Association for
Computing Machinery

Advancing Computing as a Science & Profession

Austin SIGKDD's Spark Talks
Structured Streaming

23 August 2017

Typical sensor/device data flows

Ulind Narang

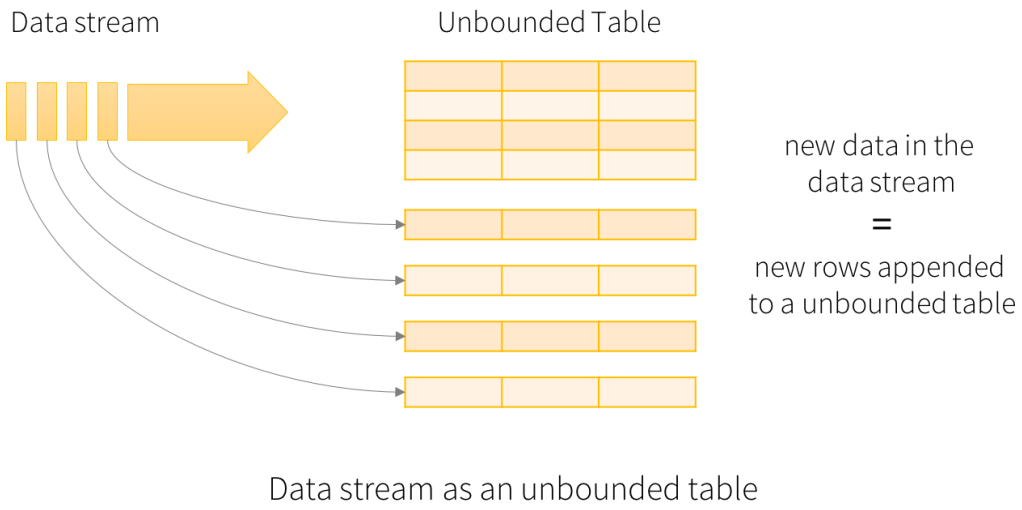
Agenda

- Structured Streaming in Spark
- Programming Model
- Why does Structured Streaming work well?
- Streaming Dataframes and Datasets
- Operations on streams
- Common terms - Watermarking, Checkpointing etc.
- Hands-on Exercise
  - Dataflow for AUS Weather / typical device setup

Structured Streaming in Spark

- Structured Streaming provides fast, scalable, fault-tolerant, end-to-end exactly-once stream processing without the user having to reason about streaming
  - Express streaming computation the same way as batch computation on static data
  - Use Datasets / DataFrame API in Scala, Java, Python or R
  - Executed on Spark SQL engine

Programming Model

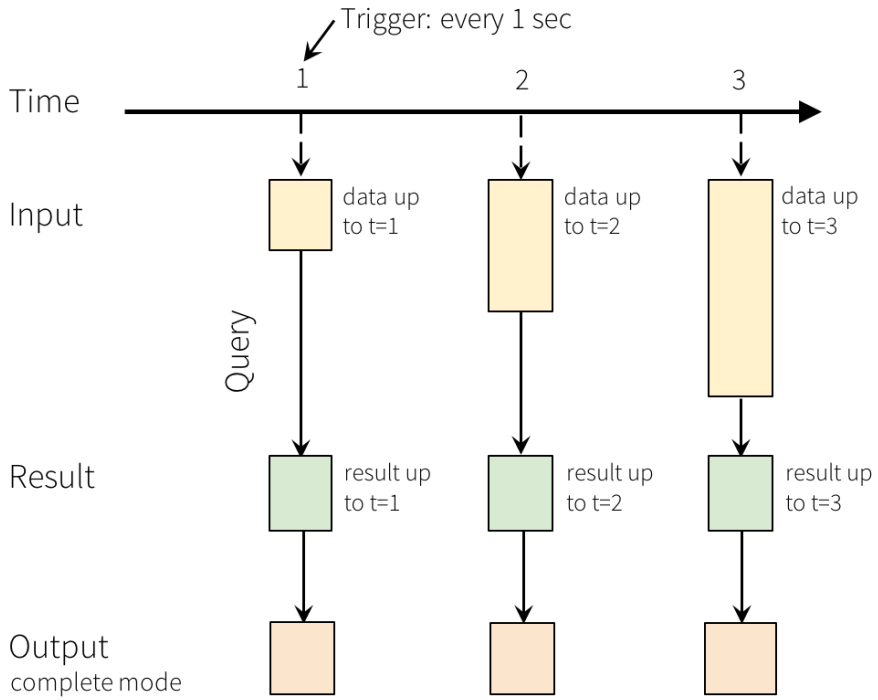


Data stream as an unbounded table

- Every data item that is arriving on the stream is like a new row being appended to the Input Table.

Programming Model cont'd

- A query on the input will generate the "Result Table". Every trigger interval (say, every 1 second), new rows get appended to the Input Table, which eventually updates the Result Table. Whenever the result table gets updated, we would want to write the changed result rows to an external sink.



- Output is in different modes: Complete Mode, Append Mode, Update Mode

Why does structured streaming work well?

- Handling Event-time and Late Data
  - Event-time is embedded in the data and is different from when Spark received data
  - Allows window-based aggregations
  - Similar operation as on a static dataset
  - Handles later than expected data
  - Watermarking for threshold of late data

- Fault Tolerance Semantics
  - Sources, sink and execution engine track exact progress of processing. Failure is handled by restarting or reprocessing
  - Check pointing and write ahead logs to record offset range
  - Replayable sources and idempotent sinks enable e2e exactly once semantics

Streaming Dataframes and Datasets

- Static bounded data
- Streaming unbounded data

- Input Sources
  - Files as a source
  - Kafka source
  - Socket source

- Data schema
  - Ad-hoc use cases can have inferred schema
  - File based sources use specified schema

Operations on Streams

- Basic Operations – selection, projection, aggregation
- Windowing operations on event time
- Handling Late data and water marking
  - Define watermark of a query by specifying the event time column and the threshold on how late the data is expected to be in terms of event time.

```
val windowedCounts = words
  .withWatermark("timestamp", "10 minutes")
  .groupBy( window($"timestamp", "10 minutes", "5 minutes"), $"word")
  .count()
```

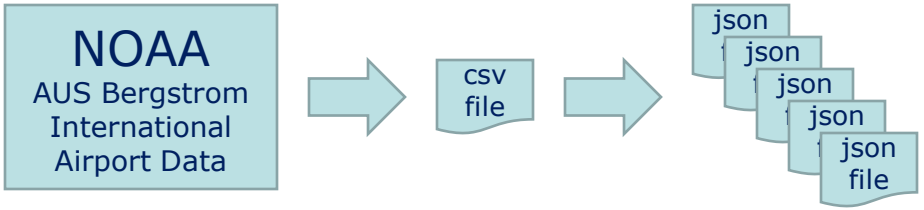
- Join operations

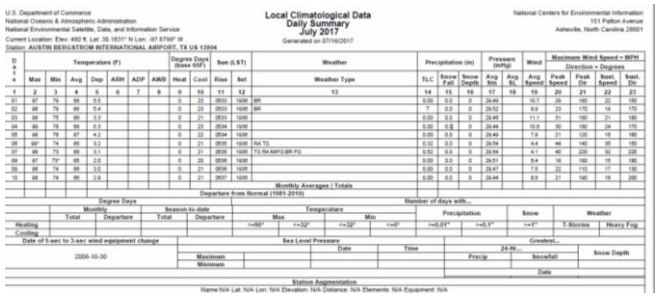
Common Terms

- Event-time
- Late Data
- Watermarking
- Checkpointing
- Exactly-once fault-tolerance guarantee
- Write Ahead logs

- Other concepts to learn about streaming queries
  - Managing queries
  - Monitoring queries

Hands-on Exercise: Dataflow for AUS Weather

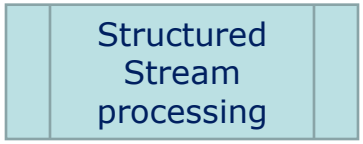




Convert each hourly data entry (row) to json
Python

Ingest data as batch

Ingest data as stream



Various queries
Scala

S3 bucket stores csv file and json files

Streaming and non-streaming operations – how similar

```
val df_staticWeatherInput =
  spark
    .read
    .schema(jsonWeatherSchema)
    .json(inputFilesPath)
```

```
val df_StreamWeatherInput =
  spark
   .readStream
   .schema(jsonWeatherSchema)
   .option("maxFilesPerTrigger", 1)
   .json(inputFilesPath)
```

References

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