PÓS-GRADUAÇÃO

Linguagens de programação para ciência de dados (*Python* com *Spark*)

PÓS-GRADUAÇÃO

Processando *Big Data* com *Spark*

Bloco 1

Marcelo Tavares de Lima

Objetivos

- Apresentar conceitos fundamentais de *Big Data* e *Apache Spark*.
- Descrever como se utiliza *Apache Spark* para realizar operações e consultas em *Big Data*.
- Apresentar exemplos de algoritmos para manipulação de arquivos, utilizando a linguagem de programação *Python* e *Apache Spark*.

Apache Spark e *Big Data*

- *Apache Spark* é um *framework* para processamento de grande volume de dados (*Big Data*) e tem como principais características: velocidade no processamento de grande volume de dados, suporte para diversos formatos de dados, facilidade de uso, e suporte para diversos tipos de linguagem de programação como *Python*, Java, R e Scala.

Apache Spark e *Big Data*

- **Dados estruturados:** geralmente, estão armazenados em bancos de dados relacionais (*Relational Database Management Systems* - RDBMS). Nesse esquema, os dados estão bem organizado para facilitar a consulta e extração de informação. Por esse motivo, os dados estruturados trazem benefícios inerentes ao lidar com grandes volumes de informações.

Apache Spark e *Big Data*

- **Semiestruturado:** salvos em estruturas de dados chamadas de registros, mas não necessariamente possuem um esquema global bem definida, por exemplo: JSON e XML. Os benefícios dos formatos de dados semiestruturados são que fornecem maior flexibilidade para expressar seus dados, pois cada registro é autodescritivo. Esses formatos são muito comuns em muitos aplicativos, pois existem muitos API em quase todas as linguagens de programação para lidar com esses tipos de dados.

Apache Spark e *Big Data*

- **Dados não estruturados:** geralmente, texto de forma livre ou objetos binários que não contêm marcação ou metadados.
    - Exemplos: artigos de jornais, imagens e *blogs* de aplicativos.
- Esses tipos de dados, geralmente, exigem que o contexto em torno dos dados seja analisável.
- A desvantagem desse tipo de dados está no processo de extração de valores dessas fontes de dados, pois são necessárias muitas transformações e técnicas de mineração e filtragem para interpretar esses conjuntos de informações.

Apache Spark e *Big Data*

- O *Apache Spark* é uma plataforma de computação em *cluster*, projetada para trabalhar com grande volume de dados (*Big Data*) de forma simples e eficiente, segundo Karau (2015).
- Foi desenvolvido na linguagem Scala e executa em uma máquina virtual Java (*Java Virtual Machine* - JVM). Na versão atual, tem suporte para as seguintes linguagens de programação: *Python*, R, Scala e Java, segundo Chambers (2018).

Apache Spark e *Big Data*

- Bibliotecas *Spark*:
  - ***Spark SQL*:** biblioteca mais importante do *framework Apache Spark*. Por meio dela você pode executar consultas SQL nativas em dados estruturados ou semiestruturados. Tem suporte para linguagem em Java, Scala, *Python* e R.
  - ***Spark Streaming*:** biblioteca usada para processar dados de *streaming* em tempo real. Dessa forma, podemos desenvolver algoritmos para processamento de dados à medida que os dados chegam (em tempo real) e não em um processo em lote.

Apache Spark e *Big Data*

- Bibliotecas *Spark*:
  - ***Spark MLlib*:** biblioteca de aprendizado de máquina (*Machine Learning*), que consiste em diversos algoritmos de aprendizagem de máquina supervisionado e não supervisionado, incluindo classificação, regressão e agrupamento (*clustering*).
  - ***Spark GraphX*:** é uma API do *Spark* para trabalhar com grafos e computação paralela. O GraphX contém uma biblioteca de algoritmos para simplificar tarefas de análise de grafos.

Apache Spark e *Big Data*

- O *Apache Spark* contém duas estruturas de dados para trabalhamos com coleções distribuídas: *DataFrame* e *DataSet*.
- São estruturas de dados que armazenam os dados em forma de tabela com linhas e colunas.

Apache Spark e *Big Data*

- Recursos de um *DataFrame*:
  - Capacidade de processar os dados no tamanho de *kilobytes* para *petabytes* em um *cluster* de nó único.
  - Suporta diferentes formatos de dados (*Elastic Search* e Cassandra) e sistemas de armazenamento (HDFS, MySQL etc.).
  - Pode ser facilmente integrado a todas as ferramentas e estruturas de *Big Data* via *Spark-Core*.
  - Fornece bibliotecas de funções para *Python*, Java, Scala e R.

PÓS-GRADUAÇÃO

Processando *Big Data* com *Spark*

Bloco 2

Marcelo Tavares de Lima

Apache Spark e *Big Data*

- Considere que desejamos criar um *DataFrame* com o *Apache Spark*.
- Será necessário importar as classes *SparkSession* e *SparkContext* do pacote *pyspark*.

Apache Spark e *Big Data*

- O *DataFrame* a ser criado conterá dados de número de voos internacionais realizados no mês de setembro de 2019.
- Os dados a serem importados estão no formato CSV.

Apache Spark e *Big Data*

- Os dados em CSV são:

EMPRESA, PAIS_ORIGEM,PAIS_DESTINO, QTDE_VOO

Latam, Brasil, EUA, 3000

KLM, Brasil, Itália, 500

Gol, Brasil, Irlanda, 700

KLM, Brasil, Londres, 2500

Azul, Brasil, Portugal, 100

Apache Spark e *Big Data*

- O código fonte em *Python* é:

```
import pyspark

from pyspark.context import SparkContext

from pyspark.sql.session import
SparkSession

sc = SparkContext.getOrCreate()

spark = SparkSession(sc)

df_csv = spark.read.option("inferSchema",
"true").option("header",
"true").csv("<caminho_do_arquivo_csv>")

 print  df_csv.show()
```

Apache Spark e *Big Data*

- Também, é possível importar os dados no formato JSON.

Apache Spark e *Big Data*

- voos_setembro_2019.json

{"EMPRESA":"Latam","PAIS_ORIGEM":"Brasil","PAIS_DESTINO": "EUA", "QTDE_VOO":"3000"}
{"EMPRESA":"KLM","PAIS_ORIGEM":"Brasil","PAIS_DESTINO":"Itália", "QTDE_VOO":"500"}
{"EMPRESA":"Gol","PAIS_ORIGEM":"Brasil", "PAIS_DESTINO":"Irlanda", "QTDE_VOO":"7000"}
{"EMPRESA":"KLM","PAIS_ORIGEM":"Brasil","PAIS_DESTINO":"Londres", "QTDE_VOO":"2500"}
{"EMPRESA":"Azul","PAIS_ORIGEM":"Brasil","PAIS_DESTINO":"Portugal", "QTDE_VOO":"100"}

Apache Spark e *Big Data*

- Código fonte para ler dados em JSON.

```
import pyspark
from pyspark.context import SparkContext
from pyspark.sql.session import
SparkSession
sc = SparkContext.getOrCreate()
spark = SparkSession(sc)
df_json =
spark.read.format("json").load("<caminho_
do_arquivo_json>")
print  df_json.show()
```

Apache Spark e *Big Data*

- Saída:

Figura 1 - Saída

```
+-------+------------+-----------+--------+
|EMPRESA|PAIS_DESTINO|PAIS_ORIGEM|QTDE_VOO|
+-------+------------+-----------+--------+
|  Latam|         EUA|     Brasil|    3000|
|    KLM|      Itália|     Brasil|     500|
|    Gol|     Irlanda|     Brasil|    7000|
|    KLM|     Londres|     Brasil|    2500|
|   Azul|    Portugal|     Brasil|     100|
+-------+------------+-----------+--------+
```

Fonte: elaborado pelo autor.

Apache Spark e *Big Data*

- Pela biblioteca *Spark SQL*, você também pode executar consultas SQL por meio do método sql da classe *SparkSession*.
- O método sql retorna um objeto *DataFrame,* utilizando os dados do arquivo voos_setembro_2019.json. Desenvolveremos um algoritmo para salvar os dados numa visão (*view*) temporária.

Apache Spark e *Big Data*

- Código fonte em linguagem *Python*:

```
import pyspark

from pyspark.context import SparkContext

from pyspark.sql.session import SparkSession

sc = SparkContext.getOrCreate()

spark = SparkSession(sc)

df = spark.read.json("<caminho_do_arquivo_json>")

# Criando um view temporária usando o DataFrame

df.createOrReplaceTempView("nome_view")

# Visualizando os dados da tabela

sc.sql("select name from <nomel_view>").show()
```

PÓS-GRADUAÇÃO

Teoria em prática

Bloco 3

Marcelo Tavares de Lima

Apache Spark e *Big Data*

- A quantidade de dados criados e armazenados globalmente continua crescendo a cada ano. Esses dados são classificados em três grupos: não estruturado (*logs* de servidores e aplicativos, imagens e vídeos de câmera de segurança), semiestruturado (XML, CSV e JSON) e estruturado (banco de dados).
- Utilizando o *framework Apache Spark*, como você processaria esses dados, de forma extrair informações importantes para empresa?

PÓS-GRADUAÇÃO

Dica do professor

Bloco 4

Marcelo Tavares de Lima

Dica de site

Figura 2 - Site

Fonte: infoq.com. Acesso em: 21 jan. 2020.

Referências

CHAMBERS, B.; ZAHARIA, M. **Spark:** the definitive guide: Big Data processing made simple. San Francisco: O'Reilly Media, 2018.

KARAU, H., KONWINSKI, A., WENDELL, P., ZAHARIA, M. **Learning spark**: lightning-fast big data analysis. O'Reilly Media, 2015.