Prezado candidato.

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?
É um mecanismo para acelerar aplicativos que acessam o mesmo RDD várias vezes, caso não seja armazenado em cache, nem com checkpoint, é reavaliado e executado novamente toda vez que uma ação é invocada.

O mesmo código implementado em Spark é normalmente mais rápido que a implementação equivalente em MapReduce. Por quê?
Sim, o processamento em spark roda em memoria.

Qual é a função do **SparkContext**?
O SparkContext é o ponto de entrada para qualquer funcionalidade de ignição. Quando executamos qualquer aplicativo Spark, um programa de driver que tem a função principal com seu SparkContext. O programa do driver, em seguida, executa as operações dentro dos executores nós do trabalhador.

Explique com suas palavras o que é **Resilient Distributed Datasets** (RDD).
Conceito de um conjunto de dados resiliente distribuído (RDD), que é uma coleção tolerante a falhas de elementos que podem ser operados em paralelo.

GroupByKey é menos eficiente que **reduceByKey** em grandes dataset. Por quê?
reduceByKey e groupByKey produzirão a mesma resposta, O reduceByKey funciona muito melhor em um grande conjunto de dados, isso 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. Este é um monte de dados desnecessários para ser transferido através da rede.

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://...")
```

Obterm um arquivo dentro do hdfs e faz o "contador de palavras", faz a saida em um arquivo texto.

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.