Pontifícia Universidade Católica do Paraná

MICHAEL DA SILVA

FRAMEWORK BIG DATA

Curitiba - PR

2019

SUMÁRIO

1 INTRODUÇÃO

Nos foi apresentado nessa disciplina alguns dataset, na semana 2, junto com seu dicionário tipo, formato dos campos e descrição para que podemos usar como exemplo utilizando o framework Apache Spark, uma ferramenta Big Data que tem o objetivo de processar grandes conjuntos de dados de forma paralela e distribuída.

Mas o dataset que vamos mesmo trabalhar, pra podemos ter uma introdução, foi o de ocorrências criminais, que contem milhares de registro, junto com a linguagem Java.

2 OBJETIVO

O objetivo desse Projeto é apresentar apenas o modelo de programação do Spark utilizando os RDDs e as operações. O projeto teve por torná-lo um mecanismo de execução que funciona tanto na memória como em disco e, por isso, o Spark executa operações em disco quando os dados não cabem mais na memória. Assim, é possível usá-lo para o processamento de conjuntos de dados maiores que a memória agregada em um cluster. Visando saída de informações, sobre os dados do dataset, permitindo estratégia pra diminuir e contem certo tipo de crime.

3 MÉTODOS

O shell do Spark fornece uma maneira simples de aprender a API, bem como uma ferramenta poderosa para analisar dados interativamente. Ele está disponível em Scala (que é executado no Java VM e, portanto, é uma boa maneira de usar as bibliotecas Java existentes)

Spark introduziu o conceito de RDD, que formalmente é uma coleção de objetos somente leitura, particionados em um conjunto de nodos do cluster, podendo somente ser criado através de funções determinísticas (map, filter, join, groupBy) executadas em outros RDDs ou meios de armazenamentos estáveis como o Hadoop Filesystem. Basicamente RDDs suportam dois tipos de operações: Transformations (map, filter, join, union, etc) e Actions (reduce, count, first, etc).

Configurar uma aplicação Spark é bastante simples: basta adicionar a dependência da ferramenta no Maven. Resilient Distributed Datasets (RDD): abstraem um conjunto de objetos distribuídos no cluster, geralmente executados na memória principal. Estes podem estar armazenados em sistemas de arquivo tradicional, no HDFS (HadoopDistributed File System) e em alguns Banco de Dados NoSQL, como Cassandra e HBase. Ele é o objeto principal do modelo de programação do Spark, pois são nesses objetos que serão executados os processamentos dos dados.

Operações: representam transformações (como agrupamentos, filtros e mapeamentos entre os dados) ou ações (como contagens e persistências) que são realizados em um RDD. Um programa Spark normalmente é definido como uma sequência de transformações ou ações que são realizadas em um conjunto de dados.

Spark Context: o contexto é o objeto que conecta o Spark ao programa que está sendo desenvolvido. Ele pode ser acessado como uma variável em um programa que para utilizar os seus recursos.

4 RESULTADOS E DISCUSSÕES

Quantidade de crimes por ano?

2014 = 547438, **2003** = 944292, **2013** = 612698, **2002** = 944160, **2018** = 238950, **2007** = 871402, **2004** = 934472, **2015** = 515922 ,**2011** = 702608, **2001** = 965880, **2008** = 840362,

2012 = 670854, **2005** = 900024, **2010** = 739988, **2009** = 772350, 2016 = 534500, **2017** = 530138, **2006** = 891146.

Quantidade de crimes por ano que sejam do tipo NARCOTICS?

2014 = 57812, **2003** = 107542, **2013** = 68212, **2002** = 99704, **2018** = 11576,

2007 = 108824, **2004** = 113764, **2015** = 43216, **2011** = 77158, **2001** = 100644

2008 = 90144, **2012** = 70942, **2005** = 111904, **2010** = 86780, **2009** = 84328,

2016 = 26514, **2017** = 22950, **2006** = 110684.

Quantidade de crimes por ano que sejam do tipo NARCOTICS e tenham ocorrido em dias pares?

2014 = 28162, **null** = 12477142, **2003** = 52612, **2013** = 33574, **2002** = 48796,

2018 = 5680, **2007** = 53638, **2004** = 55356, **2015** = 21238, **2011** = 37630,

2001 = 49078, **2008** = 43988, **2012** = 34616, **2005** = 53684, **2010** = 42376,

2009 = 41390, **2016** = 13004, **2017** = 11104, **2006** = 54116.

Mês com maior ocorrência de crimes?

05=109328.

Mês com a maior média de ocorrência de crimes?

Mês por ano com a maior ocorrência de crimes?

Mês com a maior ocorrência de crimes do tipo DECEPTIVE PRACTICE?

04=5560.

Dia do ano com a maior ocorrência de crimes?

01=36295.

Estão algumas informações sobre alguns dados, com isso podemos ver o dia que tem mais crime ou filtra o tipo de ocorrencia, podendo assim evitar mais ocorrencia desse tipo.

5 CONCLUSÃO

Abordamos como o framework Apache Spark ajuda no processamento e analise de Big Data com sua API padrão. Baseado no mesmo sistema de armazenamento de arquivos HDFS como o Haddop, assim é possível usar Spark e o MapReduce em conjunto. Isto é especialmente interessante se você já tem um investimento significativo e configuração de infraestrutura com o Hadoop.

Com várias integrações e adaptadores, é possível combinar o Spark com outras tecnologias. Um exemplo disto é a utilização de Spark, Kafka, e Apache Cassandra juntos, no qual o Kafka pode ser utilizada para entrada dos dados de streaming, o Spark para fazer processamento e, finalmente, armazenar no banco de dados Cassandra os resultados desse processamento.

CÓDIGO-FONTE E LOGS DE EXECUÇÃO

```
package com.mycompany.mavenproject1;
import java.util.Map;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;

public class Mainspark {

    public static void main(String args[]){

        // CONFIGURAÇÃO SPARK
        SparkConf conf = new SparkConf().setMaster("local[*]").setAppName("praticas");
        JavaSparkContext sc = new JavaSparkContext(conf);

        //CARREGA ARQUIVO
        JavaRDD<String> arquivo =
sc.textFile("/home/Disciplinas/Frameworks/spark/ocorrencias_criminais_sample.csv");
        //FILTRA OS DADOS POR ANO
        JavaRDD<String> anoRDD = arquivo.map(s -> {
              String[]campos = s.split(";");
              return campos[2];
           });

        resultadoNumero (anoRDD.countByValue());
    }
        public static void resultadoNumero (Map< String, Long> genericRDD){
           genericRDD.entrySet().forEach((entrada) -> {
              System.out.println(entrada.getKey() +  " = " + entrada.getValue());
           });
        }
}

        //FILTRA POR NARCOTICS
        JavaRDD<String> filtrado = arquivo.filter(s ->  {
              String [] campos = s.split(";");
              String tipo = campos[4];
              return tipo.equalsIgnoreCase("NARCOTICS");
        });

        //FILTRA POR DIA PAR
        JavaRDD<String> anoRDD = arquivo.map(s -> s.split(";")[4].equals("NARCOTICS")&&
Integer.parseInt(s.split(";")[0])%2==0 ? s.split(";")[2] : null);



        //FILTRA POR MES
        JavaRDD<Integer> mes = arquivo.map(s -> Integer.parseInt(s.split(";")[1]));
        JavaRDD<Integer> ordenado = mes.sortBy(s -> s, false, 0);
```

```
// FILTRA POR DECEPTIVE PRACTICE
JavaRDD<String> filtrado = arquivo.filter(s ->  {
        String [] campos = s.split(";");
        String tipo = campos[4];
        return tipo.equalsIgnoreCase("DECEPTIVE PRACTICE");
 });
 JavaRDD<String> anoRDD = filtrado.map(s -> {
        String[]campos = s.split(";");
        return campos[1];
    });
 JavaRDD<String> ordenado = anoRDD.sortBy(s -> Integer.parseInt(s), false, 0);

//FILTRA POR DIA
JavaRDD<Integer> dia = arquivo.map(s -> Integer.parseInt(s.split(";")[0]));
JavaRDD<Integer> ordenado = dia.sortBy(s -> s, false, 0);
System.out.println(ordenado.countByValue());
```

LOGIN:silva.michael

SENHA:larice3106@

REFERÊNCIAS

https://www.infoq.com/br/articles/apache-spark-introduction/

https://www.lume.ufrgs.br/bitstream/handle/10183/193992/001092825.pdf?sequence=1

https://www.devmedia.com.br/introducao-ao-apache-spark/34178