Handling Streaming Data with Spark Structured Streaming

Mohit Batra
Founder, Crystal Talks

linkedin.com/in/mohitbatra

Overview

Understand how streaming works & options in Spark

How Spark Structured Streaming works?

Extract streaming data from source

Transform and load data to sink

Understanding Streaming in Spark

Data Pipelines

Batch Pipeline

Streaming Pipeline

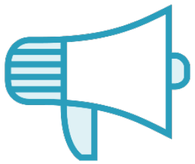
Ecommerce

What kind of solutions can we build with batch and streaming pipelines?

Batch Pipeline

How much sales have happened this week across product categories?

What is the growth in revenue – MoM / YoY?



What is the impact of multiple promotions?

Works with finite datasets

Involves lot of historical data

Processes data periodically

Streaming Pipeline

Works with infinite datasets

Involves real-time data

Processes data continuously

Provide recommendations to users

Monitor the application logs for system failures

Track the deliveries

Streaming Applications

Near Real-Time

Speed is important
but don't need immediate output

10 sec - 10 min

Movie recommendations

Social media tracking

Application monitoring

Weather updates etc.

Real-Time

Information needs to be processed immediately

100 ms - 10 sec

Fraud detection

Self-driving cars

Online gaming

Network monitoring etc.

Complexities in Stream Processing

Separate batch & stream pipelines

Diverse data sources & formats

Dirty / late arriving data

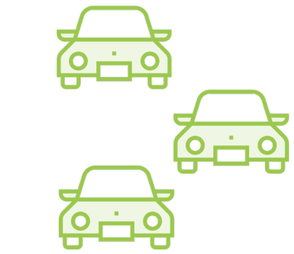
Run interactive queries on streaming data

Apply machine learning

Fault tolerance

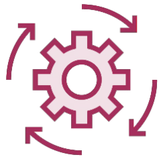
Apache Spark allows to build unified batch & streaming data pipelines!

Streaming Scenario



App / Cab Device

Stream Ingestion
(Azure Event Hubs)



Stream Processing
(Structured Streaming)

Data Lake

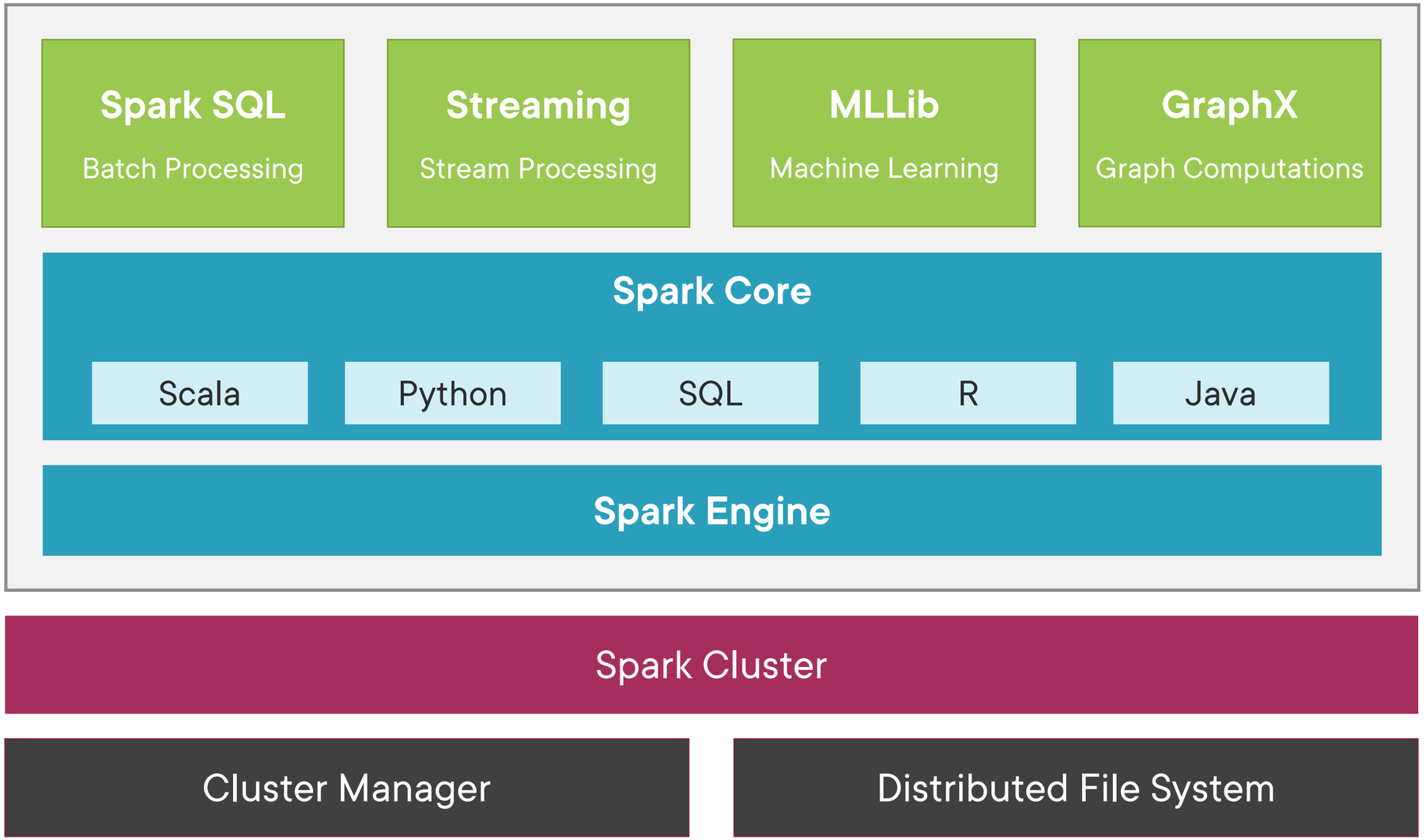
Database

Reporting

Static Files

Historical Data
(Azure Data Lake)

Spark Architecture



Streaming in Spark

Spark Streaming

First implementation (part of Streaming lib)

Works on RDDs

Separate batch and streaming APIs

Spark Structured Streaming

Introduced with Spark 2.x (part of Spark SQL)

Works on DataFrames

Unified batch and streaming APIs

Modes

- Fixed-Interval (100 ms latency)
- Continuous (1 ms latency)

Performance improvements in Spark 3+

Structured Streaming Processing Model

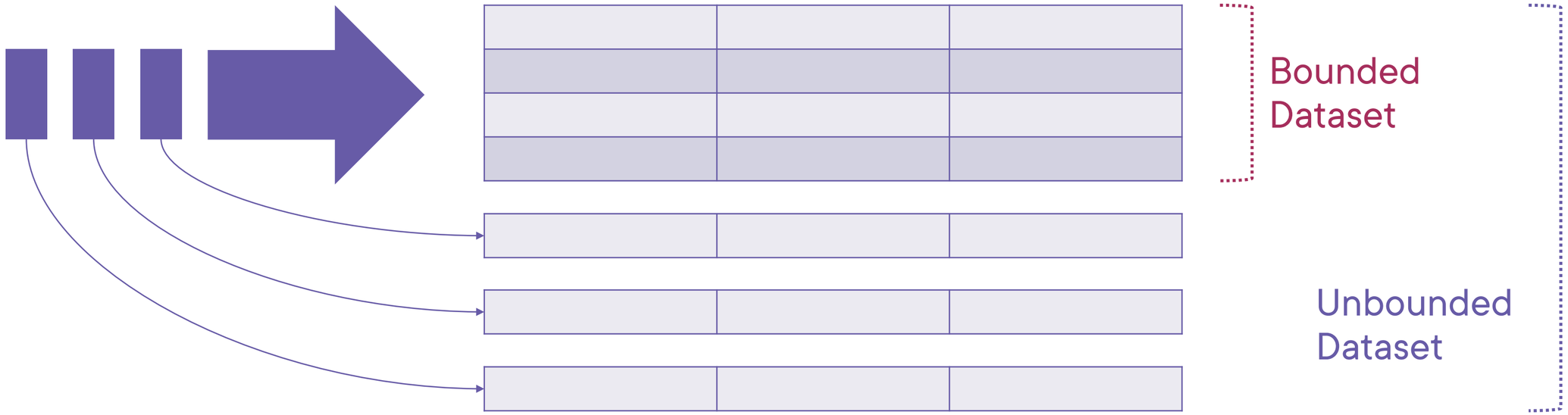
Spark Structured Streaming
is a scalable & fault-tolerant
stream processing engine
built on Spark SQL

Streaming Source

Streaming source should have support for offsets

Helps to track current read position in the stream

Unbounded Dataset



New events are like new rows added to Unbounded Dataset

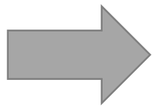
In Spark, it's a Streaming DataFrame

Structured Streaming Query

```
/* Extract data */
inputDF = spark.readStream \
                .format(“...”) \
                .load()

/* Transform data */
resultDF = inputDF \
                .where(...) \
                .withColumn(...) \
                .select(...)

/* Load data */
resultDF.writeStream \
                .format(“...”) \
                .start()
```

Execution Plan

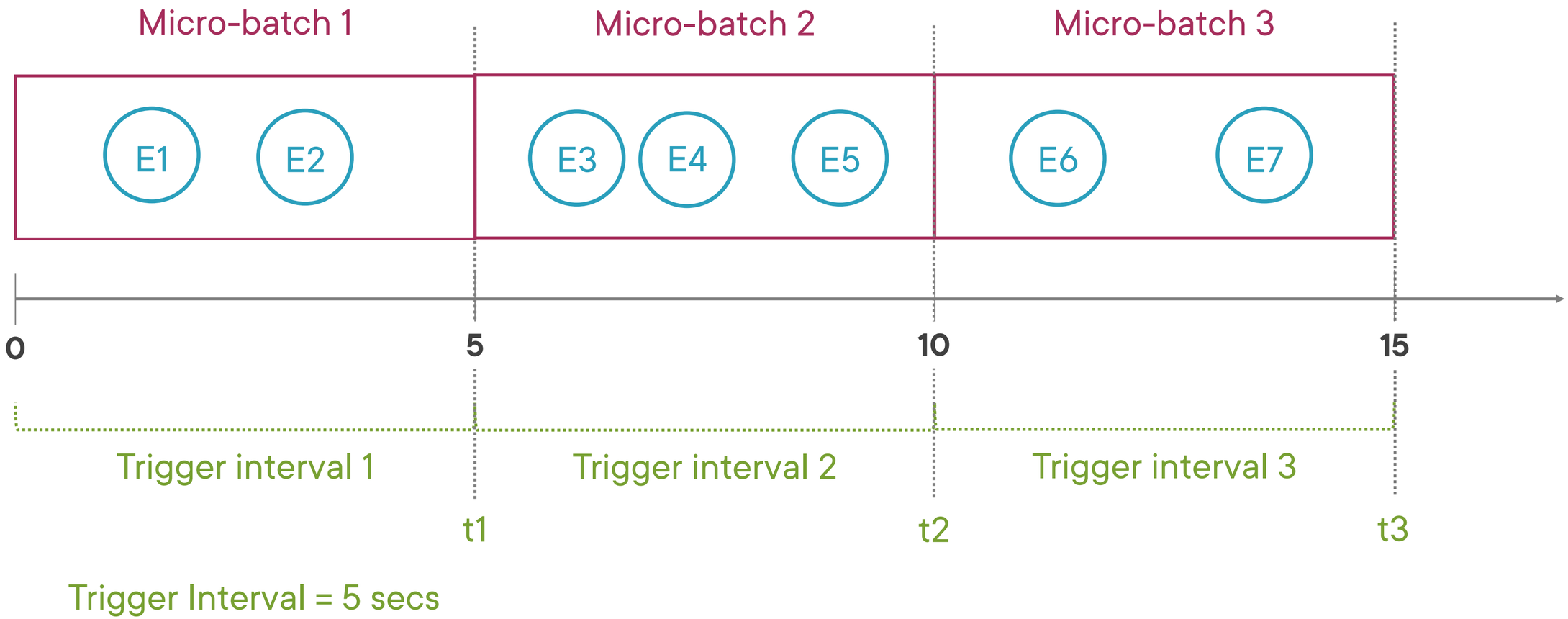
```
/* Extract data */
inputDF = spark.readStream \
            .format("...") \
            .load()

/* Transform data */
resultDF = inputDF \
            .where(...) \
            .withColumn(...) \
            .select(...)

/* Load data */
resultDF.writeStream \
            .format("...") \
            .start()
```



Optimized Query Plan

Catalyst Optimizer prepares an optimized query plan to execute streaming job

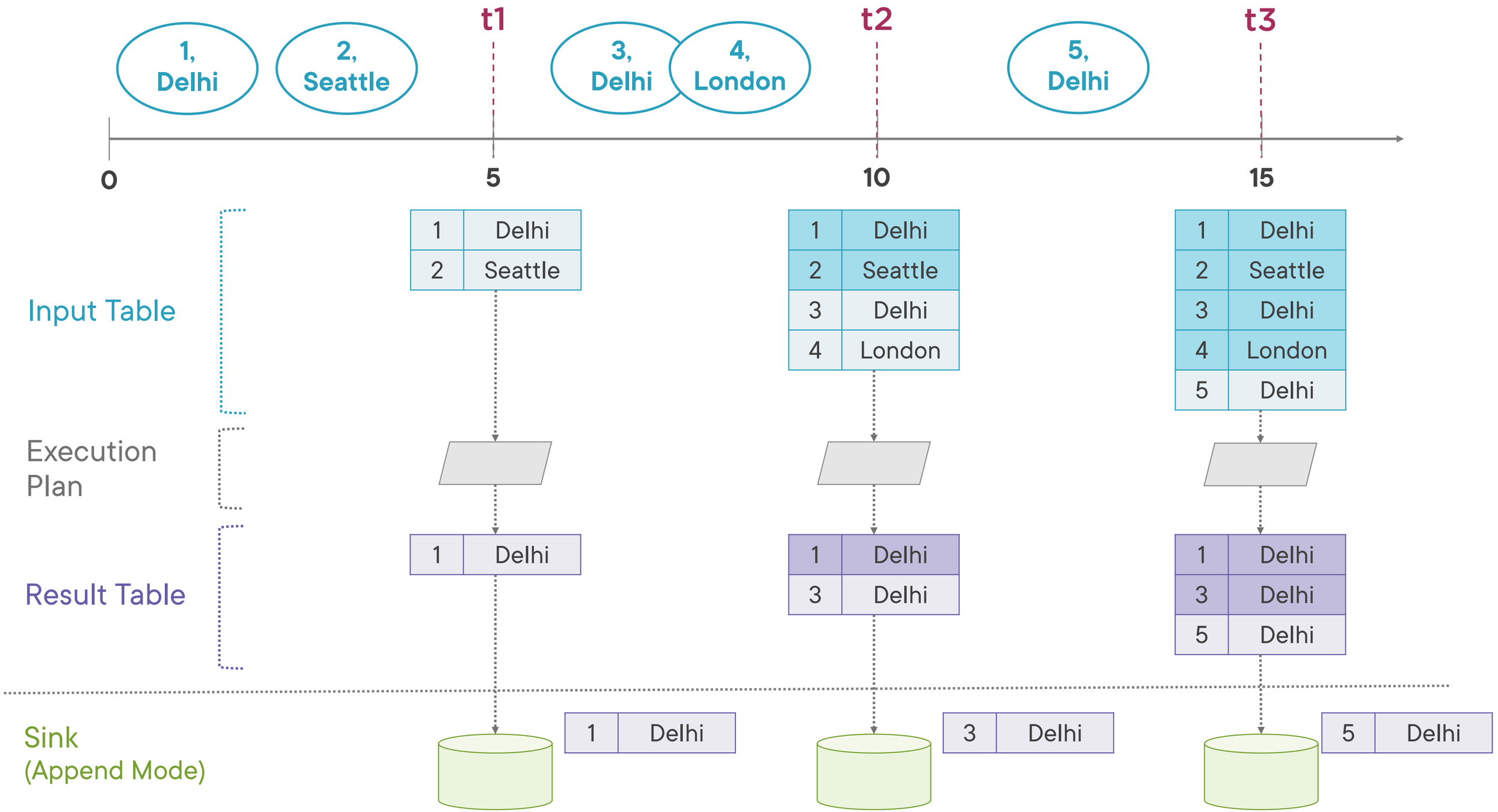
Triggers and Micro-batches



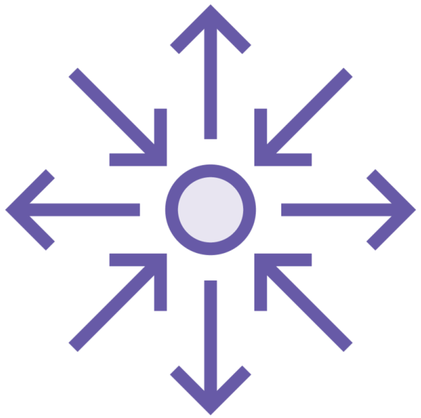
Trigger Interval = 5 secs

Adding Trigger to Query

```
/* Extract data */

inputDF = spark.readStream.format(“...”).load()

/* Transform data */

resultDF = inputDF \
                        .select(...) \
                        .withColumn(...) \
                        .withColumnRenamed(...) \

/* Load data */

resultDF.writeStream.format(“...”) \
        .trigger(processingTime=‘5 seconds') \
        .start()
```


t1
t2
t3
1, Delhi
2, Seattle
3, Delhi
4, London
5, Delhi
0
5
10
15
Input Table
Execution Plan
Result Table
Sink (Append Mode)
1 Delhi
2 Seattle
1 Delhi
2 Seattle
3 Delhi
4 London
1 Delhi
2 Seattle
3 Delhi
4 London
5 Delhi
1 Delhi
1 Delhi
3 Delhi
1 Delhi
3 Delhi
5 Delhi
1 Delhi
3 Delhi
5 Delhi

Output Mode



Defines what data is written from result table to sink

Mode Types

- Append
- Update
- Complete

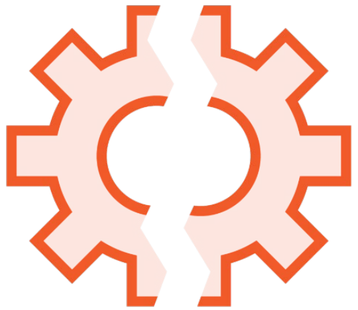
Streaming Source

E4 E3 E2 E1

4 3 2 1

Offsets

Spark Structured Streaming



E2 E1

2 1

Offsets: 1 to 2
Status: In progress
Status: Complete

Checkpoint Directory

Sink

E2 E1

Structured Streaming runs batch-like queries on streaming data using incremental execution plans, and provides fault tolerance

Extracting Streaming Data from Source

Streaming Sources

Must support offsets to provide fault tolerance

Spark has built-in sources

- File, Kafka, Socket etc.

Configure external sources

- Azure Event Hubs, Cosmos DB, AWS Kinesis etc.
- Delta Lake

Schema for source must be available

- Some sources provide the schema (like Event Hubs)
- For file-based sources, it needs to defined

Transforming and Loading Data

You can apply almost all transformations
on a Streaming DataFrame
as you apply on a Batch DataFrame

Streaming Sinks

Spark has built-in sinks

- File and Kafka
- For debugging – Console & Memory sinks
- For custom logic – ForEach & ForeachBatch sinks

Configure external sinks

- Azure Event Hubs, Cosmos DB, AWS Kinesis etc.
- Delta Lake

Each sink supports different Output Modes

Further Learning

Other concepts

- Stateless and stateful operations
- Types of timestamps
- Watermarking etc.

To learn more about Spark Structured Streaming, check out course

- Handling Streaming Data with Azure Databricks Using Spark Structured Streaming

Summary

Spark can build unified batch & streaming pipelines

Spark Structured Streaming

- Run queries on streaming data using micro-batches
- Provides fault-tolerance out-of-the-box

Build streaming pipeline

- Extract data from various sources like File System, Azure Event Hubs etc.
- Use console sink to display output
- Configure micro-batch interval
- Use common DataFrame methods like select, withColumn, withColumnRenamed etc.
- Configure checkpoint directory to track progress
- Use file sink to write to disk

Up Next:
Working with Spark in Cloud