**Desafio Semantix – Engenheiro de Dados**

**Perguntas técnicas:**

**1) Qual o objetivo do comando cache em Spark?**

R: Armazenar os dados do RDD em memória, obtendo ganhos bem maiores de performance pois quando os dados forem reutilizados, o acesso será muito

mais rápido pois já estarão em cache. Recomendado utiliza com cautela, carregando dados menores, pois há limitação de recursos de memória.

Carregar os dados já tratados, ao invés de trazer os dados para serem agrupados no Spark.

Exemplo: meuRDD.cache()

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

R.: Porque o Spark leva o conceito do MapReduce ao próximo nível, conseguindo utilizar os recursos de memória e do cluster fazendo operações paralelas e distribuidas.

Já o MapReduce realiza suas operações em disco, degradando muito sua performance.

Vantagens do uso do Spark:

• API de alto nível:

desenvolvimento mais rápido, mais fácil

• Latência baixa:

processamento próximo de tempo real

• Armazenamento in-memory:

até 100x de melhoria de performance

**3) Qual é a função do SparkContext?**

R: O SparkContext configura serviços internos e estabele conexão com o ambiente do Spark.

Uma vez criado, você criar RDD, executar jobs, cancelar jobs, etc. Tem a função de conectar a aplicação ao cluster.

Por exemplo, utilizando o SparkContext.textFile será atribuído a um RDD (com o nome 'nome\_qualquer' o arquivo (/home/training/arq\_spark.log), por exemplo.

Cada linha do arquivo será um registro separado no RDD

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

R.: O significado de RDD é Resilient Distributed Dataset.

Resilient: Se o dados em memória são perdidos, eles podem ser recriados.

Distributed: É armazenado em memória através do cluster.

Dataset: Os dados podem vir de um arquivo ou pode ser criado por meio de um programa.

RDDs são a unidade fundamental do Spark. É uma coleção de objetos distribuidos e são imutáveis.

Há 2 tipos de operações em RDD:

1 - Transformação:

Operações que retornam um novo RDD, utilizando Lazy Evalution (Spark não irá executar até haver uma ação).

exemplo de função: map, filter, distinct, union, combineByKey(), groupByKey(), join, etc.

2 - Ação:

Uma ação que retorna um valor (resultado) para o driver ou escreve no storage.

A ação mais comum utiliza reduce que realiza operações envolvendo dois elementos do mesmo tipo e retorna um novo elemento do mesmo tipo.

Exemplos: Collect(), count(), top(num), reduce(func)

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

R.: Sempre que possível utilizar o reduceBykey, do que groupByKey.

O reduceByKey, primeiro agrupa por palavras em cada node e traz o somente o resultado. Tem pouco Shuffle pois tem pouco tráfego de rede.

Se utilizar o groupByKey, irá trazer todas as ocorrências da palavra de cada node do cluster distribuido, gerando muito Shuffle causando muito tráfego na rede, devida informações que irão trafegar pela rede distribuida.

**6) 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://...")**

val textFile = sc.textFile("hdfs://...")

**-> Carrega os dados e cria o RDD "textFile"**

val counts = textFile.flatMap(line => line.split(" "))

.map(word=>(word,1))

.reduceByKey(\_+\_)

-> **Separa as palavras e conta a quantidade de cada palavra, agrupando-as com suas respectivas quantidade no final**

-> flatMap + split = **Faz o split das linha em palavras,**

**map = cria a tupla (word,1). Função de "transformação"**

reduceByKey = sumariza a quantidade das palavras agrupando as quantidade em cada uma. Executa a função de "ação" no RDD

counts.saveAsTextFile("hdfs://...")

**-> Salva o resultado em um arquivo no HDFS. Função de "ação"**

**Questões Responda as seguintes questões devem ser desenvolvidas em Spark utilizando a sua linguagem de preferência.**

**Questões do Desafio Spark**

A resolução encontra-se no arquivo **Desafio\_Semantix - Engenheiro de Dados.ipynb**.

Seguem as respostas das questões:

**1. Número de hosts únicos.**

%sql

**select** count(**distinct**(host)) **as** host\_unicos

**from** tab\_final\_df

|  |
| --- |
| **137979** |
| **host\_unicos** |

**2. O total de erros 404.**

%sql

**select** count(\*) **as** qtd\_erro\_404

**from** tab\_final\_df

**where** return\_code = '404'

|  |
| --- |
| **20901** |
| **qtd\_erro\_404** |

**3. Os 5 URLs que mais causaram erro 404.**

%sql

**select** host **as** url, count(\*) **as** qtd

**from** tab\_final\_df

**where** return\_code = '404'

**group** **by** host

**order** **by** count(\*) **desc**

**limit** 5

**url qtd**

**hoohoo.ncsa.uiuc.edu 251**

**piweba3y.prodigy.com 157**

**jbiagioni.npt.nuwc.navy.mil 132**

**piweba1y.prodigy.com 114**

**www-d4.proxy.aol.com 91**

**4. Quantidade de erros 404 por dia.**

%sql

**select** substr(timestamp,1,2) **as** dia\_req, count(\*) **as** qtd\_dia

**from** tab\_final\_df

**where** return\_code = '404'

**group** **by** substr(timestamp,1,2)

**order** **by** dia\_req

**dia\_req qtd\_dia**

**01 559**

**02 291**

**03 778**

**04 705**

**05 733**

**06 1013**

**07 1107**

**08 693**

**09 627**

**10 713**

**11**

**.**

**.**

**.**

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

%sql

**select** sum(tab\_final\_df.\_4) **as** total\_bytes\_retornado

**from** tab\_final\_df

|  |
| --- |
| **65524314915** |
| **total\_bytes\_retornado** |