Qual o objetivo do comando cache em Spark?

Objetivo do cache do Spark é alocar objeto de pesquisa (RDD) na mémoria “ram” para melhorar as pesquisas. Na realidade ele somente aponta esse alocamento de mémoria, pois o mesmo só ocorre quando uma consulta sobre o objeto for efetuada porque se trata de uma operação Lazy

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

No Spark os objetos consultados podem ser consultados em disco, manipulados e depois criados em memória “ram”. Após isso persistidos novamente em disco.
No MapReduce todas as consultas e manipulações de objetos é feita diretamente no disco

Comparando os mesmos, como há economia de IO’s fazendo parte de seu ciclo o armazenamento em memória o Spark se torna mais rápido

Qual é a função do SparkContext ?

Similarmente feitas em outras linguagens, o Context são classes encapsuladas de conexões entre os objetos da linguagem e o banco de dados relacional.

No caso do Spark, esta é a função do SparkContext, que encapsula essa relação entre as fontes de dados (SqlContext ou HiveContext), podendo criar consultas SQL no pySpark.

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

É similar ao dataSet no C#, onde os dados de consulta de tabelas estruturadas são armazenadas em memória

No Spark o RDD é subdividido em algumas partições, para procesamentos paralelos multiplos. Este dados podem ser modificados o que ocasiona a criação de um ouro RDD.

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

O groupByKey é como um group by do SQL, no qual agrupa uma lista de valores em relação a uma chave, isso causa um aumento de consumo de memória "ram". Isso acontece porque todo o dadaset é percorrido para armazenar os grupos. Já no reduceByKey, o dataset original é reduzido a um dataset que contenha apenas os dados que contenham a chave de redução. Isto diminui o consumo de memória "ram", pois o processo é dividido em duas partes. Primeiro reduzimos e depois escolhemos a key correta.

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://..." )
```

variavel "textFile", onde se lê o arquivo indicado no parametro do textFile().

```
val counts = textFile . flatMap ( line => line . split ( " " ))
. map ( word => ( word , 1 ))
. reduceByKey ( _ + _ )
```

variavel "counts". FlatMap para splitarmos as palavras de cada linha separadas por " "; em seguida usamos a função map para criar uma lista de string e inteiros, onde depois usamos reduceByKey para somarmos o numeros de vezes que esta palavra aparece no texto selecionado.

Como resultado temos uma "lista" do tipo chave/valor (string,int);

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

salvamos nossos dados no caminho indicado;

Codificação no arquivo - DesafioParteCodificacao.py