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

Reposta. Mantem os comandos de transformação em cache, o que por um lado agiliza o processamento quando necessitado novamente, mas que deve ser utilizado com cuidado porque o Spark utiliza memória para processamento.

Fonte: Curso. Processamento de dados com Spark e PySpark na plataforma Udemy (Felipe Mafra)

Fonte: <https://www.infoq.com/br/articles/apache-spark-introduction/>

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

Reposta. O Apache Spark utiliza em cache via memória enquanto MapReduce Hadoop utiliza disco. O processamento em memória é cerca de 100 vezes mais rápido. Mesmo quanto o Apache Spark utiliza disco (quando o uso da memória já está completo) ele otimiza o uso do disco sendo ainda 10 vezes mais rápido.

Além disso, no Apache Spark temos bibliotecas nativas.

Fonte: Curso. Processamento de dados com Spark e PySpark na plataforma Udemy (Felipe Mafra)

Qual é a função do **SparkContext** ?

Enviar o código para cada uma das máquinas do Cluster (Worker) executar. Ele é gerado pelo Driver.

O SparkContext pode ser nomeado, sendo um contexto por Node.

Fonte: Curso. Processamento de dados com Spark e PySpark na plataforma Udemy (Felipe Mafra)

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

De forma bem simples é o conjunto de dados particionado para processamento paralelo. Um ponto importante é que os RDDs se recuperam automaticamente de eventuais falhas em um Worker, possibilitando reprocessar somente o ‘pedaço’ que falhou em um outro worker.

Fonte: <https://spark.apache.org/docs/latest/rdd-programming-guide.html>

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

Ambos trabalham com chave valor, porém o GroupByKey forma uma chave com uma lista de valores na rede. Já o reduceByKey primeiro reduz agregando os valores para a mesma chave (só precisa que tenham o mesmo tipo de valor). Isso faz que o reduceByKey trafegue um volume de dados menor na rede, o que o torna mais eficiente.

Fonte: <https://www.linkedin.com/pulse/groupbykey-vs-reducebykey-neeraj-sen>

Código Scala

Este código dado um texto separa as palavras por espaço (“ “) e em seguida conta as palavras reduzino a um par chave e valor (a palavra e quantas vezes aparece), salvando em seguinda em um arquivo.

Fonte: <https://spark.apache.org/examples.html>

Quanto aos exercícios práticos em Spark, como eu ainda não domino, eu carreguei as informações das bases fornecidas em Tableau e gerei gráficos para responder.

O Dashboard resultante está disponível no Tableau Public

<https://public.tableau.com/profile/warner.de.souza.silva7838#!/vizhome/ExercicioNASA/Dashboard1>

As respostas as questões abaixo estão no Dashboard publicado no link acima

1. Número de hosts únicos.

2. O total de erros 404.

3. Os 5 URLs que mais causaram erro 404.

4. Quantidade de erros 404 por dia.

5. O total de bytes retornados.