Big Data Analytics

Dr Sirintra Vaiwsri | Email: sirintra.v@itm.kmutnb.ac.th

Event-Time

Event Stream Processing (Antolínez García, 2023)

- Data needs to be processed over a specific interval of time because new sets of data are continuously ingested.
- As a group by operation often needs to see all data before executing the aggregation, windows can be used to illustrate finite data at a specific interval of time.
- Three types of windows:
  - Tumbling
  - Sliding
  - Session

Tumbling Window (Antolínez García, 2023; Macrometa, 2023)

- The tumbling window is a fixed size and non-overlap window.
- This means each element is bound into a single window.
- The main features are disjuncts repeat, and event only belongs to one and only one window.

Sliding Window (Antolínez García, 2023; Macrometa, 2023)

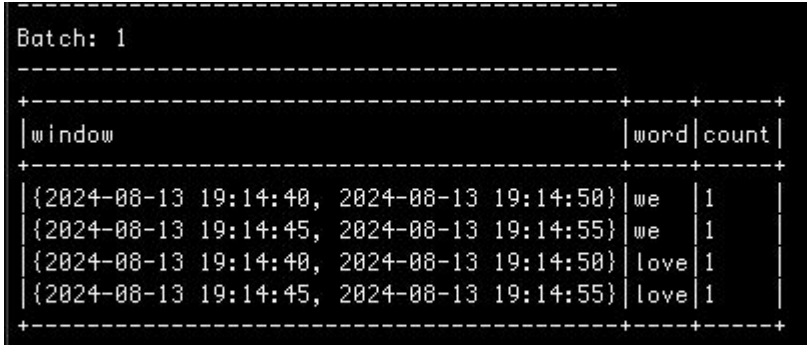
- The sliding window is a fixed size and it overlaps the windows.
- A sliding offset (overlapping dimension) and the interval (window size) are required.
- For example, a window will slide every 5 seconds to create a new window of 10 seconds.

Session Window (Antolínez García, 2023; Macrometa, 2023)

- The session window looks for elements that have occurred continuously.
- It depends on incoming data.
- It starts when input has been received and continues receiving new data if the incoming data comes within a given time interval.
- For example, all received elements within 5 seconds are inserted into the same window. If there is no new data coming for 5 seconds, the current window will be closed.

Example of Implementation

```
from pyspark.sql.functions import window
# Create Socket
# Split lines into words to store in words dataframe
# Add column timestamp into words dataframe
windowCounts = words.groupBy(window(words.timestamp, “10 seconds”,\
       “5 seconds”), words.word).count()
# Write stream to console
```



```
-------------------------------------------
Batch: 1
-------------------------------------------
+------------------------------------------+----+-----+
|window                                    |word|count|
+------------------------------------------+----+-----+
|{2024-08-13 19:14:40, 2024-08-13 19:14:50}|we  |1    |
|{2024-08-13 19:14:45, 2024-08-13 19:14:55}|we  |1    |
|{2024-08-13 19:14:40, 2024-08-13 19:14:50}|love|1    |
|{2024-08-13 19:14:45, 2024-08-13 19:14:55}|love|1    |
+------------------------------------------+----+-----+
```

Watermark

Watermark (Antolínez García, 2023)

- Spark Structured Streaming uses watermark as a cutoff for controlling of how long the Processing will wait for late events.

- A timestamp is required to declare a watermark.

- Example of adding column date and timestamp, and with watermarking for writing stream:

```
words = lines.withColumn(“date”, \
    split(lines[“status_published”], “ ”).getItem(1)).\
    withColumn(“timestamp”, F.current_timestamp()).\
    withWatermark(“timestamp”, “10 seconds”) # F is the imported
                                                # functions as F
```

Assignment (1 point)

- Please implement a code to get the result, as shown in the example shown on slide 8.
- Please execute your code and show the result to get 1 point.

References

- Antolínez García, A. (2023). *Hands-on Guide to Apache Spark 3: Build Scalable Computing Engines for Batch and Stream Data Processing*. Berkeley, CA: Apress.
- Macrometa. Spark Structured Streaming. https://www.macrometa.com/event-stream-processing/spark-structured-streaming. Accessed: 2023-10-16.