Respostas:

1. ***Qual o objetivo do comando cache em Spark?***

O Spark fornece os próprios mecanismos de cache nativo que podem ser utilizados através de diferentes métodos, como .persist(), .cache() e CACHE TABLE. Esse cache nativo é efetivo com pequenos conjuntos de dados, bem como em pipelines ETL, onde é necessário armazenar em cache resultados intermediários. No entanto, o cache nativo do Spark atualmente não funciona bem com o particionamento, uma vez que uma tabela armazenada em cache não mantém os dados de particionamento. Uma técnica de cache mais genérica e confiável é o cache de camada de armazenamento.

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

Sim. Devido a utilização de manipulação de dados em memória ser mais eficiente e mais rápida que dividir os dados gravados em disco. A perda de eficiência é devido ao tempo de gravação em disco para o MapReduce.

1. ***Qual é a função do SparkContext?***

Um SparkContext é um cliente do ambiente de execução do Spark e atua como o mestre do aplicativo Spark. O SparkContext configura serviços internos e estabelece uma conexão com um ambiente de execução do Spark.

O SparkContext permite que o aplicativo do driver Spark acesse o cluster por meio de um gerenciador de recursos.

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

O RDD no Apache Spark é uma coleção imutável de objetos que calcula os vários nós do cluster. Resiliente, ou seja, tolerante a falhas com a ajuda do RDD DAG e, portanto, pode recalcular partições ausentes ou danificadas devido a falhas de nó no Spark. Distribuído, pois os Dados residem em vários nós.

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

Enquanto reducebykey e groupbykey produzirão a mesma resposta, o exemplo de reducebykey funciona muito melhor em um grande conjunto de dados. Isso ocorre porque o Spark sabe que pode combinar a saída com uma chave comum em cada partição antes de embaralhar os dados.

Por outro lado, ao chamar groupByKey - todos os pares de valores-chave são embaralhados. São muitos dados indesejáveis para serem transferidos pela rede.

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

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

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

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

.reduceByKey(\_ + \_)

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

Para cada linha obtida, é segmentada por palavras separadas por " ".O resultado da contagem de palvras é gravada em um arquivo texto.