CS744A1

Hao Fu, Kan Wu, Huayu Zhang

September 2017

1 Part A

I upgraded Spark to 2.2.0 as it is the stable version for structured streaming. The CPU/mem configuration is in Table 1. To git rid of the annoying logs, I set the log level as WARN.

spark.driver.memory	8g
spark.executor.cores	4
spark.executor.memory	8g
spark.task.cpus	1

Table 1: CPU/mem configuration

Question 1. The key is to count the RT, MT, RE within a 60-minute window. The is done by

```
val windowedCounts = fileStreamDf.groupBy(
window($"timestamp", "60 minutes", "30 minutes"), $"interaction"
).count().orderBy("window")
```

To print the complete table, I set the numRows 563500 (number of files $\times$ maximum number of entries in each file) as an upper bound. The output mode is set "complete".

Question 2. The critical part is to select userB from MT entries.

```
val selectedUser = fileStreamDf.select("userB").where("interaction = 'MT'")
```

To process the data every 10 seconds. I use the Trigger class in the query.

```
val query = selectedUser.writeStream.format("csv")
...
.trigger(Trigger.ProcessingTime("10 seconds"));
```

The output mode is set "append" as I do not need to repeat previous items.

Question 3. I generated the list data as all odd numbers from 1 to 100000. The important thing is to inner join the list data with the stream by "userA".

```
val filteredCounts = fileStreamDf.join(whiteList, "userA").groupBy("userA").count()
```

Part B
In this part, we developed a Storm application to process streaming data from Twitter.

1 Final submitted code:

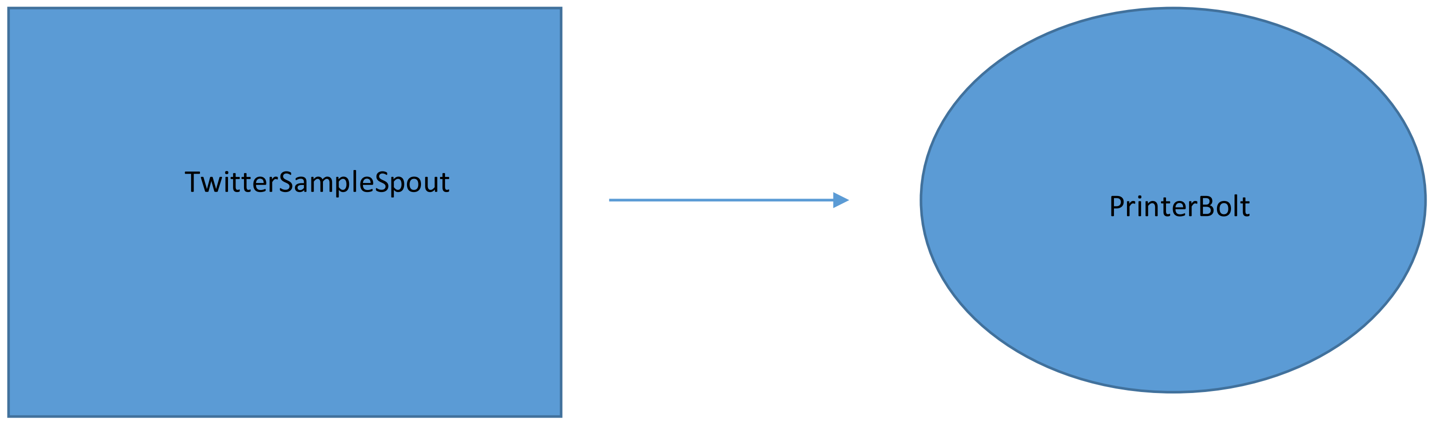
a. Question 1: run PartB/PartBQuestion1.sh to see the result. It will run locally by default. Uncomment the corresponding line then you can run it remotely, however you need to start nimbus and supervisor before remotely running it. The results are stored in local disk /home/ubuntu/q1_tweets.txt in the machine runs it.
Related files: Question1.java, TwitterSampleSpout.java, PrinterBolt.java.

b. Question 2: run PartB/PartBQuestion2.sh to see result. It will run locally by default. Uncomment the corresponding line then you can run it remotely, however you need to start nimbus and supervisor before remotely running it. The results are stored in local disk /home/ubuntu/q2_tweets.txt and /home/ubuntu/q2_ranking.txt in the machine runs it.
Related files: Question2.java, TwitterSampleSpout.java, HashtagGenerator,java, ThresholdGenerator.java, TweetsPrinterBolt.java, RollingCountBolt.java, IntermediateRankingBolt.java, TotalRankingsBolt.java, PrintRankBolt.java.

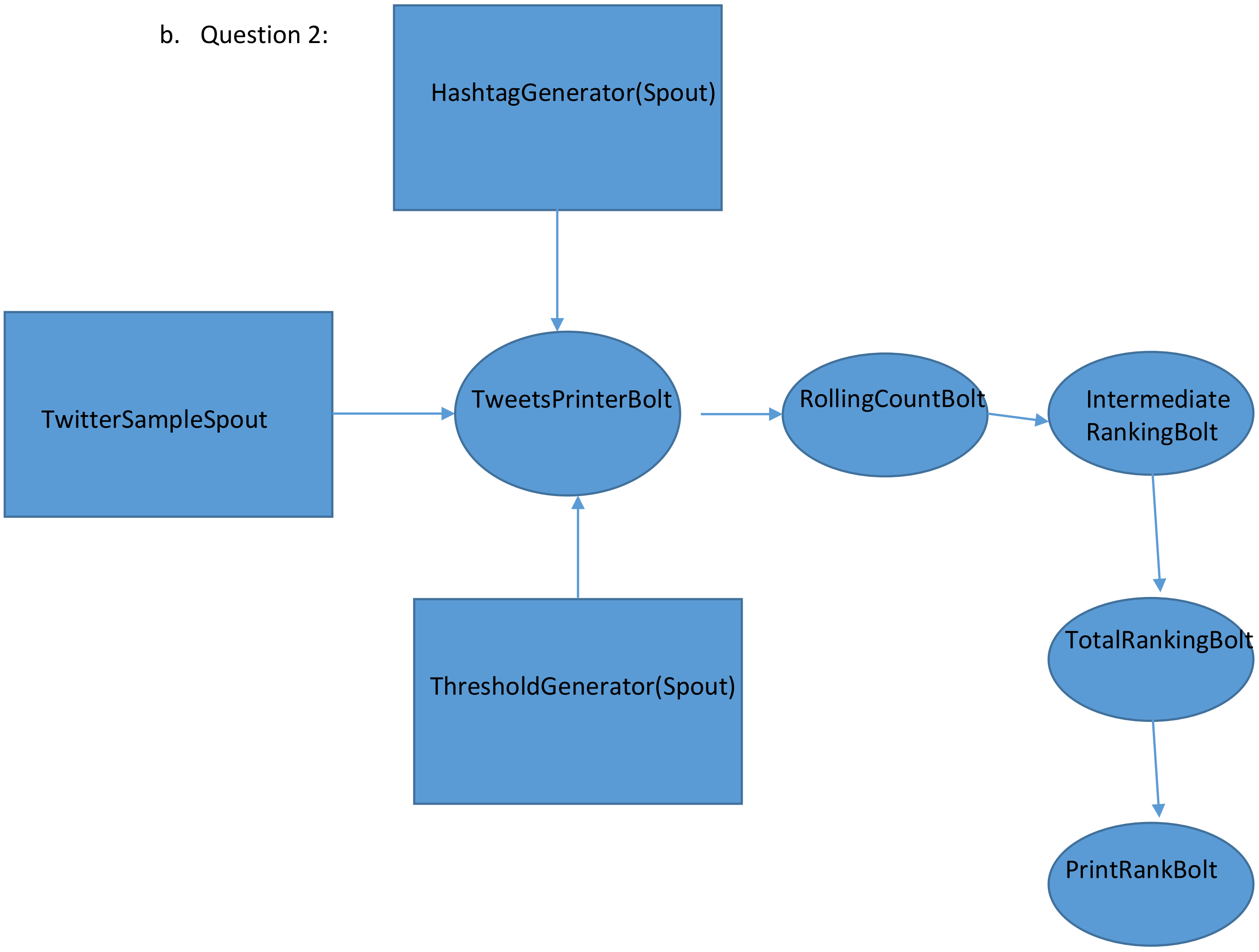
Noted that the popular hashtags now may be not that popular when you grade it.

2 Topology graphs:

a. Question 1:



b. Question 2:


HashtagGenerator(Spout)
TwitterSampleSpout
TweetsPrinterBolt
RollingCountBolt
Intermediate RankingBolt
ThresholdGenerator(Spout)
TotalRankingBolt
PrintRankBolt

Part C

1. **Final Submitted Code**
    a. Question 1:
        i. PartCQuestion-1-disjoint.java
            1. To run: grader_dir/PartCQuestion1-1.sh
        ii. PartCQuestion-1-joint.java
            1. To run: grader_dir/PartCQuestion1-2.sh
    b. Question 2:
        i. PartCQuestion-2.java
            1. To run: grader_dir/PartCQuestion2.sh
            2. This application supports passing in parameter to set allowed lateness
            3. Eg. “flink run -c org.myorg.quickstart.late ~/software/flink-1.3.2/quickstart/target/quickstart-0.1.jar 100”
                a. This will set allowed lateness to be 100 seconds.

2. **Output**
    a. When running each application, the output will be printed to screen, as well as stored in both HDFS:/ and your working directory
    b. To note: before running single application, all previous output from this part will be deleted by default.
    c. But, all sample outputs are also stored in PartC/output/..., eg. C_Q1_disjoint.output

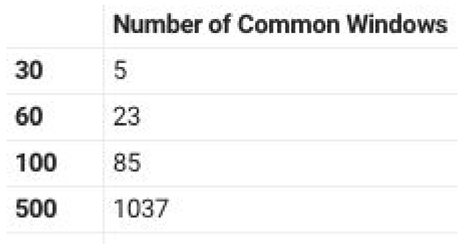
3. Analysis

a. Compare the common windows between allowing lateness with allowed lateness as {30, 60, 100, 500}, and the disjoint results from Question 1

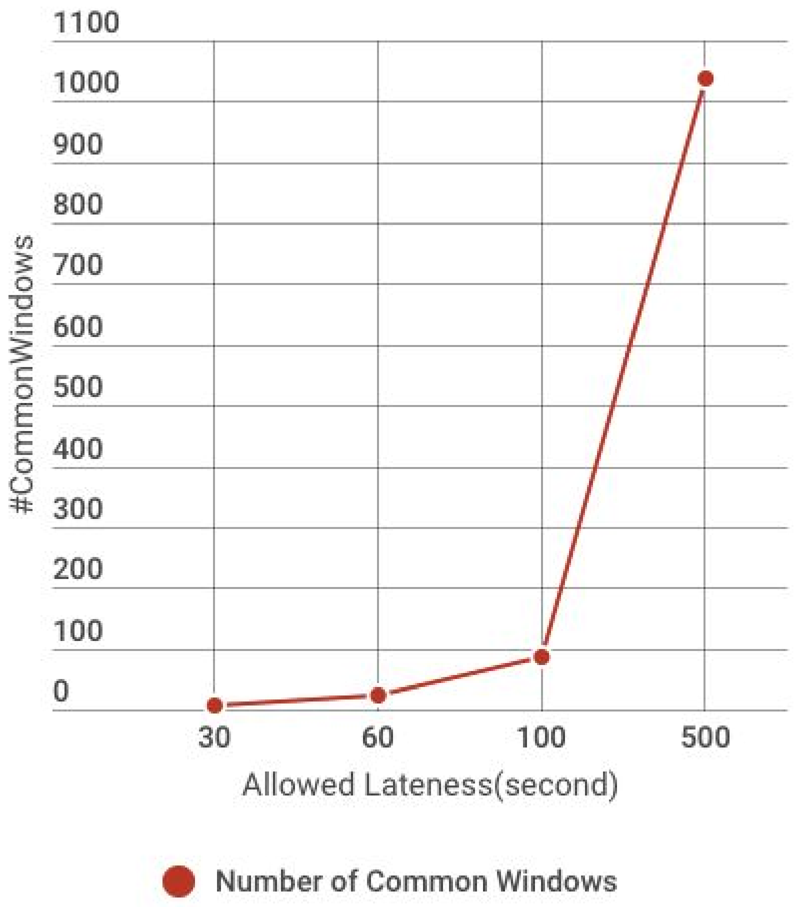
   i. To get common windows, a script is provided: PartC/compare.sh, it mainly dose: “comm -12 <(sort $1) <(sort $2) > $3”

   ii. All common windows are stored in /PartC/output/common_windows/..., eg. common_30

   iii. Number of common windows:



	Number of Common Windows
30	5
60	23
100	85
500	1037


#CommonWindows
Allowed Lateness(second)
● Number of Common Windows