ex63

August 20, 2022

[1]:
```python
from pyspark.streaming import StreamingContext
```

[4]:
```python
# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 2)
```

[5]:
```python
inputFileStations = "data/Ex63/data/stations.csv"
```

[6]:
```python
# "Standard" RDD associated with the characteristics of the stations
# Extract (stationId, name)
stationNameRDD = sc.textFile(inputFileStations)\
.map(lambda line: (line.split("\t")[0], line.split("\t")[3]) ).cache()
```

[3]:
```python
# Create a (Receiver) DStream that will connect to localhost:9999
readingsDStream = ssc.socketTextStream("localhost", 9999)
```

[7]:
```python
# Each readings has the format:
# stationId,#free slots,#used slots,timestamp
# Select readings with num. free slots = 0
fullReadingsDStream = readingsDStream.filter(lambda line: int(line.
  ↪split(",")[1])==0)
```

[8]:
```python
# Extract pairs (stationId, timestamp)
stationIdTimestampDStream = fullReadingsDStream.map(lambda line: (line.
  ↪split(",")[0],line.split(",")[3]))
```

[9]:
```python
# Join the content of the DStream with the "standard" RDD to retrieve
# the name of each station.
# To perform this join between streaming and
# non-streaming RDDs the transform transformation must be used
joinDStream = stationIdTimestampDStream.transform(lambda batchRDD: batchRDD.
  ↪join(stationNameRDD))
```

[10]:
```python
# Extract (name of the station, timestamp)
# It is the value part of the returned pairs
stationNameTimestampDStream = joinDStream.map(lambda pair: pair[1])
```

[11]:
```python
stationNameTimestampDStream.pprint()
```

[14]:
```
#Start the computation
ssc.start()
```

[ ]:
```
# Run this application for 90 seconds
ssc.awaitTerminationOrTimeout(90)
ssc.stop(stopSparkContext=False)
```

[ ]:
```
```