

CS744: Assignment 2

Group - 11

Abhanshu Gupta (agupta89@wisc.edu)

Parikshit Sharma (psharma43@wisc.edu)

Vishnu Lokhande (lokhande@wisc.edu)

PART A – Structured Streaming

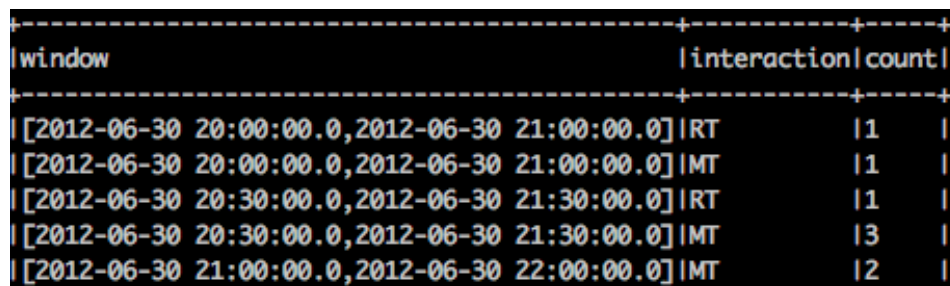
Dataset Description

It is a structured data with each row of the format <userA, userB, timestamp, interaction>. Here interaction can be retweet (RT) or mention (MT) or reply (RE). It is userA who retweets/mentions/replies to userB. The dataset has been divided into 1127 small files and streaming is emulated by periodically moving the files from one directory to the other.

Question 1: Window operations on Structured streaming

In this question, we count the number of retweets, mentions and replies using the event time for an hourly window of 60 minutes, the window getting updated for every 30 minutes. We write the output onto the console and set the output mode to be “complete”. Output mode “complete” prints out all the records, along with the current update on the records.

A few records from the output after accumulating a couple of streams are as follows-



window	interaction	count
[2012-06-30 20:00:00.0,2012-06-30 21:00:00.0]	RT	1
[2012-06-30 20:00:00.0,2012-06-30 21:00:00.0]	MT	1
[2012-06-30 20:30:00.0,2012-06-30 21:30:00.0]	RT	1
[2012-06-30 20:30:00.0,2012-06-30 21:30:00.0]	MT	3
[2012-06-30 21:00:00.0,2012-06-30 22:00:00.0]	MT	2

One thing which can be improved for the API is to allow a functionality to print all the rows on the console. Currently we are required to specify a number to print the specified number of rows, to print all the rows, we specify an arbitrarily large number.

Question 2: Changing the frequency of processing the data

In this question, we store the twitter ids of the users that have been mentioned by other users. This corresponds to the column userB in the data. We filter for those tweets which have interaction as ‘MT’. We set the periodicity of processing the data by adding a trigger and setting

the processing time field in the trigger as 10 secs. The output is written to a hdfs directory /output_partA2/.

Question 3. Mixing static data and streaming computations

In this question, we output the user Id and the number of tweet counts of a user if the user Id is present in a static file. The static file contains a list of user Id's in a comma separated fashion. We read the streaming data as a data frame and filter only those tuples for which the userA's Id is present in the static file. We set the output mode as complete and processing time as 5 seconds.

Sample output is as follows –

```
+-----+-----+
|userA |count|
+-----+-----+
|373941|1    |
|51640 |1    |
|408352|1    |
|38152 |1    |
```

Part C – Flink

Question 1

A total of 1223 disjoint windows were found that contained more than 100 RT, MT or RE. If we instead opted for sliding window with a slide interval of 1s, a total of 73238 windows were found that contained more than 100 RT, MT or RE.

The files can be found at

/home/ubuntu/grader_assign2/partC/data/ass2-q3-a-tumbling.txt

/home/ubuntu/grader_assign2/partC/data/ass2-q3-a-sliding.txt

Question 2

Flink handles late arriving data using Watermarks. A watermark tells the operators about the logical time of the system and can trigger computation on a window.

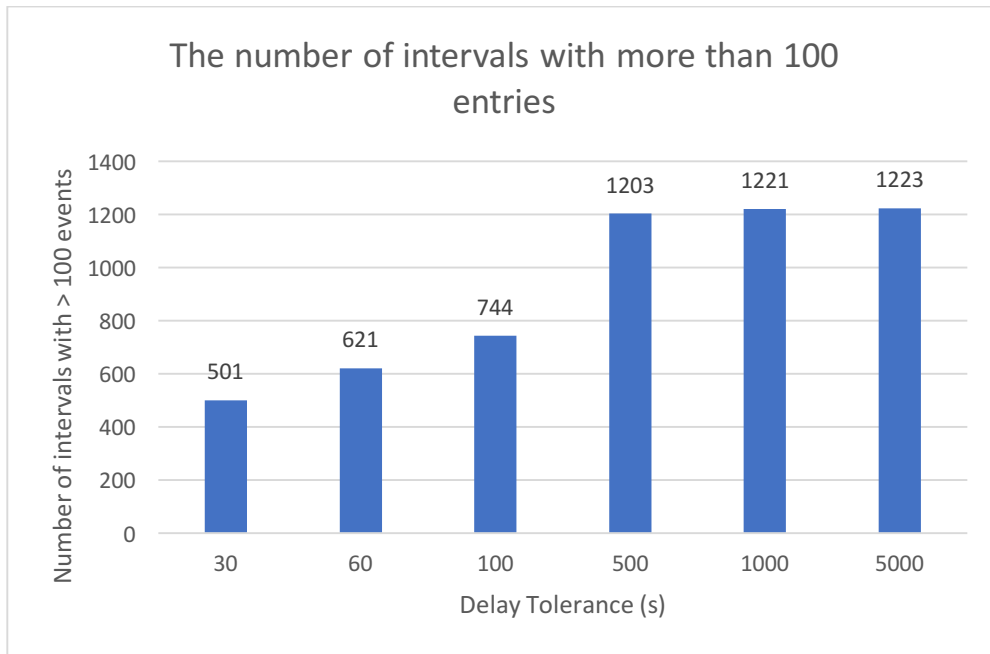
There are multiple strategies for handling late data –

- 1) Delayed Periodic Watermark – Flink provides *'BoundedOutOfOrdernessTimestampExtractor'* that delays watermarks by a specified bound to allow late data to arrive.
- 2) Delayed Punctuated Watermarks – We can also generate watermarks for each arriving entry and account for lateness by delaying the watermark time by a certain amount.
- 3) latenessBound in window – Flink allows windows to accept late entries by a certain amount. This triggers recomputation and duplicates that need to be handled upstream. System watermark is not affected by this.

We used Delayed Periodic Watermark for our results. We found that the results were not deterministic, possibly because of the periodic nature of the watermarks. The numbers also depend on the frequency of watermark generation. For e.g. when using a frequency of 1ms for generating watermarks, there were only 13 windows for late tolerance = 30s which satisfied the > 100 conditions, whereas for the default frequency is 200ms the number of windows was around 500.

The figures on the next page summarize our findings. All numbers are reported for the default frequency of 200ms.

The figure below shows the number of intervals which have more than 100 RT, MT or RE events for different delay tolerances.



The figure below shows the number of intervals that match the intervals generated for the tumbling window with no late arrivals. A delay tolerance of 1000 seconds almost ensures that all late data is accounted for. A delay tolerance of 5000 seconds ensures that no late data is dropped.

