1 – Qual o objetivo do comando cache em Spark ?
Ao utilizar o comando “cache” em spark, torna-se possível trabalhar com conjuntos de dados em memória. Isso torna as ações realizadas em tais datasets mais rápidas e eficientes. É muito útil quando há a necessidade de manipularmos um conjunto de dados repetidamente.

2 – O mesmo código implementado em Spark é normalmente mais rápido que a implementação equivalente em MapReduce. Por quê ?
Spark é uma ferramenta que permite o processamento de dados em alto volume. Um dos seus diferenciais em relação ao MapReduce é a capacidade do Spark de executar ações nos datasets em memória, isso permite uma maior eficiência ao manipular um mesmo conjunto de dado mais de uma vez.
Em um cenário de cluster, a execução do MapReduce se dá por meio da leitura de um dado no cluster,e então a execução de um algoritmo, após isso há o retorno do resultado de volta ao cluster. Com spark temos as mesmas sequências executando somente em um lugar, isso torna o processo muito mais rápido.

3 – Qual é a função do SparkContext ?
O Spark Context é um objeto que permite a integração do spark ao programa em desenvolvimento. Podemos atribuir tal objeto à uma variável e usufruir das funcionalidades presentes em tal framework.

4 – Explique com suas palavras o que é Resillent Distributed Datasets (RDD).
O RDD é a representação do Spark a respeito de um conjunto de dados. É um objeto fundamental no desenvolvimento com spark. Nele poderá ser armazenado qualquer tipo de dado, e será possível executar transformações e ações nele. Uma de suas características é ser imutável. O spark armazena tais objetos em partições, e em um cenário de cluster, tais partições são replicadas e divididas no ambiente.

5 – GroupByKey é menos eficiente que reduceByKey em grandes dataset. Por quê ?
GroupByKey é menos eficiente que reduceByKey em grandes dataset devido ao processo que tal método executa para alcançar seu resultado.
No reduceByKey temos os dados combinados a partir de suas chaves, resultando em uma redução de dados que estão acumulados em sua respectiva key, e só após o processo de redução, temos o shuffle(Processo responsável por ordenação das tuplas).
No groupByKey temos os dados inicialmente mapeados em chave e valor, após isso temos um processo de shuffle, que ordenará todas as tuplas com key semelhantes(Ou seja, se temos um grande dataset, tal processo será custoso), e então o agrupamento da key com seus respectivos values dados em uma lista. Esse processo é muito mais custoso, tanto em desperdício quanto em eficiência, já que poderá haver alguns dados que não serão utilizados.

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://...")
```

Explicação:

No início do algoritmo, há a leitura de um arquivo presente no hdfs(sistema de arquivos distribuídos do hadoop). Tal conteúdo do arquivo será armazenado em uma variável declarada como **textFile,** e será um objeto RDD**.**

Após isso, é atribuído à uma variável chamada **counts** uma série de transformações realizadas no objeto rdd **textFile**, tais transformações serão esclarecidas a seguir:

Inicialmente temos a execução do método **flatMap**. Ele será responsável por percorrer cada linha pertencente ao conteúdo do arquivo armazenado no objeto RDD. Em cada linha, será realizado o método split, que irá separar o conteúdo dessa linha, aglomerando-os em um único array. Diferente do método **map**, todo conteúdo que sofre transformações no **flatMap** é reduzido a um array.

Após a realização do flatMap, há a execução do método **map.** Tal método, percorrerá cada elemento da lista que está sendo manipulada, e irá retornar, em lugar do elemento percorrido, uma tupla com **chave** e **valor.** Sendo a chave, cada palavra percorrida no array, e o valor, inicialmente para cada chave dados como **1.**

E então, ao final da transformação, haverá uma outra etapa realizada, chamada de **reduceByKey,** que irá agrupar todas as **chaves semelhantes** (criadas no processo de map) e somará os valores, formando, para cada chave, uma única tupla de **chave** e **valor,** onde agora cada **chave** será única, e seus valores serão correspondentes a **soma** dos valores de todas as chaves semelhantes.

No final, o novo RDD formado será armazenado no hdfs.

Questões práticas

1. **Número de hosts únicos:** 137.978

2. O total de erros 404: 20.895

3. Os 5 URLs que mais causaram erro 404:

- hoohoo.ncsa.uiuc.edu
- piweba3y.prodigy.com
- jbiagioni.npt.nuwc.navy.mil
- piweba1y.prodigy.com
- www-d4.proxy.aol.com

4. Quantidade de erros por dia: Aproximadamente **360** erros por dia.

5. O total de bytes retornado: 6.5524314098E10
Obs: (VALOR DADO EM NOTAÇÃO CIENTÍFICA, POR SER MUITO EXTENSO)