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

Armazenar um RDD na memória dos nodes para que essa informação seja utilizada mais tarde de forma mais rápida porque, já estará processada. Sendo uma boa alternativa para itens que são muito utilizados no decorrer do código. O cache e tolerante a falha, se uma partição do RDD for perdida ele será automaticamente reprocessado com base na transformação que originou ele.

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

No Spark os dados são distribuídos em memória a qual possui um throughput muito maior que o disco.

Por exemplo um cenário onde queremos tratar informações que estão no HDFS. O processo de MapReduce faz o mapeamento dos dados que estão em disco, aloca na área de stage que também é armazenada em disco para depois realizar o reduce e persiste o resultado novamente no disco. Já o Spark coleta os dados armazena em memória processa e persiste novamente em memória. Ele também pode ser configurado para se o volume de dados for maior que a memória disponível no cluster ele armazena parte da informação em disco.

**Qual é a função do SparkContext​?**

Ser o ponto de entrada para Spark API. Ele configura os serviços internos e permite acesso as funcionalidades do Spark.

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

RDDs são a unidade principal do Spark. É uma coleção de objetos distribuídos pelo cluster. E são imutáveis para ser tolerante a falhas, caso ocorra um problema ele pode ser recriado com base nos RDDs anteriores tornando segura a manipulação de grandes datasets.

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

Porque quando utilizamos o reduceByKey o Spark entende que pode executar a operação em cada partição utilizando a key como referência, só após a conclusão da operação é feito o shuffle. Como os dados já estão consolidados são menos itens para serem separandos. Entretanto o groupByKey faz antes o shuffle deixando todos os itens que possuem a mesma key juntos para depois realizar a operação.

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

Conta quantas vezes cada palavra apareceu no dataset.

val textFile = sc.textFile("hdfs://...") // Cria um RDD a partir de um arquivo armazenado no HDFS

val counts = textFile.flatMap(line => line.split(" ")) //Utilizando o espaço com delimitador é criado um coleção de itens

.map(word => (word, 1)) // É criado um mapeamento chave valor onde a palavra é a chave e o número 1 é um valor que será utilizado em outro passo

.reduceByKey(\_ + \_) // Nesse momento é iniciada a ação, é realizada a soma dos valores por chave.

counts.saveAsTextFile("hdfs://...") // O resultado é armazenado no HDFS