Semantix

Spark - Básico

Aula 9

Quem sou eu?

Eu sou Rodrigo Augusto Rebouças.

Engenheiro de dados da Semantix
Instrutor do Semantix Mentoring Academy

Você pode me encontrar em:
rodrigo.augusto@semantix.com.br

Processamento de Dados

Spark

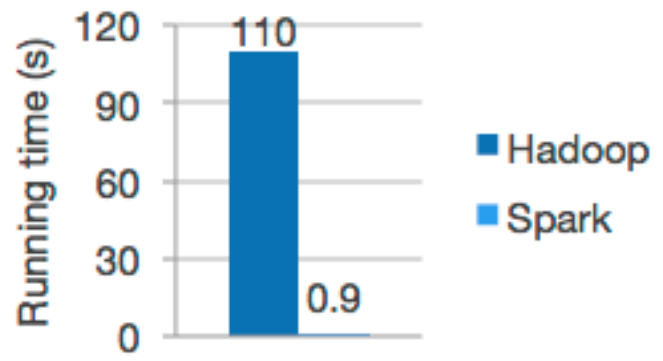
Semantix

Processamento de Dados

- Map Reduce
- Spark
- Fink

Apache Spark

- Plataforma de computação em cluster
  - Suporte
    - MapReduce
    - Streaming
    - Análises interativas
  - Execução em memória
  - Compatível com Hadoop
    - Funciona com YARN
    - Acessar os dados
      - HDFS
      - Tabelas Hive
      - Tabelas Hbase
- Linguagem: Scala, Java, Python e R



Logistic regression in Hadoop and Spark

Semantix

Spark História

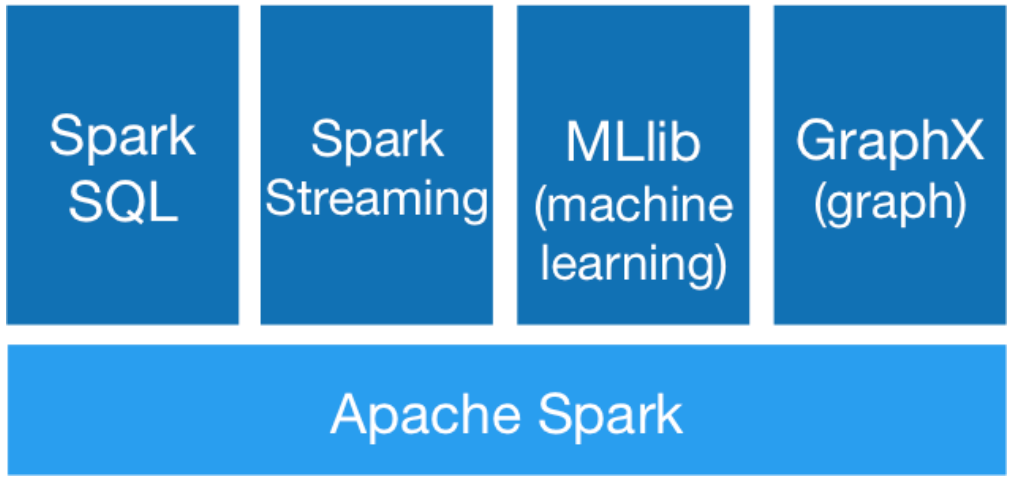
- 2009
  - Projeto de pesquisa na Universidade da California, Berkeley
    - AMPLab
- 2010
  - Open source
- 2013
  - Apache
- Representação dos dados
  - RDD - 2011
  - Dataframe - 2013
  - Dataset - 2015
  - Qual a melhor forma de trabalhar com dados no Spark?

Estrutura e inicialização

Semantix

Spark Ferramentas

- **Spark**
  - ETL e processamento em batch
- **Spark SQL**
  - Consultas em dados estruturados
- Spark Streaming
  - Processamento de stream
- Spark MLib
  - Machine Learning
- Spark GraphX
  - Processamento de grafos



Spark Shell

- Ambiente Spark interativo
- Inicialização no Cluster
  - pyspark para Python
  - spark-shell para Scala

- Se existir a instalação da versão 1 e 2 no cluster
  - Versão 1
    - pyspark para Python
    - spark-shell para Scala
  - Versão 2
    - pyspark2 para Python
    - spark2-shell para Scala

DataFrame

Introdução

Semantix

DataFrame Indrodução

- Representação de dados no spark
- DataFrames
  - Dados estruturados e semiestruturados em forma tabular
  - Java, Scala e Python
  - Operações
    - Transformação
    - Ação

Leitura

DataFrame

Semantix

DataFrame Leitura de arquivo

- Dados suportados para leitura e escrita no DataFrame
  - Text files
    - CSV
    - JSON
    - Plain text
  - Binary format files
    - Apache Parquet (Muito utilizado)
    - Apache ORC
  - Tables
    - Hive metastore
    - JDBC
  - Configurar outros tipos

DataFrame Leitura de arquivo

- val <dataframe> = spark.read.<formato>(“<arquivo>”)
  - <formato>
    - textFile(“arquivo.txt”)
    - csv(“arquivo.csv”)
    - jdbc(jdbcUrl, "bd.tabela", connectionProperties)
    - load ou parquet(“arquivo.parquet”)
    - table(“tabelaHive”)
    - json(“arquivo.json“)
    - orc(“arquivo.orc”)
  - <arquivo>
    - “diretório/”
    - “diretório/*.log”
    - “arq1.txt, arq2.txt”
    - “arq*”

DataFrame Leitura de arquivo

- val <dataframe> = spark.read.format("<formato>").load("<arquivo>")
- Opções para configurar o arquivo de leitura
  - spark.read.option(...)

```
scala> val userDF = spark.read.json(“user.json”)

scala> val userDF = spark.read.format(“json”).load(“usr.json”)
```

Ações

DataFrame

Semantix

DataFrame Operações

- Ação
  - count: retorna o número de linhas
  - first: retorna a primeira linha
  - take(n): retorna as primeiras n linhas como um array
  - show(n): exibe as primeiras n linhas da tabela
  - collect: Trazer toda a informação dos nós do drive
    - Cuidado para não estourar a memória
  - distincts: retorna os registros, removendo os repetidos
  - write: salvar os dados
  - printSchema() Visualizar a estrutura dos dados
    - DataFrame sempre tem um esquema associado

DataFrame Exemplo ações

- <dataframe>.<ação>

```
scala> val clienteDF = spark.read.json("cliente.json")
scala> clienteDF.printSchema()
scala>  clienteDF.count()
scala>  clienteDF.first()
scala> clienteDF.take(5)
scala> clienteDF.show(5)
scala> clienteDF.distinct()
scala> clienteDF.collect() //Cuidado para estourar memória
```

DataFrame Salvar Dado

- dadosDF.write.
  - save(“arquivoParquet”)
  - json("arquivoJson")
  - csv(“arquivocsv”)
  - saveAsTable("tableHive")
    - (/user/hive/warehouse)

```
scala> dadosDF.write.save(“outputData”)
$ hdfs dfs -ls /user/cloudera/outputData

scala> dadosDF.write. \
mode("append"). \
option("path","/user/root"). \
saveAsTable("outputData")
```

Transformações

DataFrame

Semantix

DataFrame Operações

- Transformação
  - Imutáveis
    - Dados no DataFrame nunca são modificados
  - Exemplos
    - select: Selecionar os atributos
    - where: Filtrar os atributos
    - orderBy: Ordenar os dados por atributo
    - groupBy: Agrupar os dados por atributo
    - join: Mesclar Dados
    - limit(n): Limitar a quantidade de registros

DataFrame Transformações

- Transformação individual
  - <dataframe>.<ação>
- Sequência de transformações
  - <dataframe>.<ação>.<ação>.<ação>.<ação>

```
scala> val prodQtdDF = prodDF.select("nome","qtd")

scala> val qtd50DF = prodQtdDF.where("qtd > 50")

scala> val qtd50ordDF = qtd50DF.orderBy(“nome”)

scala> qtd50ordDF.show()

scala> prodDF.select("nome","qtd").where("qtd > 50").orderBy(“nome”)
```

DataFrame – Transformações

- Agrupar (groupBy) com uma função de agregação
  - count
  - max
  - min
  - mean
  - sum
  - pivot
  - agg (Funções de agregação adicional)

```
scala> peopleDF.groupBy(“setor").count()

scala> peopled.show()
```

DataFrame – Transformações

- Acessar o atributo do dado
  - “<atributo>”
  - $“<atributo>”
  - <dataframe>(“<atributo>”)

```
scala> prodDF.select("nome","qtd").show()

scala> prodDF.select($"nome", $“qtd” * 0,1).show()

scala> prodDF.where(prodDF("nome").startsWith("A")).show()
```

Laboratório

Resolução de Exercícios

Semantix

Exercícios DataFrame

1. Enviar o diretório local “/input/exercises-data/juros_selic” para o HDFS em “/user/aluno/<nome>/data”
2. Criar o DataFrame **jurosDF** para ler o arquivo no HDFS “/user/aluno/<nome>/data/juros_selic/juros_selic.json”
3. Visualizar o Schema do jurosDF
4. Mostrar os 5 primeiros registros do jutosDF
5. Contar a quantidade de registros do jurosDF
6. Criar o DataFrame **jurosDF10** para filtrar apenas os registros com o campo “valor” maior que 10
7. Salvar o DataFrame jurosDF10 como tabela Hive “<nome>.tab_juros_selic”
8. Criar o DataFrame **jurosHiveDF** para ler a tabela “<nome>.tab_juros_selic”
9. Visualizar o Schema do jurosHiveDF
10. Mostrar os 5 primeiros registros do jurosHiveDF
11. Salvar o DataFrame jurosHiveDF no HDFS no diretório “/user/aluno/nome/data/save_juros” no formato parquet
12. Visualizar o save_juros no HDFS
13. Criar o DataFrame **jurosHDFS** para mostrar o “save_juros” da questão 8
14. Visualizar o Schema do jurosHDFS
15. Mostrar os 5 primeiros registros do jurosHDFS

Semantix

Obrigado!

Alguma pergunta?

Você pode me encontrar em:

rodrigo.augusto@semantix.com.br

GET SMARTER