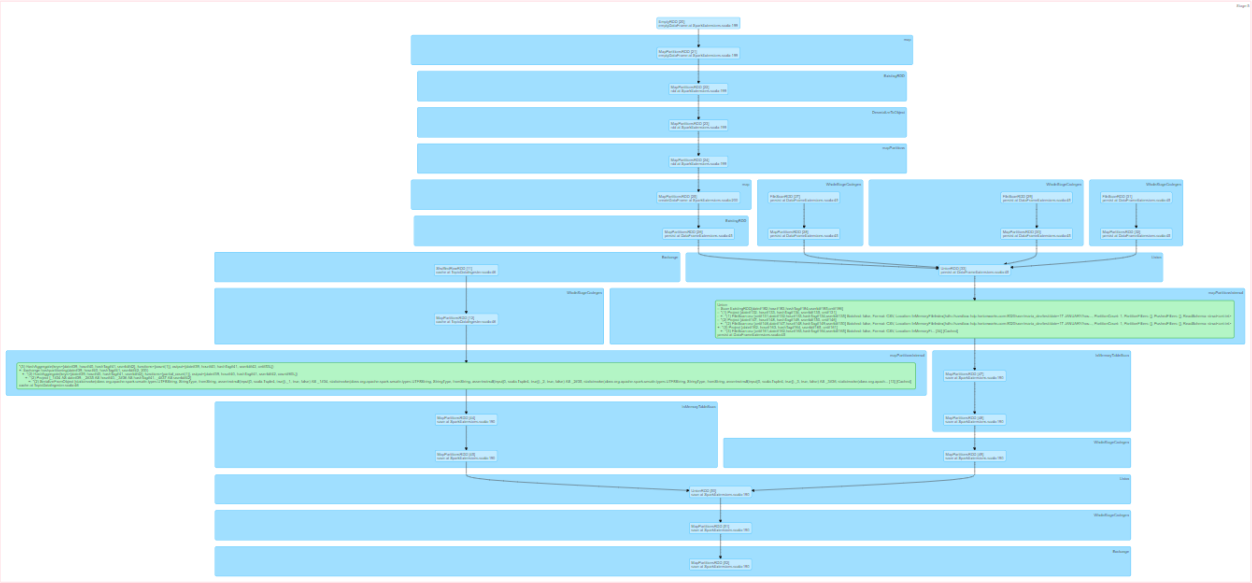
Total Time Across All Tasks: 3 s
Locality Level Summary: Node local: 3; Process local: 197
Output: 18.0 B / 3
Shuffle Read: 747.0 B / 6

DAG Visualization

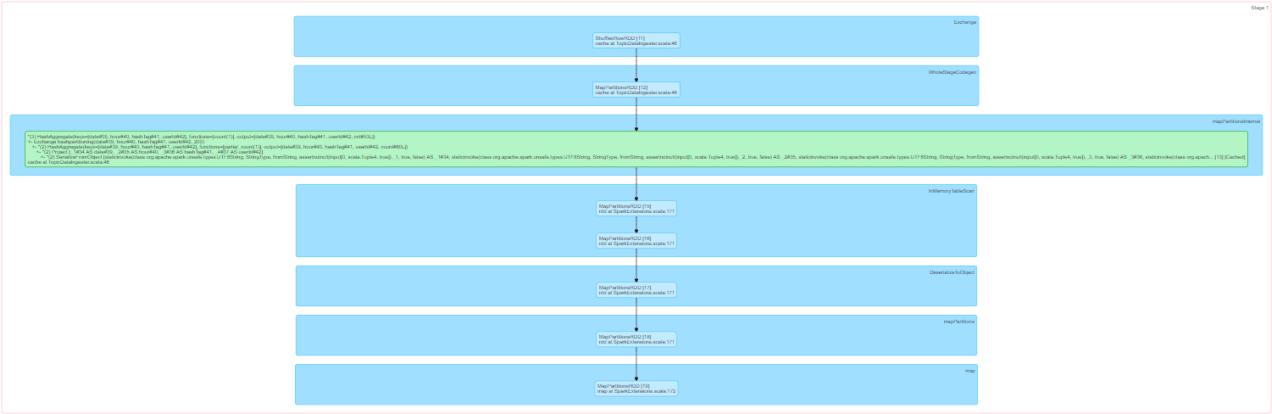


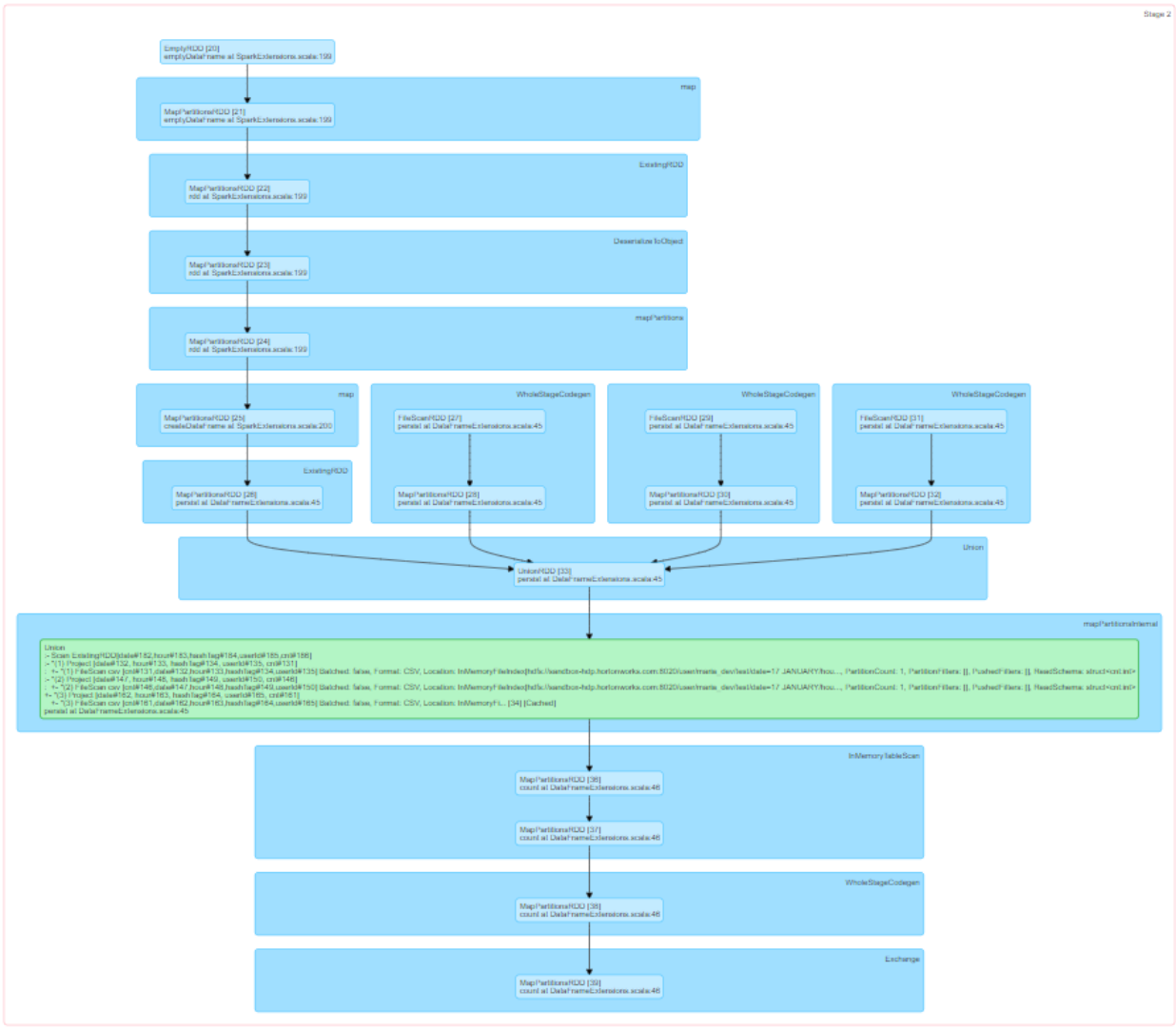

Stage 3
Exchange
ShuffledRowRDD [40]
count at DataFrameExtensions.scala:46
WholeStageCodegen
MapPartitionsRDD [41]
count at DataFrameExtensions.scala:46
mapPartitionsInternal
MapPartitionsRDD [42]
count at DataFrameExtensions.scala:46


Stage 0
KafkaSourceRDD [0]
cache at TopicDataIngester.scala:46
map
MapPartitionsRDD [1]
cache at TopicDataIngester.scala:46
ExistingRDD
MapPartitionsRDD [2]
cache at TopicDataIngester.scala:46
DeserializeToObject
MapPartitionsRDD [3]
cache at TopicDataIngester.scala:46
mapPartitions
MapPartitionsRDD [4]
cache at TopicDataIngester.scala:46
mapPartitions
MapPartitionsRDD [5]
cache at TopicDataIngester.scala:46
WholeStageCodegen
MapPartitionsRDD [6]
cache at TopicDataIngester.scala:46
DeserializeToObject
MapPartitionsRDD [7]
cache at TopicDataIngester.scala:46
MapPartitions
MapPartitionsRDD [8]
cache at TopicDataIngester.scala:46
WholeStageCodegen
MapPartitionsRDD [9]
cache at TopicDataIngester.scala:46
Exchange
MapPartitionsRDD [10]
cache at TopicDataIngester.scala:46



▸ Show Additional Metrics





Show Additional Metrics

After the program is finished running this is what we have in our HDFS data folder:

/ > user > maria_dev > test

Total: 4 files or folders

+ Select All | New Folder | Upload

Search in current directory...

Name	Size	Last Modified	Owner	Group	Permission
_SUCCESS	0.1 kB	2019-01-16 05:19	maria_dev	hdfs	-rw-r--r--
date=16 JANUARY	--	2019-01-16 00:52	maria_dev	hdfs	drwxr-xr-x
date=17 JANUARY	--	2019-01-16 00:55	maria_dev	hdfs	drwxr-xr-x
date=18 JANUARY	--	2019-01-16 00:52	maria_dev	hdfs	drwxr-xr-x

This is what “date=17 JANUARY” folder has:

Name	Size	Last Modified	Owner	Group	Permission
hour=16	--	2019-01-16 00:55	maria_dev	hdfs	drwxr-xr-x

And finally here are some of the result files:

part-00098-65bf641c-2d54-41d8-a8df-1c...	0.1 kB	2019-01-16 15:05	maria_dev	hdfs	-rw-r--r--
part-00129-65bf641c-2d54-41d8-a8df-1c...	0.1 kB	2019-01-16 15:05	maria_dev	hdfs	-rw-r--r--
part-00146-65bf641c-2d54-41d8-a8df-1c...	0.1 kB	2019-01-16 15:05	maria_dev	hdfs	-rw-r--r--

If we open it in excel, we’ll see the hashtag, user id and count data (by which we did not partition):