Custom Learnings

Day 16

Spark & PySpark

It is used to process data in the spark engine.

Apache spark is a unified analytics engine for large scale data processing.

For distributed computing there should be one driver and 0 or more workers.

Inside the worker there can be multiple executors.

Spark is fully configurable.

Slot : Inside the executor will be multiple slots which will do the actual work.

Driver and executor are JVMs.

Spark supports: Python, Scala, R and SQL.

Entire spark engine is built on java code.

Spark breaks up the data into chunks called partitions, in order to allow every executor to perform work in parallel.

Lazy evaluation: Until and unless you call an action the transformation will not get executed. Show() or write() method is used for the same.

For one action one job gets created.

RDD Operations:

Transformation: reduce, map, groupyby, etc.

Action: count, sum, collect, etc.

Types of Transformation:

1. Narrow Transformation 2. Wide Transformation

Shuffle is very costly which happens during wide transformation.

How many task is created in a stage depends on the number of input partitions.

Based on number of shuffling stage gets created.

Cluster manager is responsible for launching tasks in the worker nodes. It manages the master node and the worker node in the cluster.

A Spark job can be any task that needs to be performed on a large amount of data that is too big to be processed on a single machine. Whenever an action is executed, the job gets created. Spark doesn't run an action it creates a job and then runs the job.

A Spark job is divided into Spark stages, where each stage represents a set of tasks that can be executed in parallel. A stage consists of a set of tasks that are executed on a set of partitions of the data. It represents a sequence of transformations that can be executed in a single pass without the need of shuffling. All the tasks within a stage perform the same computation.

In-Memory Processing: Using the spark caching we can avoid going to spark nodes every time to fetch the data.

Fundamental Components of Spark:

RDD(Resilient Distributed Dataset): If the one of the nodes gets failed and is able to recover quickly on its own.

Catalyst and Tungsten:

Catalog always holds the metadata.

Logical plan: Actual plan that the machine is going to conduct in the backend.

Physical Plan: Final plan that is passed to machine to get executed.

Based on cost model physical plan is selected.

The selected physical plan is converted into RDD.

Adaptive Query Execution:

Adapts changes during the runtime based on the feedback and gives it to the logical plan. Based on this the logical plan will get modified and accordingly RDDs will be formed.

For reference: https://medium.com/@harun.raseed093/spark-logical-and-physical-plans-e111de6cc22e

For structured data: Dataframe will provide SQL rich API.

In [1]:
```
import findspark
```

In [2]:
```
findspark.init()
```

In [4]:
```
from pyspark.sql import SparkSession
#Initilize Sparksession
spark = SparkSession.builder.appName("WordCount").getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/09/21 06:22:31 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-ja
va classes where applicable
```

In [6]:
```
rdd=spark.sparkContext.parallelize([1,2,3,4])
```

In [7]:
```
rdd.collect()
```

Out[7]:
```
[1, 2, 3, 4]
```

In [8]:
```
resultrdd = rdd.map(lambda x:x*2)
```

In [9]:
```
resultrdd.collect()
```

Out[9]:
```
[2, 4, 6, 8]
```

In [10]:
```
rdd1 = spark.sparkContext.parallelize([1,2,3,4,5])
```

In [11]:
```
resultrdd1 = rdd1.flatMap(lambda x:(x,x*2))
resultrdd1.collect()
```

Out[11]:
```
[1, 2, 2, 4, 3, 6, 4, 8, 5, 10]
```

In [12]:
```
rdd3 = spark.sparkContext.parallelize([1,2,3,4,5,6,7])
```

In [13]:
```
resultrdd3=rdd3.filter(lambda x:x % 2==0)
resultrdd3.collect()
```

Out[13]:
```
[2, 4, 6]
```

In [12]:
```
rdd4 = spark.sparkContext.parallelize([(1,2),(3,4),(1,6),(2,3)])
res4 = rdd4.reduceByKey(lambda x,y : x+y)
res4.collect()
```

Out[12]:
```
[(2, 3), (1, 8), (3, 4)]
```

In [14]:
```
rdd5 = spark.sparkContext.parallelize([(1,2),(3,4),(1,6),(2,3)])
res5 = rdd5.groupByKey()
res5.collect()
```

Out[14]:
```
[(2, <pyspark.resultiterable.ResultIterable at 0x7fc3406577d0>),
 (1, <pyspark.resultiterable.ResultIterable at 0x7fc340647850>),
 (3, <pyspark.resultiterable.ResultIterable at 0x7fc340657810>)]
```

In [15]:
```
for key, values in res5.collect():
    print(f"Key: {key}, Values:{list(values)}")
```

```
[Stage 9:>                                                          (0 + 2) / 2]
```

```
Key: 2, Values:[3]
Key: 1, Values:[2, 6]
Key: 3, Values:[4]
```

In [18]:
```
word_list = ["this", "is", "a", "sample", "text", "document", "for", "word", "count", "example", "word", "count"]
rdd = spark.sparkContext.parallelize(word_list)
words_count = rdd.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)
results = words_count.collect()
for word, count in results:
    print(f"{word}: {count}")
```

```
this: 1
sample: 1
text: 1
for: 1
word: 2
is: 1
a: 1
document: 1
count: 2
example: 1
```

In [20]:
```
purchaserdd = spark.sparkContext.textFile("/home/labuser/Documents/Pandas_datasets/purchases.csv")
purchaserdd.collect()
```

Out[20]:
```
[',apples,oranges', 'June,3,0', 'Robert,2,3', 'Lily,0,7', 'David,1,2']
```

In [21]:
```
purchasedf=spark.read.csv("/home/labuser/Documents/Pandas_datasets/purchases.csv")
```

In [22]:
```
purchasedf.show()
```

```
+------+------+-------+
|   _c0|   _c1|    _c2|
+------+------+-------+
|  null|apples|oranges|
|  June|     3|      0|
|Robert|     2|      3|
|  Lily|     0|      7|
| David|     1|      2|
+------+------+-------+
```

In [23]:
```
purchasedf_01=spark.read.option("inferschema", True).option("header", True).csv("/home/labuser/Documents/Pandas_data
purchasedf_01.show()
```

```
+------+------+-------+
|   _c0|apples|oranges|
+------+------+-------+
|  June|     3|      0|
|Robert|     2|      3|
```

Giving Schema Manually:

In [35]:
```
moviedf=spark.read.schema(movieSchema).option("header", True).csv("/home/labuser/Documents/Pandas_datasets/IMDB-Mov
moviedf.printSchema()
```

```
root
 |-- Rank: integer (nullable = true)
 |-- Title: string (nullable = true)
 |-- Genre: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Director: string (nullable = true)
 |-- Actors: string (nullable = true)
 |-- Year: string (nullable = true)
 |-- Runtime (Minutes): string (nullable = true)
 |-- Rating: string (nullable = true)
 |-- Votes: string (nullable = true)
 |-- Revenue (Millions): double (nullable = true)
 |-- Metascore: double (nullable = true)
```

In [34]:
```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType
movieSchema=StructType([StructField("Rank", IntegerType(),True),
                        StructField("Title", StringType(),True),
                        StructField("Genre", StringType(),True),
                        StructField("Description", StringType(),True),
                        StructField("Director", StringType(),True),
                        StructField("Actors", StringType(),True),
                        StructField("Year", StringType(),True),
                        StructField("Runtime (Minutes)", StringType(),True),
                        StructField("Rating", StringType(),True),
                        StructField("Votes", StringType(),True),
                        StructField("Revenue (Millions)", DoubleType(),True),
                        StructField("Metascore", DoubleType(),True)])
```

Deployment types:

Client mode: Runs the process in the same machine.

Cluster mode: Runs the process in cluster machine.

Transformations:

1. withColumn: We can create new, calculated columns, even rename the column name with this.
2. withColumnRenamed: We rename the column name.

```
In [40]: from pyspark.sql.functions import *
moviedf = moviedf.withColumn("rev_col",col("Revenue (Millions)")*100).withColumn("Batch", lit("Batch3"))
moviedf.show()
```

```
+----+--------------------+--------------------+--------------------+--------------------+--------------------+---
----------------+----------------+------+------+-----------------+---------+----------------+------+
|Rank|               Title|               Genre|         Description|            Director|              Actors|
Year|Runtime (Minutes)|Rating| Votes|Revenue (Millions)|Metascore|          rev_col| Batch|
+----+--------------------+--------------------+--------------------+--------------------+--------------------+---
----------------+----------------+------+------+-----------------+---------+----------------+------+
|   1|Guardians of the ...|Action,Adventure,...|A group of interg...|          James Gunn|Chris Pratt, Vin ...|
2014|             121|   8.1|757074|            333.13|     76.0|         33313.0|Batch3|
|   2|          Prometheus|Adventure,Mystery...|Following clues t...|        Ridley Scott|Noomi Rapace, Log...|
2012|             124|     7|485820|            126.46|     65.0|         12646.0|Batch3|
|   3|               Split|     Horror,Thriller|Three girls are k...|  M. Night Shyamalan|James McAvoy, Any...|
2016|             117|   7.3|157606|            138.12|     62.0|         13812.0|Batch3|
|   4|                Sing|Animation,Comedy,...|In a city of huma...|Christophe Lourdelet|Matthew McConaugh...|
2016|             108|   7.2| 60545|            270.32|     59.0|         27032.0|Batch3|
|   5|       Suicide Squad|Action,Adventure,...|A secret governme...|          David Ayer|Will Smith, Jared...|
2016|             123|   6.2|393727|            325.02|     40.0|         32502.0|Batch3|
|   6|      The Great Wall|Action,Adventure,...|European mercenar...|         Yimou Zhang|Matt Damon, Tian ...|
2016|             103|   6.1| 56036|             45.13|     42.0|          4513.0|Batch3|
|   7|          La La Land|  Comedy,Drama,Music|A jazz pianist fa...|     Damien Chazelle|Ryan Gosling, Emm...|
2016|             128|   8.3|258682|            151.06|     93.0|         15106.0|Batch3|
|   8|            Mindhorn|              Comedy|"A has-been actor...| whom he believes...|          Sean Foley|Ess
ie Davis, Andr...|            2016|    89|   6.4|            2490.0|     null|        249000.0|Batch3|
|   9|  The Lost City of Z|Action,Adventure,...|A true-life drama...|          James Gray|Charlie Hunnam, R...|
2016|             141|   7.1|  7188|              8.01|     78.0|           801.0|Batch3|
|  10|          Passengers|Adventure,Drama,R...|A spacecraft trav...|      Morten Tyldum|Jennifer Lawrence...|
```