1. Qual objetivo do comando cache em Spark?

R: O Spark utiliza um conceito da computação chamado Lazy Evaluation, o que significa que existem dois tipos de operações: Transformation e Action. Ao utilizar uma transformation, como .map ou .filter, elas não são executadas imediatamente, mas sim mapeadas em uma pilha de instruções. Ao utilizar uma action, como .show ou .collect, aí sim as transformations são executadas. O comando cache é responsável por armazenar os resultados das transformações em memória quando a primeira action é disparada, isso resulta em um aumento de perfomance para as actions seguintes. Caso você não utilize o comando cache e utilizar várias actions para o mesmo RDD ou DataFrame, em todos esses momentos todas as transformações deveram ser computadas novamente. Ao utilizar o comando cache, o resultado já estará em memória, o que aumentará drasticamente a performance do seu job. Importante citar que existem outras formas de persistir os resultados das transformações com o método .persist, porém o cache armazena apenas em memória.

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

R: O código implementado em Spark é mais performático que o método utilizando MapReduce (até 100x mais performático segundo os desenvolvedores), uma vez que o seu processamento é realizado in-memory.

3. Qual é a função do SparkContext?

R: Toda aplicação Spark consiste em um *driver program* que dispara várias operações em um cluster. Os *driver programs* acessam o Spark através de um objeto SparkContext, que representa basicamente uma conexão com o cluster. A partir do objeto do SparkContext, pode-se criar RDDs.

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

R: RDD são basicamente collections de objetos distribuídos. Eles podem possuem dois tipos de operações, sendo elas as transformations e as actions, como descrito na explicação do Lazy Evaluation da primeira pergunta. RDDs, ao contrário dos DataFrames, trabalham bem com dados heterogêneos, uma vez que não precisam de um schema para serem utilizados. Importante frisar que os RDDs não armazenam os dados propriamente ditos, mas sim as operações que são aplicadas aos dados. Eles podem ser criados a partir da leitura de dados externos, mas também podem ser criados a partir de uma collection de objetos, como listas.

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

R: GroupByKey e reduceByKey são operações que vão retornar o mesmo resultado, a diferença entre ambas as operações é que o método reduceByKey, os pares em cada nó do cluster com a mesma key são combinados antes do shuffle, diminuindo bastante a quantidade de dados ao

aplicar a operação. Já o GroupByKey aplica o shuffle primeiro e então aplica o processamento em todos os registros de todos as partições.

6. Explique o que o código Scala abaixo faz.

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

Cria um RDD lendo arquivo(s) do HDFS (file system distribuído do Hadoop)

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

O método flatMap cria um novo RDD com todas as palavras separadas

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

Enquanto o map aplicado a ele cria num novo RDD com uma tuple contendo a palavra e o número 1

```
.reduceByKey(_ + _)
```

O método reduceByKey está agrupando as palavras e somando os números, resultando assim em um tuple contendo a palavra e o número de vezes que ela apareceu no(s) arquivo(s)

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

O método saveAsTextFile é uma action que vai disparar todas as transformations anteriores e irá escrever o resultado em um arquivo no HDFS.