

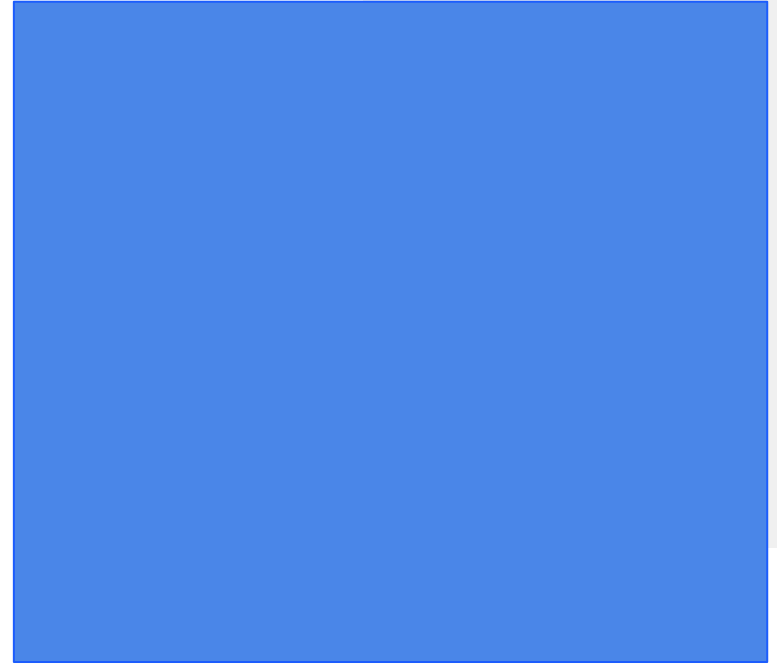


Big Data Analytics

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



Streaming Data Sources




Data Sources (Antolínez García, 2023)

- Streaming Dataframes created using `SparkSession.readStream()`
- Input Sources:
 - Socket source
 - File source
 - Kafka source



Streaming Data from File Source

(Antolínez García, 2023)

- Spark Structured Streaming uses `DataStreamReader` class for streaming text files from a file folder.
 - Spark uses the files in the defined location as a data stream.
 - Thus, the source directory must exist.
 - Also, files in the source directory must be in the same format.
- 

Streaming Data from File Source


(Antolínez García, 2023)

- Spark lists files to identify the new files.
- Spark processes the file as soon as it is discovered.
- The processed file is labelled as processed.
- Spark order processing based on timestamp.
Therefore, the file with the earliest timestamps will be processed first.



Streaming Data from File Source

(Antolínez García, 2023)

- Two main options:
 - **schema** - is the schema of the data
 - **maxFilesPerTrigger** - specifies the maximum number of files read per micro-batch.
 - Control the maximum number of files per trigger.
- 

Streaming Data from File Source

- Example of creating schema fb_part1 and fb_part2:

```
File_schema = StructType([\n    StructField("status_id", StringType(), True),\n    StructField("status_type", StringType(), True),\n    StructField("status_published", StringType(), True),\n    StructField("num_reactions", StringType(), True),\n    StructField("num_comments", StringType(), True),\n    StructField("num_shares", StringType(), True),\n    StructField("num_likes", StringType(), True),\n    StructField("num_loves", StringType(), True),\n    StructField("num_wows", StringType(), True),\n    StructField("num_hahas", StringType(), True),\n    StructField("num_sads", StringType(), True),\n    StructField("num_angrys", StringType(), True)\n])
```



Streaming Data from File Source

(Antolínez García, 2023)

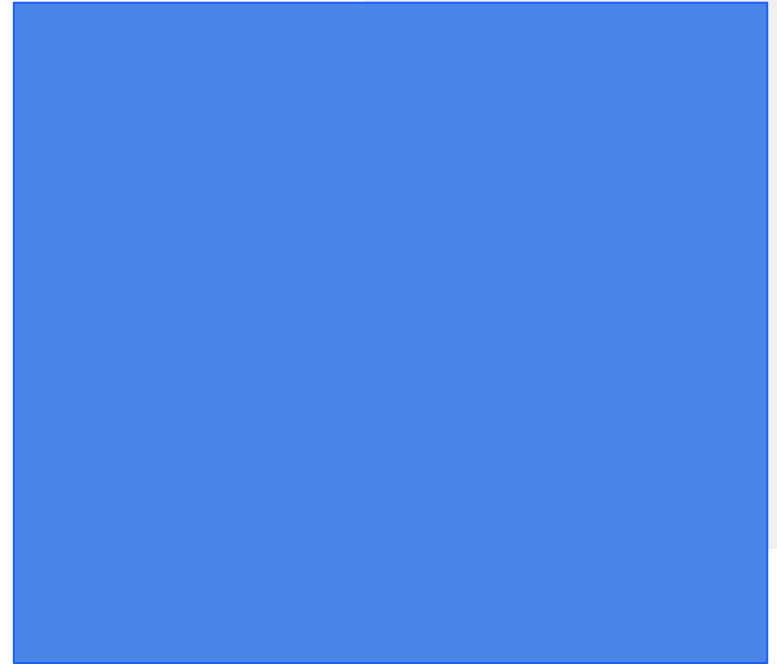
- Example of readStream from fb_part1 and fb_part2:

```
lines = spark \  
    .readStream \  
    .format("csv") \  
    .option("maxFilesPerTrigger", 1) \  
    .option("header", True) \  
    .option("path", "../data/stream") \  
    .schema(file_schema).load()
```

- Example of printing schema:

```
lines.printSchema()
```


Streaming Data Sinks




Data Sinks (Antolínez García, 2023)

- Spark Structured Streaming output sinks are used for saving processed data into an external source.
 - Console sink - used for testing and debugging
 - File sink - stores data in the file system directory.
 - It needs checkpointing streaming.



Spark Checkpointing Streaming

(Antolínez García, 2023)

- Spark uses checkpointing to recover from failures.
 - Checkpointing restores transitional states in the event of failures.
 - Trigger is used to define how often a streaming query will be triggered.
 - One Time - trigger once and stops
 - Processing Time - trigger with user-defined interval
 - Checkpoint Location points to file system directory for storing fault-tolerant in folders such as data checkpointing and metadata checkpointing.
- 

Prepare Data to Write (Antolínez García, 2023)

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

```
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
```

- Example of grouping words:

```
wordCounts = words.groupBy("date", "status_type", \
    "timestamp").count()
```

Write Streaming Data to File Sink

(Antolínez García, 2023)

- Example of writing data:

```
wordCounts.writeStream \  
    .format("csv") \  
    .option("path", "/data/savetofile") \  
    .trigger(processingTime='5 seconds') \  
    .option("checkpointLocation", "../data/savetofile") \  
    .outputMode("append") \  
    .option("truncate", False) \  
    .start().awaitTermination()
```



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.