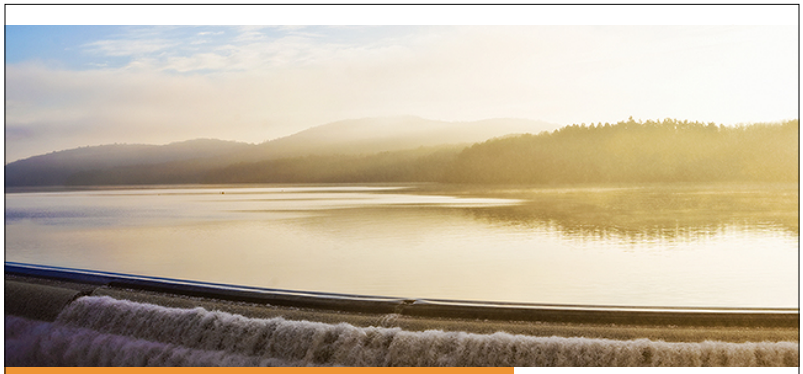


Spark Streaming with Datasets

What is Spark Streaming?

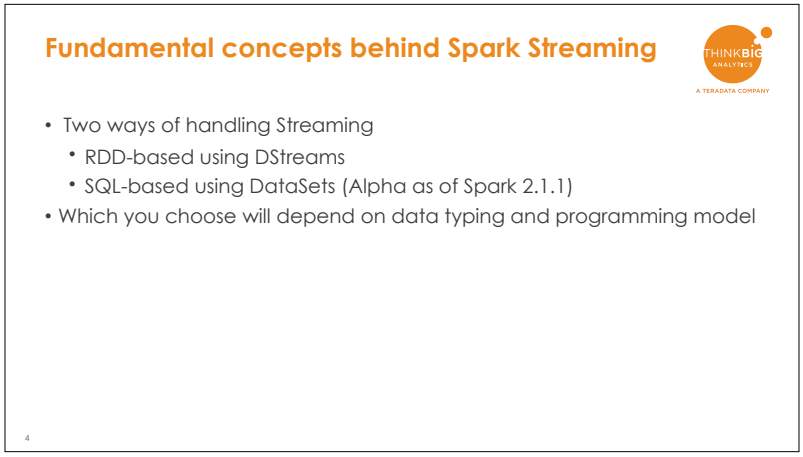
- Extension of Spark core
- Takes data real-time data from a variety of sources
- Writes data to HDFS, databases, and dashboards
- Processes that data in *microbatches*
- Can use Spark Machine Learning and Graph Processing on batches

2

Extension of Spark core
Takes data real-time data from a variety of sources
Writes data to HDFS, databases, and dashboards
Processes that data in microbatches
Can use Spark Machine Learning and Graph Processing on batches

Examples of Streaming Datasets

- Internet of Things (IoT) data
- Credit card transactions
- Files being ingested



Two ways of handling Streaming
RDD-based using DStreams
SQL-based using DataSets (Alpha as of Spark 2.1.1)
Which you choose will depend on data typing and programming model

DStreams Example (Unstructured Streaming)

Spark Supports Multiple Stream Types

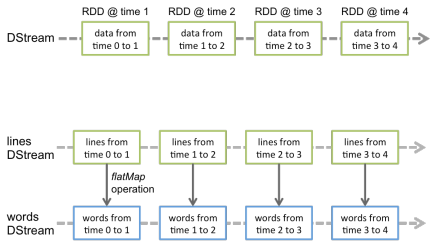
- HDFS directories
- Network connections

Additional advanced sources can be processed in stand-alone programs, but not from the spark-shell

- Kafka
- Flume
- Kinesis

What is a DStream?

- A Discretized Stream (DStream) is a series of RDDs
- Each RDD represents a microbatch of data for a certain time interval
- All operations on DStreams are applied to each microbatch



Spark Streaming Programs Have Two Components

- Receivers pull in data from an external source
- DStreams are created by receivers to create processing pipelines

The programmer must allocate an executor core for every Receiver and DStream

Example: `spark-shell --master local[2]` allows one Receiver and one DStream

Structure of a DStreams Streaming program

1. Define the input sources by creating input DStreams.
2. Write transformation and output operations for those DStreams.
3. Receive data and process it using `streamingContext.start()`.
4. Wait for the processing to end using `streamingContext.awaitTermination()`.
5. You can manually stop streaming by calling `streamingContext.stop()`.

File Streaming Example

A full discussion is in 01-Spark-DStreaming-Files.scala


```
// DStreams Wordcount example from files

import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}

    // Create the context
   val ssc = new StreamingContext(sc, Seconds(10))

   // Create the FileInputDStream on the directory and use the
   // stream to count words in new files created
   val lines = ssc.textFileStream("/tmp/streaming-input")
   val words = lines.flatMap(_.split(" "))
   val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
   wordCounts.print()
   ssc.start()
   ssc.awaitTermination()
```

Network Streaming

- We can also create DStreams for network ports
- Instead of sc.textFileStream, we use sc.socketTextStream with a socket number
- Everything else remains the same

Example: Ode to Wordcount on a DStream

A full discussion is in 02-Spark-DStreaming-Network.scala


```
// DStreams Wordcount example

import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._ // not necessary since Spark 1.3

// Create a local StreamingContext with two working thread and batch interval of 1 second.
// The master requires 2 cores to prevent from a starvation scenario.

val ssc = new StreamingContext(sc, Seconds(1))

val lines = ssc.socketTextStream("localhost", 9999)
val words = lines.flatMap(_.split(" "))

// Count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)

// Print the first ten elements of each RDD generated in this DStream to the console
wordCounts.print()
```

Structured Streaming

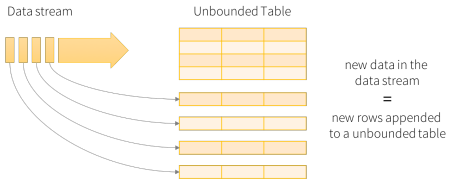
What is Structured Streaming with Datasets?

- As DStreams are to RDDs, Structured Streaming is to SparkSQL

	Spark Core	SparkSQL
Concept	Discretized Streams (DStreams)	Datasets
Non-streaming Analog	RDDs	Dataframes
Example operations	map, flatMap, reduce, filter, union	select, groupBy, join, arbitrary SQL queries
Outputs	RDDs	Files and Tables

Structured Streaming Concept

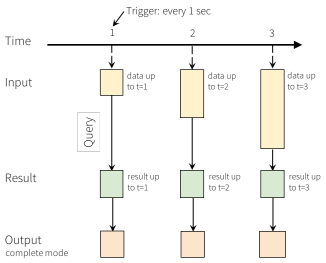
- A Dataset is simply a DataFrame that can grows with time
- New data from the data stream input appends rows to the table



Data stream as an unbounded table

Structured Streaming Concepts

- A query on the input will generate a "Result Table"
- New rows get appended to the input table every interval
- Every interval, the query gets run to update the results table
- Whenever results get updated, we write changed result rows to an external sync



Programming Model for Structured Streaming

Outputs Are What Get Written to External Storage

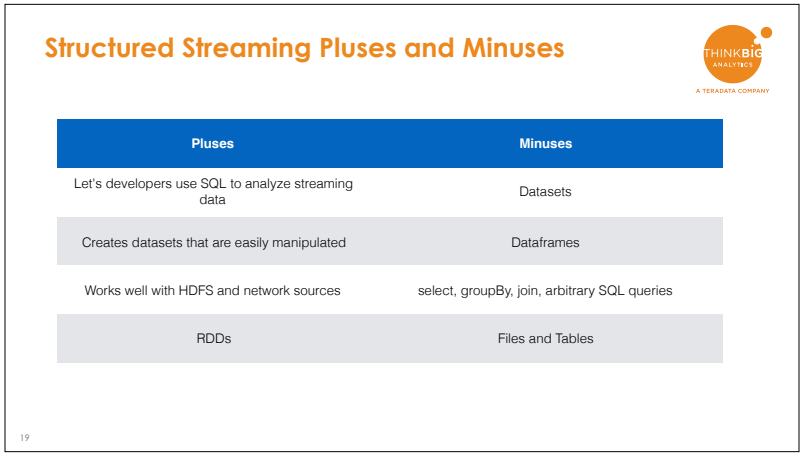
The output can be defined in a different mode:

- **Complete Mode:** he entire updated Result Table will be written to the external storage. It is up to the storage connector to decide how to handle writing of the entire table.
- **Append Mode:** Only the new rows appended in the Result Table since the last trigger will be written to the external storage. This is applicable only on the queries where existing rows in the Result Table are not expected to change.
- **Update Mode:** Only the rows that were updated in the Result Table since the last trigger will be written to the external storage (available since Spark 2.1.1). Note that this is different from the Complete Mode in that this mode only outputs the rows that have changed since the last trigger. If the query doesn't contain aggregations, it will be equivalent to Append mode.

Example: Wordcount using Structured Streaming

A full discussion is in 03-Spark-SStreaming-Network.scala


```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.SparkSession

// for a standalone Spark program, you'd need
// val spark = SparkSession
//   .builder
//   .appName("StructuredNetworkWordCount")
//   .getOrCreate()

import spark.implicits._

// Create DataFrame representing the stream of input lines from connection to localhost:9999
val lines = spark.readStream.format("socket").option("host", "localhost").option("port", 9999).load()

// Split the lines into words
val words = lines.as[String].flatMap(_.split(" "))

// Generate running word count
val wordCounts = words.groupBy("value").count()

// Start running the query that prints the running counts to the console
val query = wordCounts.writeStream.outputMode("update").format("console").start()
query.awaitTermination()
ssc.start()             // Start the computation
ssc.awaitTermination()  // Wait for the computation to terminate
```



Broken table

Summary

- Spark does streaming through micro-batching of data
- Spark has two models for streaming: DStreams and Structured Streaming
- At present, DStreams are more mature, but Structured Streaming holds promise for the future.