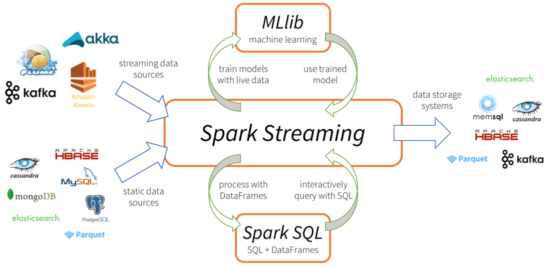
Big Data
Spark Structured Streaming

Dr. Wenceslao PALMA
wenceslao.palma@pucv.cl

ESCUELA DE INGENIERÍA INFORMÁTICA
PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Spark structured streaming



- provides fast, scalable and fault tolerant stream processing on Spark SQL engine.
- complex data and complex workloads.
- integrate with many storage systems.

Spark structured streaming: key ideas w.r.t. developing stream apps

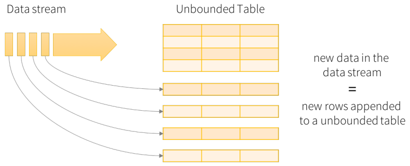
- you dont have to reason about streaming.
- you write simple batch queries.

Spark structured streaming: key ideas w.r.t. developing stream apps

- you dont have to reason about streaming.
- you write simple batch queries.

Spark automatically streamifies your queries

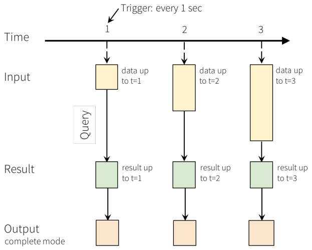
Spark structured streaming: programming model


Data stream
Unbounded Table
new data in the data stream = new rows appended to a unbounded table

Data stream as an unbounded table

- a data stream is processed as a table that is being countinuously appended.
- the stream computation is expressed as standard batch query on a static table.
- Spark runs this query as an incremental query on a unbounded input table.

Spark structured streaming: programming model



Programming Model for Structured Streaming

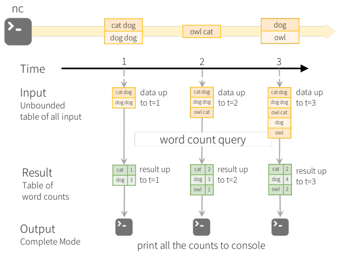
- Every new data item on the stream is a new row appended to the input table.
- A query on the input generates the result table.
- Every trigger interval (say, every 1 second) new rows are appended to the input table which eventually updates the result table

Spark structured streaming: programming model

The Output defines what is written out to the external storage, it can be defined in different modes:

- complete mode: the entire update result table is written to the external storage (or to the console).
- append mode: only the new rows appended in the result table (since the last trigger) are written to the external storage.
- update mode: not available yet in Spark 2.0. Only the rows updated in the result table since the last trigger will be written to the external storage.

Spark structured streaming: a quick example



Model of the Quick Example

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode
from pyspark.sql.functions import split

spark = SparkSession.builder().appName("StructuredNetworkWordCount").getOrCreate()
lines = spark.readStream.format('socket').option('host', 'localhost').option('port', 9999).load()

words = lines.select(explode(split(lines.value, ' ')).alias('word')

wordCounts = words.groupBy('word').count()

query = wordCounts.writeStream.outputMode('complete').format('console').start()

query.awaitTermination()
```

Spark structured streaming: DataFrames

- DataFrames are created through the DataStreamReader interface returned by SparkSession.readStream()
- We can specify details of the soure such as data format, schema, options,etc.
- Since the socket source does not provide end-to-end fault tolerance guarantees it should be be used only for testing.

```
spark = SparkSession. ...

userSchema = StructType().add("name", "string").add("age", "integer")
csvDF = spark.readStream().option("sep", ";").schema(userSchema).csv("/path/to/directory")
```

Spark structured streaming: DataFrames operations

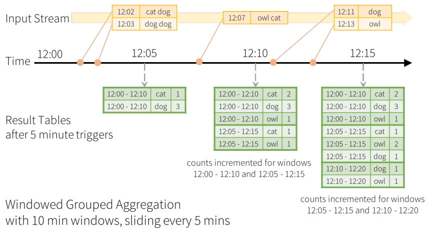
Selection, projection and aggregation:

```
df = ... # streaming DataFrame with schema{deviceName:string,type:string,signal:double,\
                                           time:DateType}

# Selection and projection
df.select("deviceName").where("signal > 10")

# Aggregation
df.groupBy("type").count()
```

Spark structured streaming: window operations


Input Stream
12:02 cat dog
12:03 dog dog
12:07 owl cat
12:11 dog
12:13 owl
Time
12:00
12:05
12:10
12:15
Result Tables
after 5 minute triggers
counts incremented for windows
12:00 - 12:10 and 12:05 - 12:15
counts incremented for windows
12:05 - 12:15 and 12:10 - 12:20
Windowed Grouped Aggregation
with 10 min windows, sliding every 5 mins

12:00 - 12:10	cat	1
12:00 - 12:10	dog	3

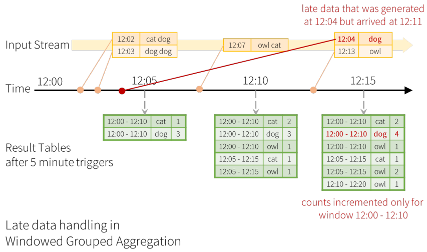
12:00 - 12:10	cat	2
12:00 - 12:10	dog	3
12:00 - 12:10	owl	1
12:05 - 12:15	cat	1
12:05 - 12:15	owl	1

12:00 - 12:10	cat	2
12:00 - 12:10	dog	3
12:00 - 12:10	owl	1
12:05 - 12:15	cat	1
12:05 - 12:15	owl	2
12:05 - 12:15	dog	1
12:10 - 12:20	dog	1
12:10 - 12:20	owl	1

```
words = ... # streaming DataFrame of schema { timestamp: Timestamp, word: String }

# Group the data by window and word and compute the count of each group
windowedCounts = words.groupBy(window(words.timestamp, '10 minutes', '5 minutes'),
                              words.word).count()
```

Spark structured streaming: window operations and late data



late data that was generated at 12:04 but arrived at 12:11
Input Stream
12:02 cat dog
12:03 dog dog
12:07 owl cat
12:04 dog
12:13 owl
Time
12:00
12:05
12:10
12:15
Result Tables after 5 minute triggers
counts incremented only for window 12:00 - 12:10
Late data handling in Windowed Grouped Aggregation

12:00 - 12:10	cat	1
12:00 - 12:10	dog	3

12:00 - 12:10	cat	2
12:00 - 12:10	dog	3
12:00 - 12:10	owl	1
12:05 - 12:15	cat	1
12:05 - 12:15	owl	1

12:00 - 12:10	cat	2
12:00 - 12:10	dog	4
12:00 - 12:10	owl	1
12:05 - 12:15	cat	1
12:05 - 12:15	owl	2
12:10 - 12:20	owl	1

Late data is automatically placed in the window w.r.t. the time it was generated

Spark structured streaming: join operations

```
staticDf = spark.read. ...
streamingDf = spark.readStream. ...
streamingDf.join(staticDf, "type")                 # inner equi-join
streamingDf.join(staticDf, "type", "right_join")  # right outer join
```