NILTON KAZUYUKI UEDA

DATA ANALYTICS

FRAMEWORK DE BIG DATA

AULA 04

SUMÁRIO

O QUE VEM POR AÍ?

Durante esta aula, você será introduzido(a) aos conceitos fundamentais do Apache Spark e aprenderá como trabalhar com Datasets e DataFrames, que são abstrações de alto nível que permitem a manipulação e a análise de dados estruturados de forma eficiente. Você explorará as principais operações disponíveis, como filtragem, transformação, agregação e junção de dados, tudo isso de forma distribuída e otimizada pelo Spark.

Ao longo da disciplina, você terá a oportunidade de praticar a manipulação de Datasets e DataFrames usando a linguagem de programação Scala ou Python, que são amplamente utilizadas no ecossistema Spark. Você aprenderá a aplicar operações avançadas em grandes volumes de dados, garantindo a eficiência e a escalabilidade necessárias para lidar com análises complexas.

Prepare-se para mergulhar no universo dos Datasets e DataFrames no Apache Spark e descubra como essas estruturas podem impulsionar suas habilidades em manipulação de dados estruturados. Com essa aula introdutória, você estará preparado(a) para enfrentar desafios analíticos mais sofisticados e aproveitar ao máximo o poder do Spark para análise de Big Data.

Não perca a oportunidade de dominar a manipulação de Datasets e DataFrames no Apache e expandir suas habilidades em análise de dados estruturados. Com esse conhecimento, você estará pronto(a) para explorar novas possibilidades e aproveitar o potencial ilimitado do Spark para transformar dados em informações valiosas.

HANDS ON

Agora conseguiremos entender, na prática, as principais diferenças entre DataSets e DataFrames. Vamos colocar a mão na massa nos conceitos e aprender muito com o nosso Hands On.

SAIBA MAIS

APIS SPARK RDD

Um RDD significa Conjuntos de Dados Distribuídos Resilientes (Resilient Distributed Dataset). É uma coleção de registros de partição em que só é possível a leitura. O RDD é a estrutura de dados fundamental do Spark e permite que especialistas em programação executem cálculos na memória em grandes grupos de maneira tolerante a falhas.

Já nos APIs do Spark DataFrame, ao contrário de um RDD, os dados são organizados em colunas nomeadas. Como, por exemplo, uma tabela em um banco de dados relacional; é uma coleção imutável de dados distribuídos. O DataFrame no Spark permite que especialistas em desenvolvimento possam impor uma estrutura em uma coleção distribuída de dados, permitindo abstração de nível superior.

Enquanto isso, nas APIs do conjunto de dados Spark (DataSets), os conjuntos de dados no Apache são uma extensão da API DataFrame, que fornece uma interface de programação orientada a objetos e com segurança de tipo. O conjunto de dados aproveita o otimizador Catalyst do Spark expondo expressões e campos de dados ao indivíduo planejador de consultas.

RDD x DATAFRAME x DATASET

O RDD é uma coleção distribuída de elementos de dados espalhados por muitas máquinas no cluster. Lembrando que RDDs são um conjunto de objetos Java ou Scala que representam dados.

Já o DataFrame é uma coleção distribuída de dados organizados em colunas nomeadas. É conceitualmente igual a uma tabela em um banco de dados relacional.

E, por fim, o DataSet é uma extensão da API DataFrame que fornece a funcionalidade de interface de programação orientada a objetos e segura para tipos de dados da API RDD, benefícios de desempenho do otimizador de consulta Catalyst e mecanismo de armazenamento fora da pilha de uma API DataFrame.

FORMATOS DE DADOS

O RDD pode processar de maneira fácil e eficiente dados estruturados e não estruturados. Mas, como o DataFrame e o DataSets, o RDD não infere no esquema dos dados ingeridos e exige que o usuário os especifique.

Já o DataFrame funciona apenas em dados estruturados e semiestruturados; ele organiza os dados na coluna nomeada. Os DataFrames permitem que o Spark gerencie o esquema.

O DataSet também processa eficientemente dados estruturados e não estruturados. Ele representa dados na forma de objetos de linha da JVM ou uma coleção de objetos de linha que é representada em formas de tabela através de codificadores.

API DA FONTE DE DADOS

A API da fonte de dados permite que um RDD possa vir de qualquer fonte de dados, como arquivo de texto, banco de dados via JDBC e etc., e manipular facilmente dados sem estrutura predefinida.

Para o DataFrame, a API da fonte de dados permite o processamento de dados em diferentes formatos (AVRO, CSV, JSON e sistema de armazenamento HDFS, tabelas HIVE, MySQL).

A API do conjunto de dados do Spark também suporta dados de diferentes fontes no DataSet.

IMUTABILIDADE E INTEROPERABILIDADE

RDDs contêm a coleção de registros que são particionados. A unidade básica de paralelismo em um RDD é chamada de partição. Cada partição é uma divisão lógica de dados que é imutável e criada através de alguma transformação nas partições existentes, sendo que a imutabilidade ajuda a obter consistência nos cálculos. Podemos passar do RDD para o DataFrame (se o RDD estiver no formato tabular) pelo método toDF () ou podemos fazer o inverso pelo método .rdd. Aprenda várias APIs de transformações e ações de RDD com exemplos.

Após a transformação no DataFrame, não é possível regenerar um objeto de domínio. Por exemplo: se você gerar testDF a partir de testRDD não poderá recuperar o RDD original da classe de teste.

O DataSet supera a limitação do DataFrame para regenerar o RDD do DataFrame. Os conjuntos de dados permitem converter seus RDD e DataFrames existentes em conjuntos de dados.

USO DE EFICIÊNCIA/MEMÓRIA

No RDD, a eficiência diminui quando a serialização é realizada individualmente em um objeto Java e Scala que levam muito tempo.

No DataFrame, o uso de memória heap desativada para serialização reduz a sobrecarga. Ele gera código de bytes dinamicamente para que muitas operações possam ser executadas nesses dados serializados. Não há necessidade de desserialização para pequenas operações.

Já o DataSet permite executar uma operação em dados serializados e melhorar o uso da memória. Assim, ele possibilita o acesso sob demanda ao atributo individual sem desserializar o objeto inteiro.

SUPORTE À LINGUAGEM DE PROGRAMAÇÃO

APIs RDD estão disponíveis em Java, Scala, Python, e R. Portanto, esse recurso fornece flexibilidade a especialistas em desenvolvimento.

O DataFrame também possui APIs em diferentes linguagens, como Java, Python, Scala e R.

Para o DataSet, conjuntos de dados estão disponíveis apenas no Scala e Java. O Spark versão 2.1.1 não suporta Python e R.

AGREGAÇÃO

A API RDD é mais lenta para executar operações simples de agrupamento e agregação.

A API DataFrame é muito fácil de usar: é mais rápida para análise exploratória, criando estatísticas agregadas em grandes conjuntos de dados.

Para o DataSet, no conjunto de dados é mais rápido executar a operação de agregação em diversos conjuntos de dados.

ENTENDENDO NA PRÁTICA COMO É O APACHE SPARK

1. Criar PySpark RDD

Primeiro, vamos criar um RDD passando o objeto de lista do Python para sparkContext.parallelize()a função. Nós precisaríamos deste RDD para todos os nossos exemplos a seguir.

No PySpark, quando você tem dados em uma lista, significa que você tem uma coleção de dados na memória do driver PySpark ao criar um RDD e essa coleção será paralelizada.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Tutorial).getOrCreate()
dept = [("Finance",10),("Marketing",20),("Sales",30),("IT",40)]
rdd = spark.sparkContext.parallelize(dept)
```

2. Converter PySpark RDD em DataFrame

A conversão do PySpark RDD para DataFrame pode ser feita usando toDF(), createDataFrame(). Nesta seção, explicaremos esses dois métodos.

2.1 Usando a função rdd.toDF()

O PySpark fornece função toDF() em RDD que pode ser usada para converter RDD em DataFrame

```
df = rdd.toDF()
df.printSchema()
df.show(truncate=False)
```

Por padrão, toDF()a função cria nomes de coluna como “_1” e “_2”. Este snippet produz o esquema a seguir.

```
root
 |-- _1: string (nullable = true)
 |-- _2: long (nullable = true)

+--------+---+
|_1      |_2 |
+--------+---+
|Finance |10 |
|Marketing|20 |
|Sales   |30 |
|IT      |40 |
+--------+---+
```

toDF() tem outra assinatura que usa argumentos para definir os nomes das colunas, conforme é possível observar a seguir.

```
deptColumns = ["dept_name","dept_id"]
df2 = rdd.toDF(deptColumns)
df2.printSchema()
df2.show(truncate=False)
```

Segue saída do esquema.

```
root
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)

+--------+-------+
|dept_name|dept_id|
+--------+-------+
|Finance |10     |
|Marketing|20     |
|Sales   |30     |
|IT      |40     |
+--------+-------+
```

2.2 Usando a função createDataFrame() do PySpark

O SparkSession fornece createDataFrame(), que é um método para criar DataFrame e leva o RDD como um argumento dentro da função.

```
deptDF = spark.createDataFrame(rdd, schema = deptColumns)
deptDF.printSchema()
deptDF.show(truncate=False)
```

Isso produz a mesma saída apresentada.

2.3 Usando createDataFrame() com esquema StructType

Quando você infere o esquema, por padrão, o tipo de dados das colunas é derivado dos dados e definido como nullable e true para todas as colunas. Podemos alterar esse comportamento fornecendo o esquema usando StructType – em que podemos especificar um nome de coluna, tipo de dados e anulável para cada campo/coluna.

Se você quiser saber mais sobre StructType, veja como usar StructType e StructField para definir o esquema personalizado.

```
rom pyspark.sql.types import StructType,StructField, StringType
deptSchema = StructType([
    StructField('dept_name', StringType(), True),
    StructField('dept_id', StringType(), True)
])

deptDF1 = spark.createDataFrame(rdd, schema = deptSchema)
deptDF1.printSchema()
deptDF1.show(truncate=False)
```

O QUE VOCÊ VIU NESTA AULA?

Durante a disciplina, aprendemos a utilizar as principais operações, como filtragem, transformação, agregação e junção de dados, de forma distribuída e otimizada pelo Spark. Foram aplicadas práticas com as linguagens Scala ou Python, amplamente utilizadas no ecossistema do Spark, permitindo a aplicação de operações avançadas em grandes volumes de dados.

Nesta aula, vimos algumas diferenças entre RDD, DataFrama e DataSet. Gostou do que aprendeu? Converse sobre a disciplina conosco no Discord! Lá podemos tirar dúvidas, interagir e muito mais!

REFERÊNCIAS

APACHE STARK. **Documentação Oficial Apache Spark**. Disponível em: https://spark.apache.org/documentation.html. Acesso em: 28 jun. 2023.

PENCHIKALA, S. **Big Data com Apache Spark Parte 1: Introdução**. 2015. Disponível em: https://www.infoq.com/br/articles/apache-spark-introduction/. Acesso em: 29 jun. 2023.

PENCHIKALA, S. **Big Data com Apache Spark Parte 2: Spark SQL**. 2015. Disponível em: https://www.infoq.com/br/articles/apache-spark-sql/. Acesso em: 29 jun. 2023.

RELVA, C. **Apache Spark**. 2015. Disponível em: https://www.ime.usp.br/~gold/cursos/2015/MAC5742/reports/ApacheSpark.pdf. Acesso em: 29 jun. 2023.

PALAVRAS-CHAVE

Palavras-chave: Apache Spark, Big Data, SQLContext, Hadoop, MapReduce, Big Data, Spark, Dados, Processamento.