Example 1: Streaming Word Count on Socket

1. Start Spark Shell

Spark-shell

1. Set log level

scala> sc.setLogLevel("ERROR")

1. Type some words in socket

nc -lk 9998

HELLO WORLD

HELLO

ABC

DEF

ABC DEF

HELLO

HELLO WORLD HELLO

COVID

COVID19

COVID

1. Run streaming code for streaming word count

scala> val lines = spark.readStream.format("socket").option("host", "localhost").option("port", 9998).load()

lines: org.apache.spark.sql.DataFrame = [value: string]

scala> val words = lines.as[String].flatMap(\_.split(" "))

words: org.apache.spark.sql.Dataset[String] = [value: string]

scala> val wordCounts = words.groupBy("value").count()

wordCounts: org.apache.spark.sql.DataFrame = [value: string, count: bigint]

scala> val query = wordCounts.writeStream.outputMode("complete").format("console").start()

1. Run every 60 seconds

scala> val query = wordCounts.writeStream.outputMode("complete").format("console").trigger(Trigger.ProcessingTime("60 seconds")).start()

1. Run only Once

scala> val query = wordCounts.writeStream.outputMode("complete").format("console").trigger(Trigger.Once()).start()

Example 2: Processing streaming Data from HDFS

1. [maria\_dev@sandbox-hdp ~]$ hdfs dfs -copyFromLocal /tmp/activity\_data /
2. Spark Code to be written in spark-shell
3. Load the data

scala> val activity\_data = spark.read.json("/activity-data/")

activity\_data: org.apache.spark.sql.DataFrame = [Arrival\_Time: bigint, Creation\_Time: bigint ... 8 more fields]

1. Visualize the data

scala> activity\_data.show(5)

+-------------+-------------------+--------+-----+------+----+-----+------------+------------+-------------+

| Arrival\_Time| Creation\_Time| Device|Index| Model|User| gt| x| y| z|

+-------------+-------------------+--------+-----+------+----+-----+------------+------------+-------------+

|1424686734964|1424688581018556031|nexus4\_2| 1|nexus4| g|stand|-0.001449585| 0.035491943| 0.027999878|

|1424686735180|1424688581235108765|nexus4\_2| 44|nexus4| g|stand|-0.006790161| -0.05102539| 0.020523071|

|1424686735384|1424688581436677369|nexus4\_2| 84|nexus4| g|stand| 6.866455E-4| 0.01626587|-0.0029754639|

|1424686735588|1424686733589417490|nexus4\_1| 117|nexus4| g|stand|0.0024719238|-0.035812378| -0.023086548|

|1424686735786|1424688581839356812|nexus4\_2| 164|nexus4| g|stand| 0.002822876|0.0013122559| 0.017318726|

+-------------+-------------------+--------+-----+------+----+-----+------------+------------+-------------+

only showing top 5 rows

1. See the Schema

scala> val activity\_schema = activity\_data.schema

activity\_schema: org.apache.spark.sql.types.StructType = StructType(StructField(Arrival\_Time,LongType,true), StructField(Creation\_Time,LongType,true), StructField(Device,StringType,true), StructField(Index,LongType,true), StructField(Model,StringType,true), StructField(User,StringType,true), StructField(gt,StringType,true), StructField(x,DoubleType,true), StructField(y,DoubleType,true), StructField(z,DoubleType,true))

d. Specify the Source

scala> val activity\_data\_stream = spark.readStream.schema(activity\_schema).option("maxFilesPerTrigger",1).json("/sparkLab/activity-data/")

activity\_data\_stream: org.apache.spark.sql.DataFrame = [Arrival\_Time: bigint, Creation\_Time: bigint ... 8 more fields]

e. Specify the transformation and action

scala> val activity\_count\_stream = activity\_data\_stream.groupBy("gt").count()

activity\_count\_stream: org.apache.spark.sql.DataFrame = [gt: string, count: bigint]

f. Specify the Sink and start the streaming

scala> val activityQuery = activity\_count\_stream.writeStream.queryName("activity\_count\_stream\_query").format("console").outputMode("complete").start()

activityQuery: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@71f9c7b

g. Await termination

activityQuery.awaitTermination()

-------------------------------------------

Batch: 0

-------------------------------------------

+----------+-----+

| gt|count|

+----------+-----+

| stairsup|10452|

| sit|12309|

| stand|11384|

| walk|13256|

| bike|10796|

|stairsdown| 9365|

| null|10449|

+----------+-----+

-------------------------------------------

Batch: 1

-------------------------------------------

+----------+-----+

| gt|count|

+----------+-----+

| stairsup|20905|

| sit|24619|

| stand|22769|

| walk|26512|

| bike|21593|

|stairsdown|18729|

| null|20896|

+----------+-----+

-------------------------------------------

Batch: 2

-------------------------------------------

+----------+-----+

| gt|count|

+----------+-----+

| stairsup|31357|

| sit|36929|

| stand|34154|

| walk|39768|

| bike|32390|

|stairsdown|28094|

| null|31343|

+----------+-----+

-------------------------------------------

Batch: 3

-------------------------------------------

+----------+-----+

| gt|count|

+----------+-----+

| stairsup|41809|

| sit|49238|

| stand|45539|

| walk|53024|

| bike|43187|

|stairsdown|37459|

| null|41791|

+----------+-----+

-------------------------------------------

Batch: 4

-------------------------------------------

+----------+-----+

| gt|count|

+----------+-----+

| stairsup|52260|

| sit|61547|

| stand|56924|

| walk|66280|

| bike|53984|

|stairsdown|46825|

| null|52239|

+----------+-----+

-------------------------------------------

Batch: 5

-------------------------------------------

+----------+-----+

| gt|count|

+----------+-----+

| stairsup|62711|

| sit|73856|

| stand|68309|

| walk|79536|

| bike|64781|

|stairsdown|56191|

| null|62687|

+----------+-----+

Example 3: Example 2 with join

1. Make some data to do join with

scala> val prev\_avg = activity\_data.groupBy("gt").avg()

1. See the data

prev\_avg: org.apache.spark.sql.DataFrame = [gt: string, avg(Arrival\_Time): double ... 5 more fields]

scala> prev\_avg.show(5)

C. Join

scala> val activityQuery = activity\_count\_stream.join(prev\_avg, "gt").writeStream.queryName("activity\_count\_stream\_query\_join").format("console").outputMode("complete").start()

activityQuery: org.apache.spark.sql.streaming.StreamingQuery = [org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@1006738a](mailto:org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@1006738a)