

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 a tecnica de otimizar a execução de Jobs, guardando resultados parciais em memória sem a necessidade de excrita em disco. |



|  |
| --- |
| O​ ​mesmo​ ​código​ ​implementado​ ​em​ ​Spark​ ​é​ ​normalmente​ ​mais​ ​rápido​ ​que​ ​a​ ​implementação​ ​equivalente​ ​em MapReduce.​ ​Por​ ​quê? |
| Com o MapReduce por exemplo quando o job2 necessita de dados do job1 este tem que gravar o resultado em disco para que o job2 venha e leia esses dados em disco tornando assim o processamento um pouco mais demorado do que em Spark que permite fazer esse mesmo processo passando diretamente em memoria(cache) os dados do Job1 para o Job2 tornando assim o processo mais rápido pois não precisa de escrita em disco. Outro ponto que o torna mais rápido que o MapReduce é porque o Spark mantém a JVM constantemente em execução em cada nó já o MapReduce inicia uma nova instancia JVM toda vez que iniciado. |

|  |
| --- |
| Qual​ ​é​ ​a​ ​função​ ​do​ **SparkContext**​? |
| Funciona como um tipo de “parametro” do ambiente Spark em execução, ele serve para alocação de recursos (memória, CPU’s entre outros), ele também serve para criar RDD’s, colocar Jobs em execução criar váriaveis de broadcast e acumuladores. |

|  |
| --- |
| Explique​ ​com​ ​suas​ ​palavras​ ​o​ ​que​ ​é​ **Resilient**​​**Distributed**​​**Datasets**​ ​(RDD). |
| É um dos conceitos proncipais do Framework SPARK, é o conjunto de dados resilientes e distribuídos. É como “tabelas de banco de dados” em memoria, essas “tabelas” são armazenadas em diferentes partições para melhor otimização no processamento dos dados. Os RDD’s são imutaveis, e quando há alguma transformação no dados é criado um novo RDD ou seja o original permanece inalterado. |

|  |
| --- |
| **GroupByKey**​é​ ​menos​ ​eficiente​ ​que​ **reduceByKey**​em​ ​grandes​ ​dataset.​ ​Por​ ​quê? |
| reduceByKey aplica uma logica de redução de dados através das Keys para obter um resultado mais “enxuto” antes de passar os mesmos para os executores calcular o resultado final, resultando assim um volume menor de dados transferidos. Já o GroupByKey não tem essa logica de reduzir o volume de dados antes passando assim todos os dados para fazer a agregação dessa forma pode gerar um volume de informações desnecessárias e que seja superior ao tamanho de memoria disponível, fazendo assim a escrita em disco dos dados causando assim em uma performance mais baixa. |

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



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://..."​)

Primeira Linha: Le um arquivo que se encontra no HDFS e carrega em uma variável

Segunda Linha: cada linha é quebrada em sequência assim que encontrado o caracter espaço

Terceira Linha: pega essa linhas quebradas em sequência e faz o mapeamento de chave-valor

Quarta Linha: agrupa os valores de chave-valor somando os valores iguais e salva os valores agregados 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.

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