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?
O comando cache tem o objetivo de guardar resultados parciais em memória para usá-los nos próximos estágios, para acelerar o processamento.

O mesmo código implementado em Spark é normalmente mais rápido que a implementação equivalente em MapReduce. Por quê?
Porque spark faz o processamento na memória dos worker nodes, sem escrever em disco a todo momento como o map reduce.

Qual é a função do **SparkContext**?
A função do sparkContext é atuar como master de uma aplicação spark e permitir que a aplicação acesse o cluster spark por meio do Resource Manager.

Explique com suas palavras o que é **Resilient Distributed Datasets** (RDD).
RDDs sao os dados que trabalhamos em spark. Esses dados (Datasets) estao distribuídos em múltiplas partições , distribuídas em múltiplos nós (Distributed). Resilient significa que se um nós falhar, os dados daquele nó podem ser recuperados.

GroupByKey é menos eficiente que **reduceByKey** em grandes dataset. Por quê?
GroupByKey pode trazer problemas de performance em dados muito grandes. Após o groupByKey, todos os dados com a mesma chave serão armazenados no mesmo node do cluster, e este node, por muitas vezes não suportar todos os dados, vai tentar guardar grande parte desses dados em memória, o que pode causar out of memory exceptions mesmo que o cluster tenha varios nós.

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://...")
```

É um exemplo simples de word count, lendo de um arquivo no hdfs, realizando map e reduce para contar as palavras desse arquivo e salvando o resultado de volta no 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.