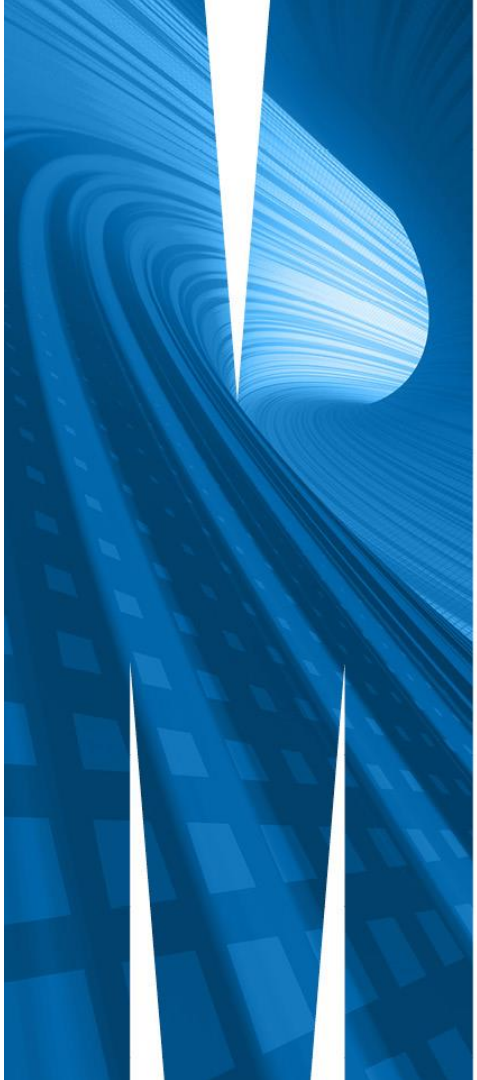
MONASH University

Week 10

FIT5202 Big Data Processing

Data Streaming using Apache Kafka and Spark

Spark Structured Streaming



Week 10 Agenda

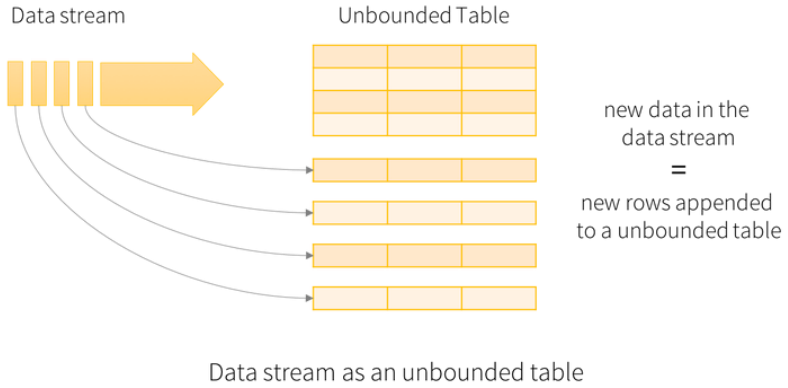
- Week 10 Review
  - Apache Kafka
    - Kafka Producer and Kafka Consumer
- Spark Structured Streaming
  - Introduction
  - Typical Use Case with Kafka
  - DEMO :
    - Word Count (Reading from Socket)
    - Click Stream Analysis (Producer and Consumer)
  - **Use case : Log Analysis**
- Other Topics on Structured Streaming
  - Output Modes
  - Output Sink
  - Triggers

Spark Streaming

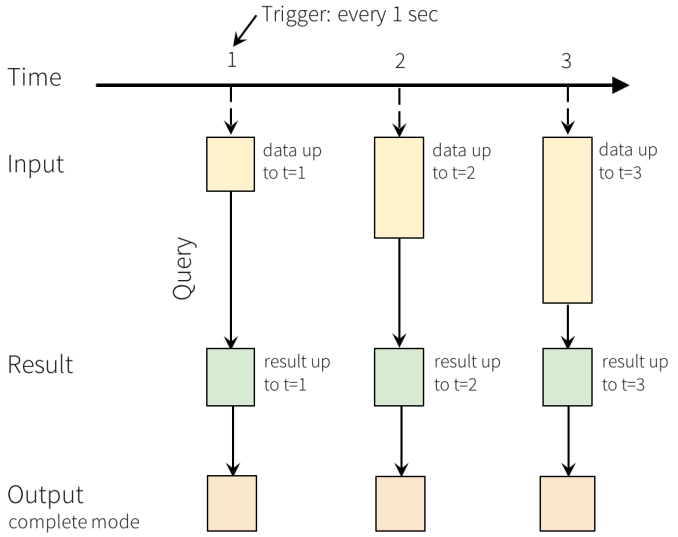
Spark Streaming : used the concept of microbatches, incoming record was a dstream, and RDD based API

Resource : [Comparison between Spark Streaming and Structured Streaming]

Structured Streaming : no concept of batch, data is received in a trigger, appended continuously to the unbounded result table.



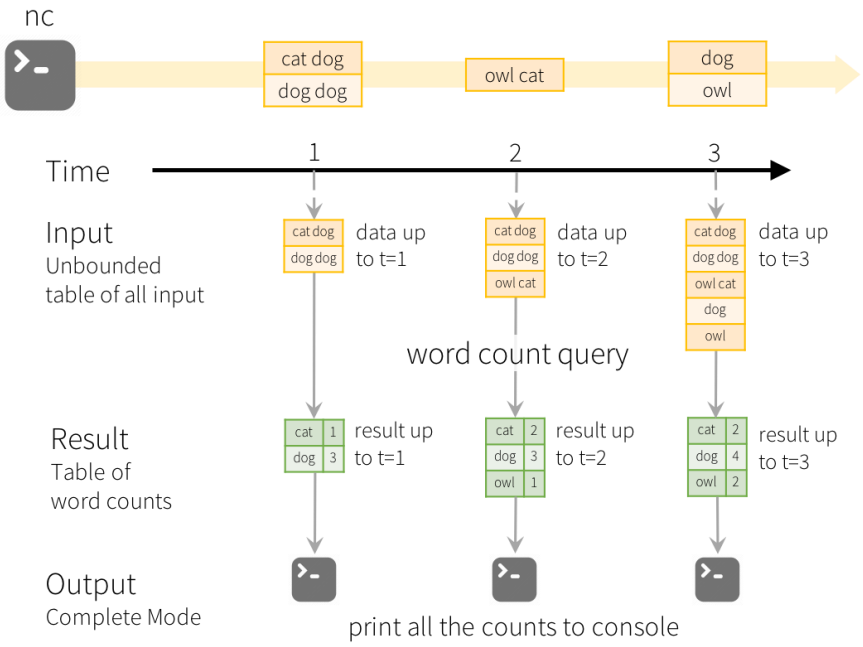
Spark Structured Streaming



Programming Model for Structured Streaming

- A query on the input generates the Result table
- At every **trigger** interval (say, every 1 second), new rows are appended to the input table, which eventually updates the result table
- Whenever the result table is updated, the changed result rows are written to an external **sink**.
- The output is defined as what gets written to external storage, it has 3 modes
    1. Complete Mode
    2. Append Mode
    3. Update Mode

Structured Stream Example (Word Count)



Model of the Quick Example

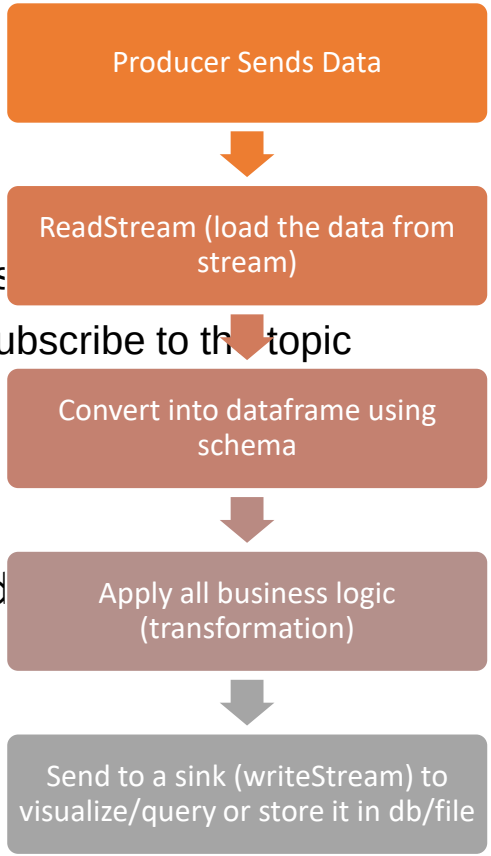
https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

Structured Streaming with Kafka

Clickstream analysis Use Case

Objective : Find and visualize total number of impressions every minute

1. Setup Kafka Producer to stream the clickstream data every 5 seconds
2. Setup Spark session and connection to Kafka Producer/Broker and subscribe to the topic
3. Pull data from Kafka topic with Spark readStream
4. Convert the data into proper structure
    1. Convert from Kafka to String type
    2. Parse it according to our schema (i.e. parse JSON format d
5. Apply the necessary aggregations
6. Create output for Spark Structured Streaming
7. Visualize the realtime data



Parsing Json Format

```
schema = StructType([
    StructField('Clicks', IntegerType(), True),
    StructField('Impressions', IntegerType(), True),
    StructField('ts', TimestampType(), True)
])
```

Parse the string data according to the schema

Cast to String

```
{
  "Clicks":"0",
  "Impressions":"6",
  "ts":1602944650
}
```

```
df = spark
  .readStr
  .format(
  .option(
  .option("subscribe", topic) \
  .load()\
```

key	value	topic	partition	offset	timestamp
[binary]	[binary]	"topic"	0	345	1486087873
[binary]	[binary]	"topic"	3	2890	1486086721

Clicks	Impressions	ts
0	6	1602944650
0	10	1602944650
0	5	1602944650

writeStream

Query/Visualize

Handling Array Json

1. Clickstream Spark Streaming - Handling Json Array DEMO

Output Modes

```
query = df_formatted \
    .writeStream \
    .outputMode("update") \
    .format("console") \
    .start()
```

•**Append mode (default)** - This is the default mode, where only the new rows added to the Result Table since the last trigger will be outputted to the sink. This is supported for only those queries where rows added to the Result Table is never going to change. Hence, this mode guarantees that each row will be output only once (assuming fault-tolerant sink). For example, queries with only select, where, map, flatMap, filter, join, etc. will support Append mode.
•**Complete mode** - The whole Result Table will be outputted to the sink after every trigger. This is supported for aggregation queries.
•**Update mode** - (*Available since Spark 2.1.1*) Only the rows in the Result Table that were updated since the last trigger will be outputted to the sink. More information to be added in future releases.

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#starting-streaming-queries

Output Sink

```
query = df_formatted \
    .writeStream \
    .outputMode("update") \
    .format("console") \
    .start()
```

- **File sink** - Stores the output to a directory.

```
writeStream
    .format("parquet")        // can be "orc", "json", "csv", etc.
    .option("path", "path/to/destination/dir")
    .start()
```

[What is Parquet file format [Ref Link]?]

Efficient as well as performant flat columnar storage format compared to row-based csv or tsv files

- **Console sink (for debugging)** - Prints the output to the console/stdout every time there is a trigger. Both, Append and Complete output modes, are supported. This should be used for debugging purposes on low data volumes as the entire output is collected and stored in the driver's memory after every trigger.

```
writeStream
    .format("console")
    .start()
```

- **Memory sink (for debugging)** - The output is stored in memory as an in-memory table. Both, Append and Complete output modes, are supported. This should be used for debugging purposes on low data volumes as the entire output is collected and stored in the driver's memory. Hence, use it with caution.

```
writeStream
    .format("memory")
    .queryName("tableName")
    .start()
```

- **Foreach sink** - Runs arbitrary computation on the records in the output. See later in the section for more details.

```
writeStream
    .foreach(...)
    .start()
```

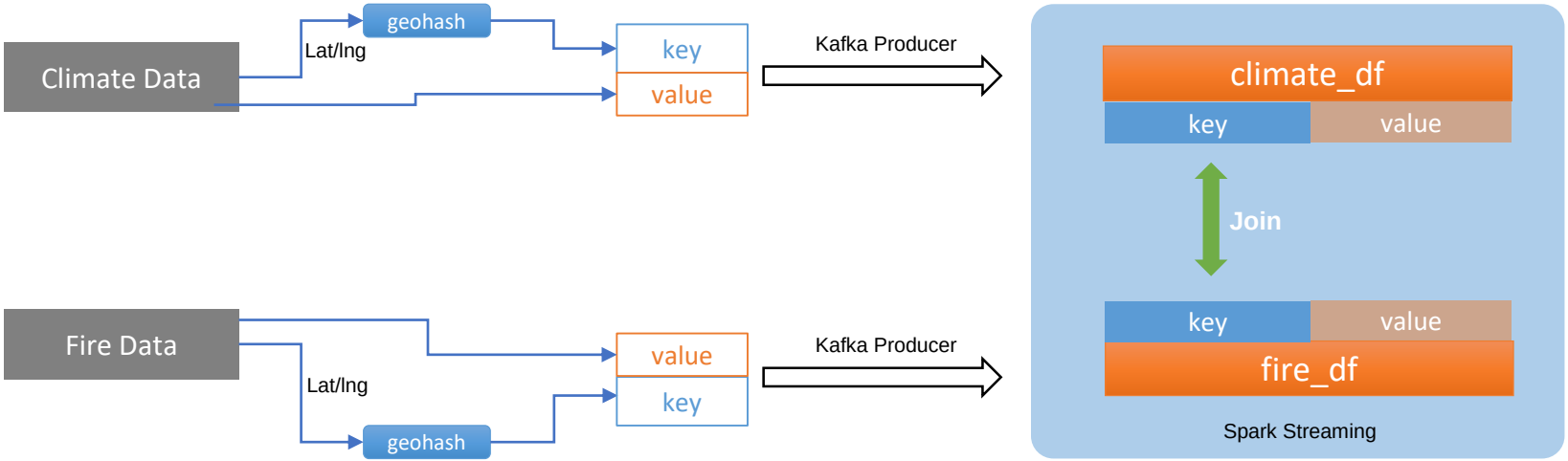
Triggers

```
query = df_formatted \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .trigger(processingTime='5 seconds') \
    .start()
```

1. **Once** : only processes once and terminates the stream
2. **Processing time** : Most widely used and recommended, gives better control over how often micro batch jobs should get triggered.
3. **Continuous** : Experimental Feature, allows to process records in milliseconds latency

https://dzone.com/articles/spark-trigger-options

Joins – Climate/Fire Use Case



Next Lab (Week 11)

- Recovering from Failures with Checkpointing
- Event-time-window based Aggregation
- Handling Late events with Watermarking

References

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

https://docs.microsoft.com/en-us/azure/databricks/getting-started/spark/streaming

Thank You!

See you next week.