ADALTAS

Data Analysis with Apache Spark

3. Spark Structured Streaming

What is Streaming Data

ADALTAS

- Continuously generated data
- Coming from many data sources simultaneously
- Small in size (kB range)
- Examples: IoT, stock market, recommendation based on geo-location, electricity consumption...

Difference between Batch and Stream Processing

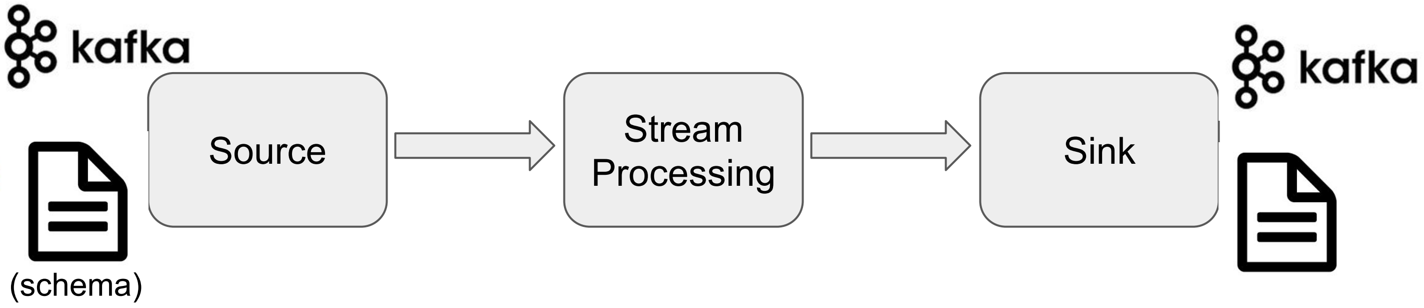
	Batch processing	Stream processing
Data scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a rolling time window, or on just the most recent data record.
Data size	Large batches of data.	Individual records or micro batches consisting of a few records.
Performance	Latencies in minutes to hours.	Requires latency in the order of seconds or milliseconds.
Analyses	Complex analytics.	Simple response functions, aggregates, and rolling metrics.

Streaming

- Unbounded data sets (in opposition to finite data sets);
- Unbounded data processing (in time);
- Spark Structured Streaming: fault-tolerant exactly-once processing

Streaming data sources and sinks

- Apache Kafka
- Amazon Kinesis
- Load files from S3 using Auto Loader
- Optimized Amazon S3 Source with Amazon SQS
- Azure Event Hubs
- Delta Lake tables
- Read and write streaming Avro data
- Write to arbitrary data sinks



Stream processing models

- **Traditional** vs **micro-batch model**

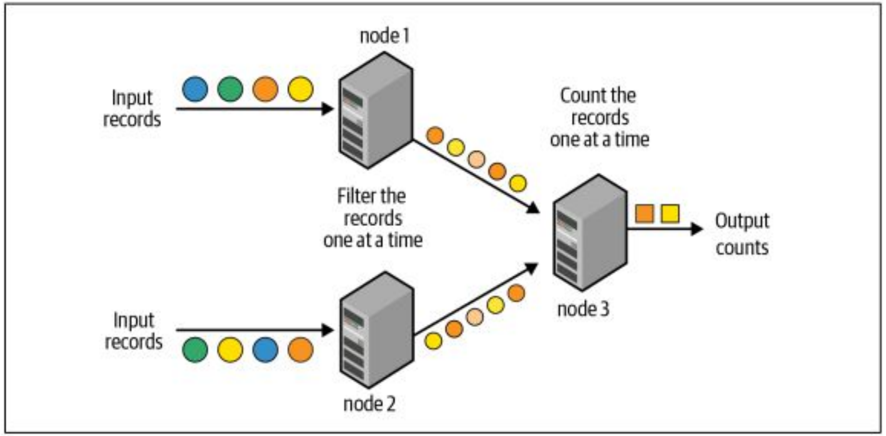


Figure 8-1. Traditional record-at-a-time processing model

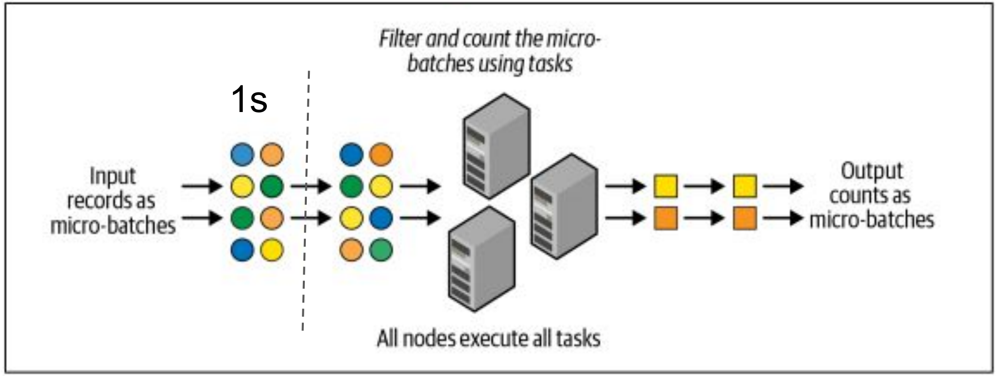


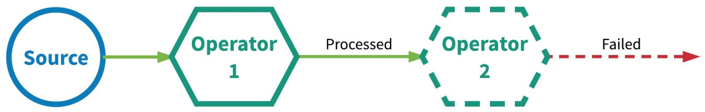
Figure 8-2. Structured Streaming uses a micro-batch processing model

- redundancy
+ latency (ms)

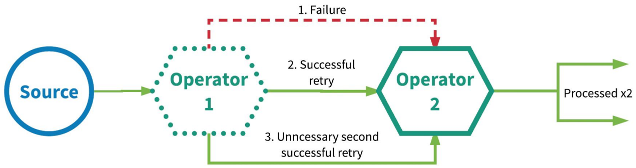
+ redundancy
- latency (100 ms: exactly-once)

Different processing semantics

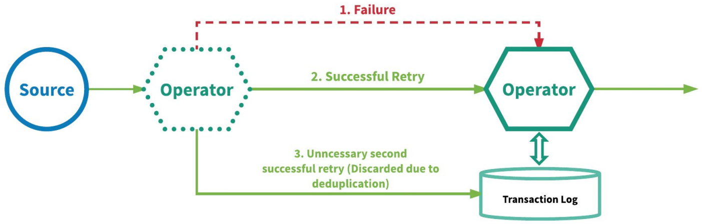
- At-most-once



- At-least-once

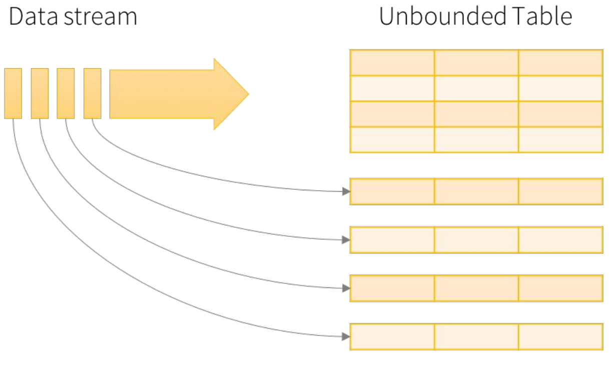


- Exactly-once

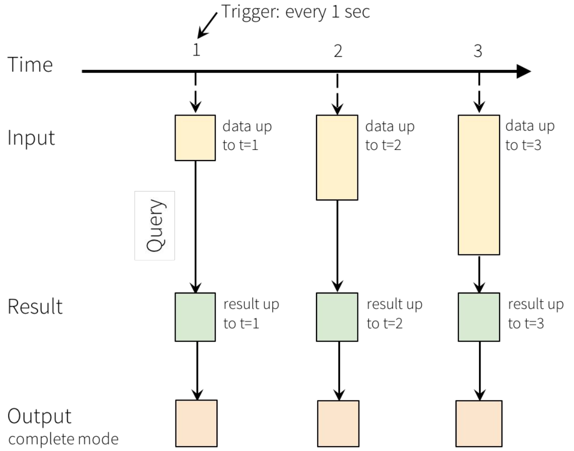


Programming model

- Streaming computation as standard batch-like query



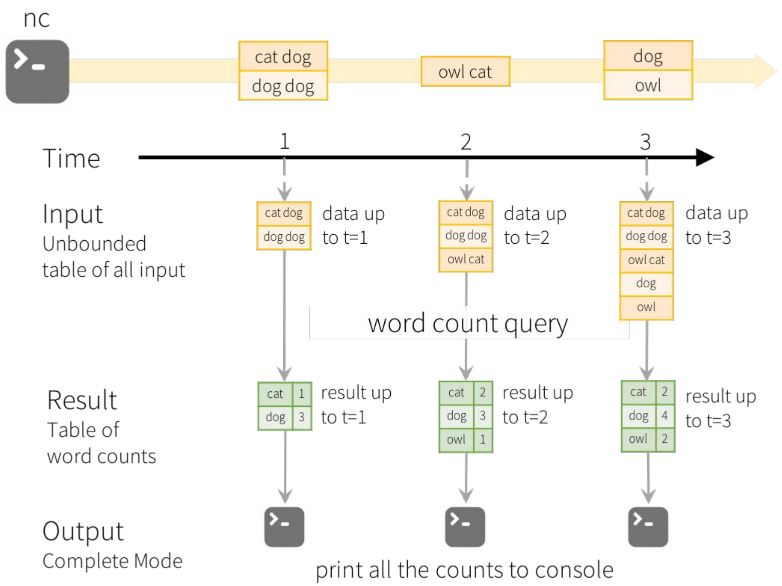
Data stream as an unbounded table



Programming Model for Structured Streaming

Programming model: example

- Word count

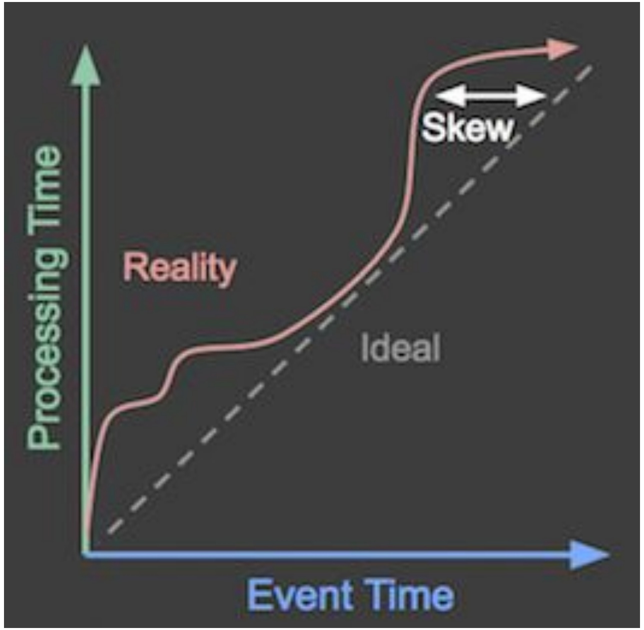


Output modes:

- complete - writes all the rows of a Result Table
- append - writes "new" rows only
- update - writes only the rows that were updated

Event-time vs Processing time

- event time - when the data was generated
- processing time - when Spark received the data



Operations on streaming DataFrames

- The syntax is very similar than what we saw with static DFs

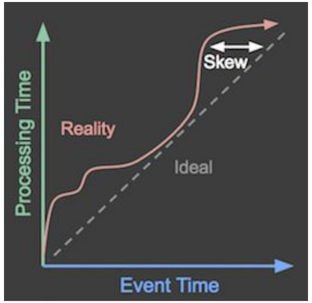
```
df = ...  # streaming DataFrame with IOT device data with schema { device: string, deviceType: string, signal: d
ouble, time: DateType }

# Select the devices which have signal more than 10
df.select("device").where("signal > 10")

# Running count of the number of updates for each device type
df.groupBy("deviceType").count()
```

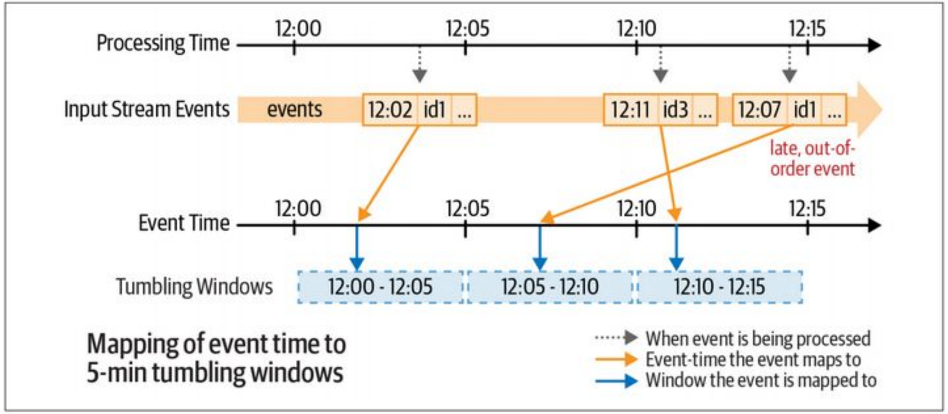
- SQL-like and RDD-like operations

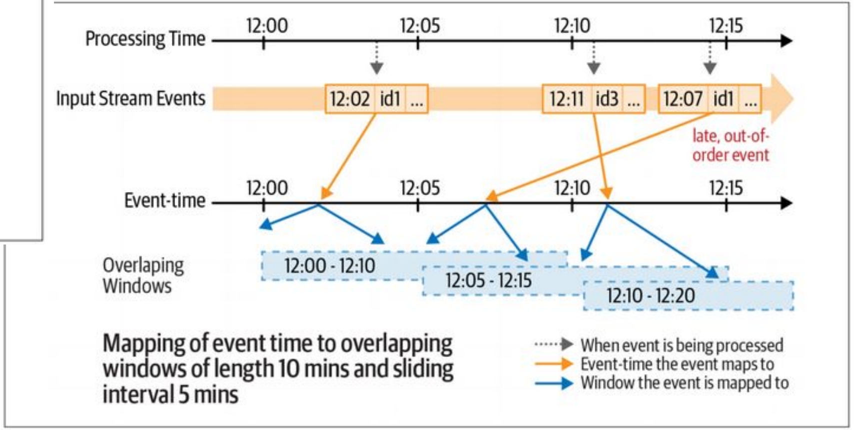
```
df.createOrReplaceTempView("updates")
spark.sql("select count(*) from updates")  # returns another streaming DF
```

- several operations are not supported: distinct, take(n), limit, chained aggregations, some types of joins...

Windows: tumbling, overlapping

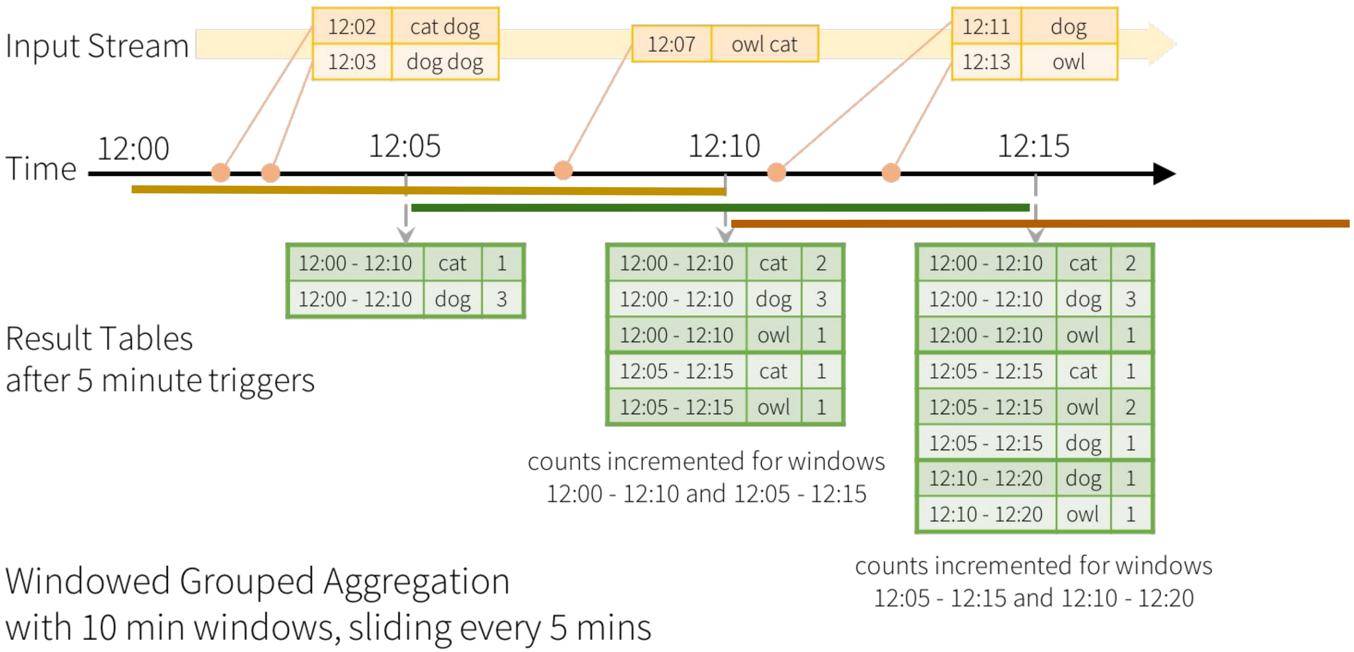






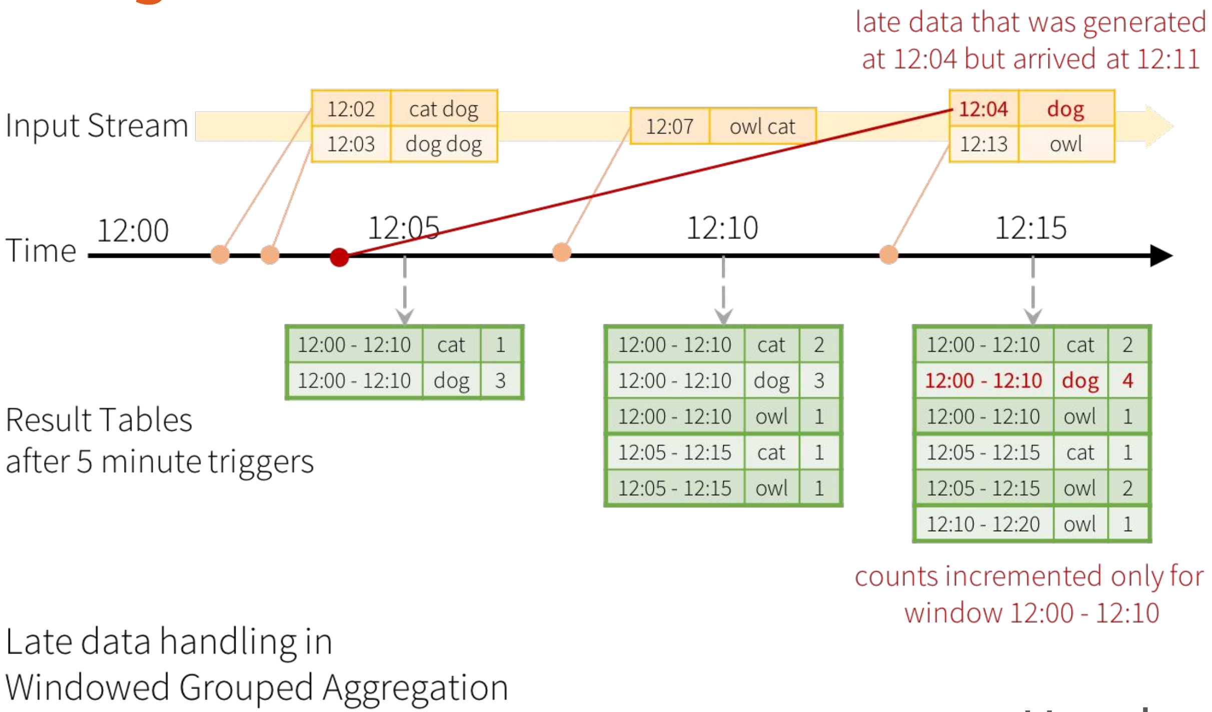
- Each event belongs to one or more windows

Aggregations over a sliding event-time window



Handling late data

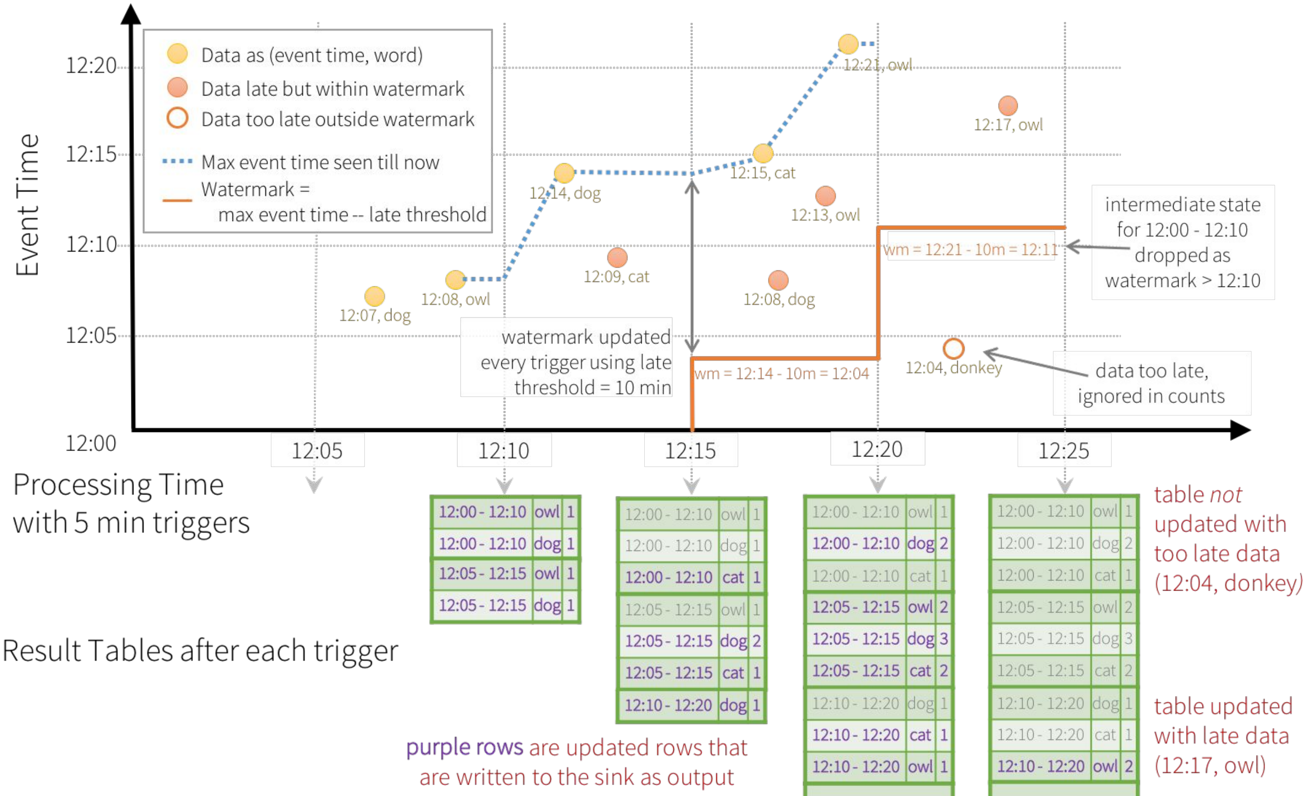
ADALTAS



How long will we wait?

Watermarking

ADALTAS



Watermarking in **Windowed Grouped Aggregation** with **Update Mode**

Vocabulary

- **Event-time** - a time when the event happens
- **Processing time** - a time when we receive the data of the event
- **Trigger** - how often we will collect the data
- **Window** - for how long will we collect the data
- **Watermark** - when the late data is discarded (how late is too late)