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?

Guardar resultados na memória através do cache. Assim, o resultado de alguma ação pode persistir na memória tornando mais rápidos acessos futuros aos dados.

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

A Solução de MapReduce, foi uma das primeiras ideias de processamento de Big Data, se apóia basicamente no acesso aos dados e os reserva em disco. Ao passar do tempo, esta abordagem apresentou um custo superior de tempo/velocidade. O Apache Spark surgiu com uma possibilidade que poderia corrigir este tempo extra de acesso, a persistência de dados na memória.
Como o acesso a dados da memória é muito mais rápido que o mesmo acesso em disco, o que possibilita códigos implementados em Spark serem executados em velocidade muito superior a códigos equivalentes para MapReduce.

Qual é a função do **SparkContext**?

É a primeira configuração a ser feita pelo código de uma aplicação Spark
é a conexão entre o Driver Program e o Cluster Manager. Esta conexão é feita a partir da criação do objeto SparkContext, onde o desenvolvedor deve passar as informações do nome da aplicação Spark, que será apresentada na interface do cluster, e o Cluster Manager a ser utilizado.

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

É uma abstração para uma forma de processamento paralelo em nós de um cluster, é uma sequência de passos de execução, que são as operações de Transformação e um conjunto de Ações que podem ser executadas em paralelo para formar uma coleção de dados que voltará ao servidor principal. Então o RDD acaba retornando uma coleção de elementos que estão divididos através dos nós do cluster.

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

O GroupByKey não realiza nenhum agrupamento de informações já no resultado de uma tarefa dentro de um nó do cluster, o reduceByKey promove um primeiro
agrupamento dentro do contexto de cada nó para ao final realizar uma única operação
de união das informações coletadas de cada nó. Dessa forma, o groupByKey acaba trafegando
muito mais dados de cada uma das pontas para apenas no final realizar um agrupamento.

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://...")
```

O método textFile está lendo todas as linhas de um arquivo de entrada no formato HDFS (Hadoop Distributed File System) e criando um RDD com estas linhas.
Em seguida, o programa gera uma coleção única com todos as "palavras" do arquivo, ou seja, cada um dos termos separados por espaço no arquivo de entrada reunidos em uma única lista.
Então, cada um dos elementos dessa lista ganha um contador iniciado em 1, ou seja, um contador de ocorrências de cada palavra.
O comando reduceByKey então irá reduzir todas as ocorrências de palavras repetidas a um único elemento com contador refletindo esse número de ocorrências.
Por fim o resultado do contador é gravado em um 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.