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 comando para armazenar o resultado obtido em determinado RDD/Dataframe na memória do Cluster. Como motivo para tal execução, posso citar dois: reuso do dado em etapas posteriores do script; e quando temos um objeto que teve intenso processamento e queremos salvar o resultado, para que em caso de falha nas etapas posteriores, a etapa com custo alto de processamento já esteja calculada. Porém, temos que tomar cuidado com o quanto que o objeto vai consumir de memória.

O mesmo código implementado em Spark é normalmente mais rápido que a implementação equivalente em MapReduce. Por quê?
Por dois principais motivos: o Spark trabalha com o processamento em memória (mas também consegue trabalhar com memória/disco ou somente disco), o que aumenta a velocidade do processamento. O mapReduce utiliza sempre o disco para armazenar o resultado do reduce, e isso faz com que o I/O seja frequente. Outro ponto é que o Spark possui um DAG altamente otimizado, criando planos de execução mais eficientes, que vão gerar resultados de forma mais rápida.

Qual é a função do **SparkContext**?
SparkContext abre uma conexão, composta de algumas configurações, para gerar um contexto de aplicação entre o Cluster Spark e seu programa. Abre uma porta para uso das funcionalidades existentes do framework. No Spark 2, esse contexto já é inicializado com a criação da Sessão. Todas as sessões utilizam do mesmo contexto.

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

RDDs são os objetos base do Spark, que utilizam do conceito de cluster para serem armazenados em memória de forma distribuída, e que por isso, podem ser processados paralelamente. Também não aceitam alterações apenas geração de novos RDDs. Não requer que o dado seja estruturado ou semi, tal como Dataframe, nem precisa que lhe diga exatamente seu schema.

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

Porque com o GroupByKey os pares de chave-valor, logo após a fase de map, são transferidos entre os nós do cluster, para a fase de shuffle. Isso faz com que um grande volume de dados trafegue de um lado para o outro. Já o reduceByKey é mais inteligente, pois ele agrega as chaves antes de fazer o shuffling, o que diminui o volume de dados trafegados na rede.

Este documento é confidencial e não pode ser distribuído, copiado em parte ou na sua totalidade

Semantix

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://...")
```

Cria o primeiro objeto RDD, que lê determinado arquivo contido no HDFS.

A partir desse objeto, é criado um segundo objeto, que contém algumas etapas:

1. Separa cada elemento a cada espaço encontrado. O flatmap faz com que um elemento possa gerar mais de uma saída, diferente do map.
2. A segunda etapa, cria o mapeamento de chave-valor para cada elemento (palavra) da primeira transformação. Chave = palavra, valor = 1.
3. Após o mapeamento, há a redução das Keys dentro de cada nó, ou seja, há uma somatória intermediária. Ainda com esse mesmo comando irá acontecer o shuffle para aí sim a redução final para cada Key, somando as quantidades intermediárias já calculadas de cada key, em cada nó do cluster.

A última linha é a ação que irá de fato desencadear a execução das atividades anteriores, pois é um comando de ação. Vai gravar o resultado da contagem de palavras 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.

***Houve pouco mais de 1700 registros que fugiram do padrão procurado, na sua grande maioria por conta da URL. Como o número é baixo em relação ao total e não houve um pedido explícito para tratar essas exceções, apenas criei um rdd para armazenas esses registros, para alguma atuação no futuro.**

1. Número de hosts únicos. **`137951 de hosts`**
2. O total de erros 404. **`20803 ocorrências`**
3. Os 5 URLs que mais causaram erro 404.

```
+-----------------------------------------------------------------+----+
|url                                                              |qtde|
+-----------------------------------------------------------------+----+
|/pub/winvn/readme.txt                                            |2004|
|/pub/winvn/release.txt                                           |1732|
|/shuttle/missions/STS-69/mission-STS-69.html                     |683 |
|/shuttle/missions/sts-68/ksc-upclose.gif                         |428 |
|/history/apollo/a-001/a-001-patch-small.gif                      |384 |
+-----------------------------------------------------------------+----+
```

4. Quantidade de erros 404 por dia.

```
+--------------------------+----+
|date                      |qtde|
+--------------------------+----+
|1995-07-01 00:00:00|315 |
|1995-07-02 00:00:00|289 |
|1995-07-03 00:00:00|474 |
|1995-07-04 00:00:00|357 |
|1995-07-05 00:00:00|495 |
|1995-07-06 00:00:00|636 |
|1995-07-07 00:00:00|566 |
...
...
```

5. O total de bytes retornados. **65.519.970.468 bytes**

Este documento é confidencial e não pode ser distribuído, copiado em parte ou na sua totalidade