Desafio Engenheiro de Dados

Thiago Caselli dos Santos

**Qual o objetivo do comando cache​ ​em Spark?**

R: O comando cache é um mecanismo para acelerar aplicações que acessam o mesmo RDD diversas vezes. As informações serão armazenadas em memória evitando a necessidade de reprocessamento dos dados e diminuindo o trabalho dos executores.

**O mesmo código implementado em Spark é normalmente mais rápido que a implementação equivalente em MapReduce. Por quê?**

R: O código implementado em Spark torna-se mais rápido que o MapReduce pelo fato dele trabalhar com os dados em memória utilizando RDDs (Resilient Distributed Datasets), onde os executores realizam o processamento dos dados. Já os algoritmos do MapReduce funcionam com os dados de leitura e escrita em disco, tornando o processo mais lento.

**Qual é a função do SparkContext​?**

R: O SparkContext é o ponto de entrada principal para acessar as funcionalidades do Spark, e permite estabelecer conexão com o cluster de Spark. O SparkContext pode ser usado para criar RDDs, por exemplo.

**Explique com suas palavras o que é Resilient​ ​Distributed​ ​Datasets​ (RDD).**

R: O RDD é um Dataset que permite trabalhar o processamento dos dados de forma distribuída dentro de um cluster, onde o driver controlará o trabalho dos executores em diferentes nós, diminuido e otimizando o tempo de processamento. O RDD é a principal estrutura de dados do Spark.

**GroupByKey​ ​é menos eficiente que reduceByKey​ ​em grandes dataset. Por quê?**

R: Quando utiliza-se o reduceByKey, os valores com a mesma chave são combinados antes de serem distribuídos pelo cluster, isto otimiza de forma significativa o processamento. Já o groupByKey realiza a distribuição dos dados antes de combinar os valores com a mesma chave, podendo causar problemas na hora de agrupá-los quando trata-se de grandes datasets.

**Explique o que o código Scala abaixo faz.**

**val textFile = sc.textFile("hdfs://...")**

**val counts = textFile.flatMap(line => line.split(" "))**

**.map(word => (word, 1))**

**.reduceByKey(\_ + \_)**

**counts.saveAsTextFile("hdfs://...")**

**val textFile = sc.textFile("hdfs://...")**

Esta etapa utiliza o SparkContext para ler um arquivo armazenado dentro do HDFS e criar um RDD chamado “textFile” que armazenará os dados.

**val counts = textFile.flatMap(line => line.split(" "))**

a variável imutável “counts” recebe a transformação do “textFile” que realizará a separação de linhas entre os dados que contém espaço em branco, por exemplo:

valor1 valor 2 valor 3

Operação Split:

valor1

valor2

valor3

**.map(word => (word, 1))**

O map atribuirá um conjunto chave valor para a variável “counts”, onde a chave será a linha atual do dataset, e o valor atriuído à esta chave será “1”.

**.reduceByKey(\_ + \_)**

O reduceByKey realizará a soma de todos os valores que contém a mesma chave, por exemplo:

(valor1, 1)

(valor1, 1)

(valor1, 1)

(valor2, 1)

(valor2, 1)

(valor3, 1)

Transformação reduceByKey:

(valor1, 3)

(valor2, 2)

(valor3, 1)

**counts.saveAsTextFile("hdfs://...")**

Salva o resultado da variável “counts” em um diretório especificado do HDFS.