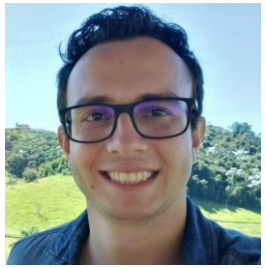
Spark – Big Data Processing

Aula 8

Semantix®

All about data

Quem sou eu?



Rodrigo Augusto Rebouças

Engenheiro de dados da Semantix
Instrutor do Semantix Academy

Contatos
rodrigo.augusto@semantix.com.br
linkedin.com/in/rodrigo-reboucas

RODRIGO.A.REBOUCAS

Semantix®
All about data

Struct Streaming

- Conceitos

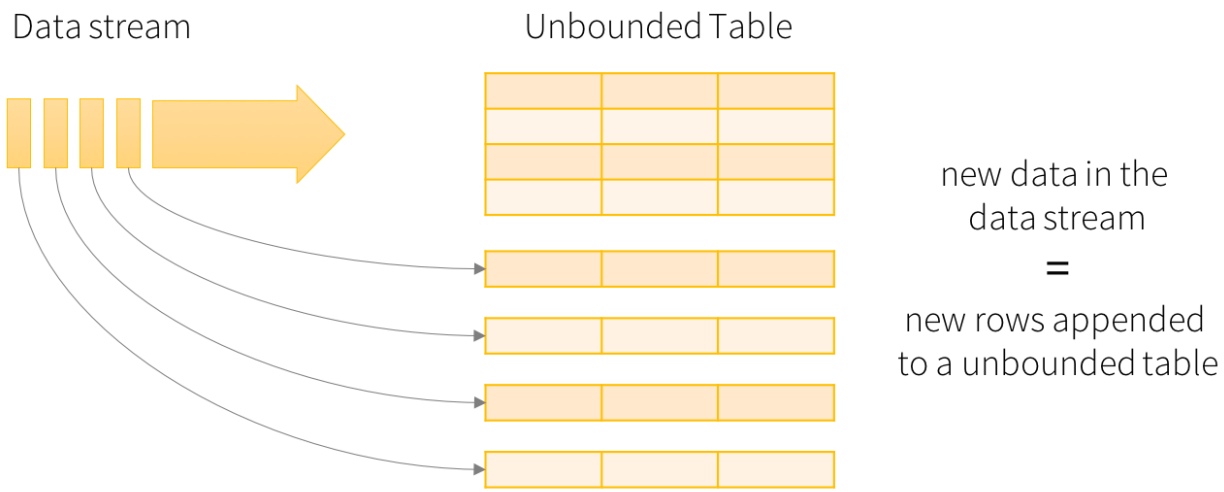
Semantix®
All about data

Struct Streaming

- Engine de processamento de stream construído na engine do Spark SQL
- Consultas do Spark Streaming são processadas usando uma engine de processamento de micro lote
  - Processar stream de dados como uma série de pequenos Jobs em batch
  - Latências de ponta a ponta de até 100 milissegundos
  - Tolerância de uma falha
- Novo modelo de stream - **Continuous processing**
  - Spark >= 2.3
  - Latências de ponta a ponta tão baixas quanto 1 milissegundo
  - Tolerância de uma falha
  - Sem alterar as operações Dataset / DataFrame em suas consultas

Struct Streaming

- Trata um stream de dados como uma tabela que está sendo continuamente anexada
- Modelo de processamento de stream muito semelhante a um modelo de processamento em lote



Data stream as an unbounded table

- https://spark.apache.org/docs/2.4.1/structured-streaming-programming-guide.html

Struct Streaming

- Leitura e Escrita de Stream

Semantix®
All about data

Struct Streaming – Exemplo Socket

- Exemplo de leitura na porta 9999 no localhost

```
read_str = spark.readStream.format("socket").
option("host", "localhost").option("port", 9999).load()
...
write_str = readStr.writeStream.format("console").start()
```

Struct Streaming – Exemplo CSV

- Leitura de um arquivo csv
  - Obrigatoriedade a definição do Schema
  - Leitura de diretório, não arquivo

```
user_schema = StructType().add(“nome", "string").add("idade", "integer")
read_csv_df = spark.readStream
  .schema(user_schema )
  .csv("/user/nomes/")
read_csv_df.printSchema()
```

Struct Streaming – Exemplo CSV

- Salvar dados
  - Orc
  - Json
  - Csv
  - Parquet

```
read_csv_df.writeStream
    .format("csv")
    .option("checkpointLocation", "/tmp/checkpoint")
    .option("path", "/home/data")
    .start()
```

Exercício 1 Spark – Structured Streaming

1. Criar uma aplicação em scala usando o spark para ler os dados da porta 9999 e exibir no console
2. Ler os arquivos csv “hdfs://namenode:8020/user/<nome>/data/iris/*.data” em modo streaming com o seguinte schema:
   - sepal_length float
   - sepal_width float
   - petal_length float
   - petal_width float
   - class string
3. Visualizar o schema das informações
4. Salvar os dados no diretório “hdfs://namenode:8020/user/<nome>/stream_iris/path” e o checkpoint em “hdfs://namenode:8020/user/<nome>/stream_iris/check”
5. Verificar a saida no hdfs e entender como os dados foram salvos
6. Bônus: Contar as palavras do exercício 1.

Struct Streaming com Kafka

- Conceitos

Semantix®
All about data

Struct Streaming com Kafka

- Structured Streaming
  - Versão >= Spark 2.0.0 (2.3)
  - https://spark.apache.org/docs/2.4.1/structured-streaming-kafka-integration.html
- Spark Streaming
  - Configurar os parâmetros do StreamContext
  - Configurar os parâmetros do Kafka
  - Configurar o Dstreams para leitura dos tópicos
- Structured Streaming
  - Configurar o Dataframe para leitura dos tópicos

Struct Streaming com Kafka

- Leitura

Semantix®
All about data

Leitura de dados do Kafka em Batch

- Criação do Kafka Souce para consultas batch

```
kafka_df = spark\
        .read\
        .format("kafka")\
        .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
        .option("subscribe", "topic1")\
        .load()
```

Leitura de dados do Kafka em Stream

- Criação do Kafka Souce para consultas streaming

```
kafka_df = spark\
        .readStream\
        .format("kafka")\
        .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
        .option("subscribe", "topic1")\
        .option("startingOffsets", "earliest")\
        .load()
```

Struct Streaming com Kafka

- Visualização e Escrita

Semantix®
All about data

Visualizar dados do Kafka em Batch

- Para visualizar a Chave e valor e necessário fazer cast

```
kafka_df.printSchema
root
|-- key: binary (nullable = true)
|-- value: binary (nullable = true)
|-- topic: string (nullable = true)
|-- partition: integer (nullable = true)
|-- offset: long (nullable = true)
|-- timestamp: timestamp (nullable = true)
|-- timestampType: integer (nullable = true)

kafka_df.select(col("key").cast(StringType), col("value").cast(StringType)).show()
```

Visualizar dados do Kafka em Stream

- Para visualizar a Chave e valor e necessário fazer cast

```
kafka_df.printSchema
root
|-- key: binary (nullable = true)
|-- value: binary (nullable = true)
|-- topic: string (nullable = true)
|-- partition: integer (nullable = true)
|-- offset: long (nullable = true)
|-- timestamp: timestamp (nullable = true)
|-- timestampType: integer (nullable = true)

kafka_df.select(col("key").cast(StringType), col("value").cast(StringType))

kafka_df.writeStream.format("console").start
```

Enviar dados Stream para o Kafka

- Fazer uso do **Continuous Processing** (Experimental)
  - Registrar o progresso da consulta a cada x tempo com o Trigger Continuos
  - O número de tarefas exigidas pela consulta depende de quantas partições a consulta pode ler das fontes em paralelo (Núcleos >= Partições)

```
kafka_df.writeStream\
        .format("kafka")\
        .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
        .option("topic", "topic_teste2")\
        .trigger(Trigger.Continuous("1 second"))\
        .start()
```

Enviar dados Batch para o Kafka

- Obrigatório ter o campo value
- Opcional ter o campo key

```
dataframe\
        .withColumnRenamed(“id”, “key”)\
        .withColumnRenamed(“nome”, “value”)
dataframe.write\
        .format("kafka")\
        .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
        .option("topic", "topic_teste2")\
        .save()
```

Exercício 2 Spark – Structured Streaming

1. Ler o tópico do kafka “topic-kvspark” em modo batch
2. Visualizar o schema do tópico
3. Visualizar o tópico com o campo key e value convertidos em string
4. Ler o tópico do kafka “topic-kvspark” em modo streaming
5. Visualizar o schema do tópico em streaming
6. Alterar o tópico em streaming com o campo key e value convertidos para string
7. Salvar o tópico em streaming no tópico topic-kvspark-output a cada 5 segundos
8. Salvar o tópico na pasta hdfs://namenode:8020/user/<nome>/Kafka/topic-kvspark-output

Semantix®

All about data

contato@semantix.com.br

www.semantix.com.br