Qual o objetivo do comando cache em Spark?

O objetivo do comando cache é otimizar o acesso da aplicação à um mesmo RDD, dado que o RDD pode ser acessado diversas vezes e que pode ser realizada diversas ações no mesmo RDD, é interessante utilizar técnicas que não desperdicem o processamento feito anteriormente. Desta forma, o método cache cria um armazenamento em memória na JVM para acelerar a reutilização dos resultados persistidos.

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

A latência do MapReduce é alta devido a quantidade de dados intermediários que são gerados e escritos no disco na fase de Map e depois passados para a fase de Reduce.

Qual é a função do SparkContext?

O SparkContext é o ponto de partida para criar uma funcionalidade Spark, ele funciona como um cliente do ambiente Spark. Além de fazer a conexão com o Cluster ele carrega as configurações das propriedades do cluster permitindo que a aplicação Spark acesse o cluster de maneira encapsulada, seja usado para criar RDDs e acumuladores.

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

O RDD é uma coleção imutável de elementos distribuídos entre os nós de um cluster que podem ser processados em paralelo a partir de apenas um SparkContext. Os elementos dessa coleção são computados nas JVMs e isso ocorre a partir de dois tipos de operações sendo elas transformações e ações, gerando novos RDDs como resultado.

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

A principal diferença entre GroupByKey e reduceByKey é que a performance do GroupByKey é impactada pelo fato de ser uma operação de transformação. O GroupByKey retorna a chave e uma lista iterável dos valores correspondentes, sendo necessário aplicar uma função para obter o resultado final. Se a lista retornada for muito grande para ser armazenado em um nó pode resultar em out of memory Exception.

Explique o que o código Scala abaixo faz:

```
1  val textFile = sc.textFile("hdfs:// ")
2  val counts = textFile.flatMap(line=>line.split(" "))
3     .map(word=>(word,1))
4     .reduceByKey(_+_)
5  counts saveAsTextFile("hdfs://...")
```

Na linha um o arquivo de texto não estruturado é lido, cria se um RDD com seu conteúdo. Na segunda linha é aplicada a função split em cada linha do RDD, ela separa todas as palavras por espaços. E a função flatMap é aplicada, retornando um RDD onde cada palavra aparece em uma linha do RDD. A 3 linha cria um RDD de tuplas (palavra , 1), a 4 linha utiliza esse RDD para reduzir as chaves (palavras) e retornar um RDD onde nenhuma chave se repete. A última linha salva esse RDD de palavras contadas. *O código acima cria um arquivo de texto contendo a contagem de palavras do arquivo inicial da seguinte forma (palavra , qtd de vezes que essa palavra ocorre no arquivo)*.

Questões HTTP requests to the Nasa Kennedy Space Center

1. Número de hosts únicos.
   O total de hosts únicos para o dataset do mês de Julho foi 81983 e para o mês de Agosto foi de 75060. Total geral de hots únicos durante os dois meses 137979.

2. O total de erros 404.
   O total de erros 404 para o mês de Julho foi 10845 e para o mês de Agosto foi 10056. Total de 2090.

3. Os 5 URLs que mais causaram o erro 404.
   A URLS que mais causaram o erro 404 para o mês de Julho foram as listadas a seguir:

```
scala> address_404_error_desc.show(5, false)
+--------------------------+-----------+-----+
|address                   |return_code|count|
+--------------------------+-----------+-----+
|hoohoo.ncsa.uiuc.edu      |404        |251  |
|jbiagioni.npt.nuwc.navy.mil|404        |131  |
|piweba3y.prodigy.com      |404        |110  |
|piweba1y.prodigy.com      |404        |92   |
|phaelon.ksc.nasa.gov      |404        |64   |
+--------------------------+-----------+-----+
only showing top 5 rows
```

   E para o mês de Agosto as URLs foram:

```
scala> address_404_error_desc.show(5,false)
+---------------------+-----------+-----+
|address              |return_code|count|
+---------------------+-----------+-----+
|dialip-217.den.mmc.com|404        |62   |
|piweba3y.prodigy.com |404        |47   |
|155.148.25.4         |404        |44   |
|maz3.maz.net         |404        |39   |
|gate.barr.com        |404        |38   |
+---------------------+-----------+-----+
only showing top 5 rows
```

4. Quantidade de erros 404 por dia.
   Quantidade de erros por dia para Julho:

```
scala> records_updated.groupBy("date","return_code").count().where("return_code = 404").show(100, false)
+----------+-----------+-----+
|date      |return_code|count|
+----------+-----------+-----+
|1995-07-06|404        |640  |
|1995-07-21|404        |334  |
|1995-07-02|404        |291  |
|1995-07-16|404        |257  |
|1995-07-18|404        |465  |
|1995-07-24|404        |328  |
|1995-07-15|404        |254  |
|1995-07-05|404        |497  |
|1995-07-07|404        |570  |
|1995-07-12|404        |471  |
|1995-07-03|404        |474  |
|1995-07-19|404        |639  |
|1995-07-20|404        |428  |
|1995-07-11|404        |471  |
|1995-07-23|404        |233  |
|1995-07-17|404        |406  |
|1995-07-22|404        |192  |
|1995-07-25|404        |461  |
|1995-07-10|404        |398  |
|1995-07-27|404        |336  |
|1995-07-04|404        |359  |
|1995-07-28|404        |94   |
|1995-07-26|404        |336  |
|1995-07-01|404        |316  |
|1995-07-13|404        |532  |
|1995-07-09|404        |348  |
|1995-07-14|404        |413  |
|1995-07-08|404        |302  |
+----------+-----------+-----+
```

Quantidade de erros por dia para Agosto:

```
scala> records_updated.groupBy("date","return_code").count().where("return_code = 404").show(100,false)
+----------+-----------+-----+
|date      |return_code|count|
+----------+-----------+-----+
|1995-08-05|404        |236  |
|1995-08-11|404        |263  |
|1995-08-26|404        |366  |
|1995-08-27|404        |370  |
|1995-08-06|404        |373  |
|1995-08-17|404        |271  |
|1995-08-14|404        |287  |
|1995-08-21|404        |305  |
|1995-08-25|404        |415  |
|1995-08-07|404        |537  |
|1995-08-18|404        |256  |
|1995-08-03|404        |304  |
|1995-08-31|404        |526  |
|1995-08-15|404        |327  |
|1995-08-20|404        |312  |
|1995-08-22|404        |288  |
|1995-08-30|404        |571  |
|1995-08-19|404        |209  |
|1995-08-28|404        |410  |
|1995-08-04|404        |346  |
|1995-08-10|404        |315  |
|1995-08-29|404        |420  |
|1995-08-09|404        |279  |
|1995-08-13|404        |216  |
|1995-08-16|404        |259  |
|1995-08-12|404        |196  |
|1995-08-24|404        |420  |
|1995-08-08|404        |391  |
|1995-08-23|404        |345  |
|1995-08-01|404        |243  |
+----------+-----------+-----+
```

:

5. O total de bytes retornados.
   Foram retornados 3.8695973491E10 bytes para o mês de Julho.

```
scala> val sum_bytes =  complete_records_updated.agg(sum("returned_bytes"))
sum_bytes: org.apache.spark.sql.DataFrame = [sum(returned_bytes): double]

scala> sum_bytes.show(false)
+------------------+
|sum(returned_bytes)|
+------------------+
|3.8695973491E10    |
+------------------+
```

   E 2.6828341424E10 bytes para o mês de Agosto.

```
scala> sum_bytes.show(false)
+------------------+
|sum(returned_bytes)|
+------------------+
|2.6828341424E10    |
+------------------+
```

   O total de bytes dos dois meses foi 6.5524314915E10.