Gostaríamos de fazer um teste que será usado para sabermos a sua proficiência nas habilidades para a vaga. O teste consiste em algumas perguntas e exercícios práticos sobre Spark e as respostas e códigos implementados devem ser armazenados no GitHub. O link do seu repositório deve ser compartilhado conosco ao final do teste.

Quando usar alguma referência ou biblioteca externa, informe no arquivo README do seu projeto. Se tiver alguma dúvida, use o bom senso e se precisar deixe isso registrado na documentação do projeto.

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

Persiste todos os dados em memória (MEMORY\_ONLY).

O mesmo código implementado em Spark é normalmente mais rápido que a implementação equivalente em

MapReduce. Por quê?

É mais rapido por conta do processamento, pois o Spark pode fazer isso na memória, enquanto o MapReduce precisa ler e gravar em um disco.

Qual é a função do **SparkContext**?

É o que conecta o Spark ao programa que está sendo desenvolvido.

Ele pode ser acessado como uma variável em um programa para utilizar os seus recursos.

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

É uma estrutura de dados fundamental do Spark. E uma coleção distribuída imutável de objetos. Cada conjunto de dados no RDD é dividido em partições lógicas, que podem ser calculadas em nós diferentes do cluster.

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

O GroupByKey quando utilizado, os dados são embaralhados e nessa transformação, muitos dados desnecessários são transferidos pela rede e o ReduceByKey antes de embaralhar os dados, os pares na mesma máquina com a mesma chave são combinados, com isso se torna mais eficiente.

val textFile = sc.textFile("hdfs://..."): Lê um arquivo texto dentro do HDFS e grava dentro de uma variável textFile.

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

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

.reduceByKey(\_ + \_): Ele vai splitar por vazio e após o Split vai fazer a soma de cada palavra e jogar dentro da variável counts.

counts.saveAsTextFile("hdfs://..."): Vai salver o resultado do counts dentro do HDFS.

**HTTP requests to the NASA Kennedy Space Center WWW server**

**Fonte oficial do dateset**: <http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html>

**Dados**:

● Jul 01 to Jul 31, ASCII format, 20.7 MB gzip compressed, 205.2 MB.

● Aug 04 to Aug 31, ASCII format, 21.8 MB gzip compressed, 167.8 MB.

**Sobre o dataset**: Esses dois conjuntos de dados possuem todas as requisições HTTP para o servidor da NASA Kennedy

Space Center WWW na Flórida para um período específico.

Os logs estão em arquivos ASCII com uma linha por requisição com as seguintes colunas:

● **Host fazendo a requisição**. Um hostname quando possível, caso contrário o endereço de internet se o nome não puder ser identificado.

● **Timestamp** no formato "DIA/MÊS/ANO:HH:MM:SS TIMEZONE"

● **Requisição (entre aspas)**

● **Código do retorno HTTP**

● **Total de bytes retornados**

**Questões**

Responda as seguintes questões devem ser desenvolvidas em Spark utilizando a sua linguagem de preferência.

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.