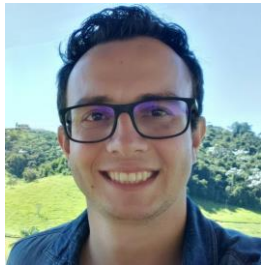
Spark – Big Data Processing

Aula 3

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

RDD

- Conceitos

Semantix®
All about data

RDD

- **R**esilient **D**istributed **D**atasets
  - **R**esiliente - Recriar dado perdido na memória
  - **D**istribuído - Processamento no cluster
  - **D**atasets - Dados podem ser criados ou vir de fontes

- Coleção de objetos distribuídos entre os nós do cluster
  - Armazena os dados em partições
- Imutáveis
- Tipos de operações
  - Ação
  - Transformação

RDD - Operações

- Ação: Retorna um valor
  - Collect
  - Count
  - First
  - Take
  - Reduce
  - CountByKey
  - Foreach
- Transformação: Retorna um RDD
  - Map
  - Filter
  - FlatMap
  - GroupByKey
  - ReduceByKey
  - AggregateByKey

Leitura e visualização de dados

- RDD

Semantix®
All about data

RDD – Leitura e visualização de dados

```
rdd = sc.textFile("entrada*")
rdd: org.apache.spark.rdd.RDD[String]

rdd.count()
res2: Long = 12

rdd.first()
res3: String = Big Data

rdd.take(5)
res1: Array[String] = Array(Big Data, 2019, Semantix, Hadoop, Semantix SP)

# Mostrar dados - Cuidado para não estourar memória
rdd.collect()
rdd.foreach(println) #Mostrar os valores sem Array em Scala
```

entrada1.txt	**entrada2.txt**
Big Data	Curso Hadoop
2019	Data
Semantix	Hadoop
Hadoop	Semantix
Semantix SP	SP
Hadoop 2019	Data

Transformações de Map e FlatMap

- RDD

Semantix®
All about data

RDD – FlatMap

Scala

```
rdd.take(2)
res1: Array[String] = Array(Big Data, Semantix SP)

val palavras = rdd.flatMap(x => x.split(" "))
//Operador Coringa
val palavras = rdd.flatMap(_.split(" "))

palavras.foreach(println)
'Big',
'Data',
‘Semantix',
‘SP’
```

Python

```
rdd.take(2)
['Big Data', 'Semantix SP’]

palavras = rdd.flatMap(lambda x: x.split(" "))

palavras.collect()
['Big',
 'Data',
 ‘Semantix',
 ‘SP’]
```

Função Anônima

Python

```
rdd.take(2)
res1: Array[String] = Array(Big Data, Semantix SP)

p = rdd.flatMap(lambda x: x.split(" "))

min = p.map(lambda linha: linha.lower())

min.collect()
['big',
 'data',
 'semantix',
 'sp']
```

```
def Func(linha):
    linha = linha.lower()
    return linha

minuscula= p.map(Func)

minuscula.collect()
['big',
 'data',
 'semantix',
 'sp']
```

RDD – Map e FlatMap

```
rdd.take(2)
['Big Data', 'Semantix SP’]

palavras = rdd.flatMap(lambda x: x.split(" "))

palavras.collect()
['Big',
 'Data',
 ‘Semantix',
 ‘SP’]

palavras = rdd.map(lambda x: x.split(" "))

palavras.collect()
[['Big', 'Data'],
 ['Semantix’, ’SP’]]
```

- Diferença entre map e flatMap?

RDD – Transformações no Map

Scala

```
val pMinuscula = palavras.map(_.toLowerCase)
val pMaiuscula = palavras.map(_.toUpperCase)
val pChaveValor = pMinuscula.map((_,1))
pChaveValor.take(4)
```

res1: Array[(String, Int)] = Array((big,1), (data,1), (2019,1), (semantix,1))

Python

```
pMinuscula = palavras.map(lambda linha: linha.lower())
pMaiuscula = palavras.map(lambda linha: linha.upper())
pChaveValor = pMinuscula.map(lambda palavra: (palavra,1))
pChaveValor.take(4)
```

[('big',1), ('data',1), ('2019',1), ('semantix',1)]

Transformações de Filter e Reduce

- RDD

Semantix®
All about data

RDD – Transformações de Filter

- Remover dados do RDD

Scala

```
val filtroA = palavras.filter(_.startsWith("a"))
val filtoTamanho = palavras.filter(_.lenght > 5)
val numPar = numero.filter(_ % 2 == 0)
```

Python

```
filtro_a = palavras.filter(lambda palavra: palavra.startswith("a"))
filtro_tamanho = palavras.filter(lambda palavra: len(palavra)>5)
num_par = numeros.filter(lambda numero: numero % 2 == 0)
```

RDD – Transformações de Reduce

entrada1.txt	entrada2.txt
Big Data	Curso Hadoop
2019	Data
Semantix	Hadoop
Hadoop	Semantix
Semantix SP	SP
Hadoop 2019	Data

Scala

```
val pChaveValor = pMinuscula.map((_,1))

pReduce = pChaveValor.reduceByKey(_+_)

pReduce.take(3)
res4: Array[(String, Int)] = Array((big,1), (2019,2), (hadoop,4))
```

Python

```
p_chave_valor = pMinuscula.map(lambda palavra: (palavra,1))

p_reduce = p_chave_valor.reduceByKey(lambda key1, key2: key1 + key2)

pReduce.take(3)
[('big', 1),
 (2019', 2),
 ('hadoop', 4)]
```

Transformações de Ordenação

- RDD

Semantix®
All about data

RDD – Transformações de Ordenação

entrada1.txt	entrada2.txt
Big Data	Curso Hadoop
2019	Data
Semantix	Hadoop
Hadoop	Semantix
Semantix SP	SP
Hadoop 2019	Data

Scala

```
pOrdena = pReduce.sortBy(-_._2)
pOrdena: org.apache.spark.rdd.RDD[(String, Int)]

pOrdena.take(4)
res25: Array[(String, Int)] = Array((hadoop,4), (semantix,3), (data,3), (2019,2))
```

Python

```
p_ordena = p_reduce.sortBy(lambda palavra: palavra[1], False)


p_ordena.take(4)
[('hadoop', 4),
 ('semantix', 3),
 ('data', 3),
 ('2019', 2)]
```

Armazenamento e visualização de dados

- RDD

Semantix®
All about data

RDD – Visualização

Scala

```
pOrdena.foreach(y => println(y._1 + " - " + y._2))
hadoop - 4
semantix - 3
data - 3
2019 - 2
sp - 2
big - 1
curso - 1
```

Python

```
lista = p_ordena.collect()
for row in lista:
    print(row[0],"-",row[1])
hadoop - 4
semantix - 3
data - 3
sp - 2
2019 - 2
big - 1
curso - 1
```

RDD – Salvar dados

```
p_ordena.getNumPatitions
3

p_ordena.saveAsTextFile("saida")
```

```
$ hdfs dfs -ls /user/root/saida
part-00000  part-00001  part-00002  _SUCCESS
```

Exercícios RDD

1. Ler com RDD os arquivos localmente do diretório “/opt/spark/logs/” ("file:///opt/spark/logs/")
2. Com uso de RDD faça as seguintes operações
   a) Contar a quantidade de linhas
   b) Visualizar a primeira linha
   c) Visualizar todas as linhas
   d) Contar a quantidade de palavras
   e) Converter todas as palavras em minúsculas
   f) Remover as palavras de tamanho menor que 2
   g) Atribuir o valor de 1 para cada palavra
   h) Contar as palavras com o mesmo nome
   i) Visualizar em ordem alfabética
   j) Visualizar em ordem decrescente a quantidade de palavras
   k) Remover as palavras, com a quantidade de palavras > 1
   l) Salvar o RDD no diretorio do HDFS /user/<seu-nome>/logs_count_word

Partições

- Introdução

Semantix®
All about data

Partições

- Spark armazena os dados do RDD em diferentes partições
- Mínimo de partições é 2
- Definir partições manualmente na leitura e redução do dado

```
rdd = sc.textFile("entrada*“, 6)
palavras = rdd.flatMap(lambda linha: linha.split(" "), 3) #ERRO
p_chave_valor = palavras.map (lambda palavra: (palavra,1), 20) #ERRRO
p_reduce = p_chave_valor.reduceByKey(lambda key1, key2: key1 + key2,10)
p_reduce5 = p_reduce.repartition(5)
pReduce.getNumPartitions()
```

Exercícios RDD com Partições

1. Ler com RDD os arquivos localmente do diretório “/opt/spark/logs/” ("file:///opt/spark/logs/") com 10 partições
2. Contar a quantidade de cada palavras em ordem decrescente do RDD em 5 partições
3. Salvar o RDD no diretorio do HDFS /user/<seu-nome>/logs_count_word_5
4. Refazer a questão 2, com todas as funções na mesma linha de um RDD

Semantix®

All about data

contato@semantix.com.br

www.semantix.com.br