PySpark Q&A

Youtube : #pysparkinterviewquestions

<https://www.davidmcginnis.net/post/spark-job-optimization-myth-1-increasing-the-memory-per-executor-always-improves-performance>

<https://spark.apache.org/faq.html>

<https://www.xenonstack.com/blog/apache-spark-optimisation>

1. What is spark?

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching and optimized query execution for fast queries against data of any size. Simply put, Spark is a fast and general engine for large-scale data processing.

1. What is the difference between spark and map reduce framework?

In Map Reduce framework the developer must know the working of mappers and reducers whereas in spark everything is handled under the hood.

1. What are the basic features of spark?

Speed, Distributed framework, Advanced Analytics, Real time processing, Powerful caching, fault tolerance and easy deployment.

1. Which is the spark version and python version you have worked with in your project?

2.4.3 and 2.7.5 respectively

1. What is pyspark?

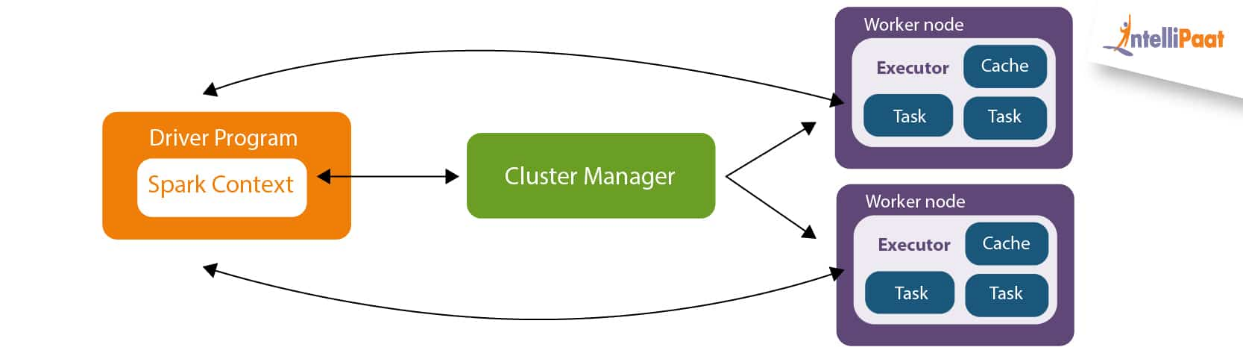
It is an apache spark API built in python.

1. What are the components of pyspark ecosystem?

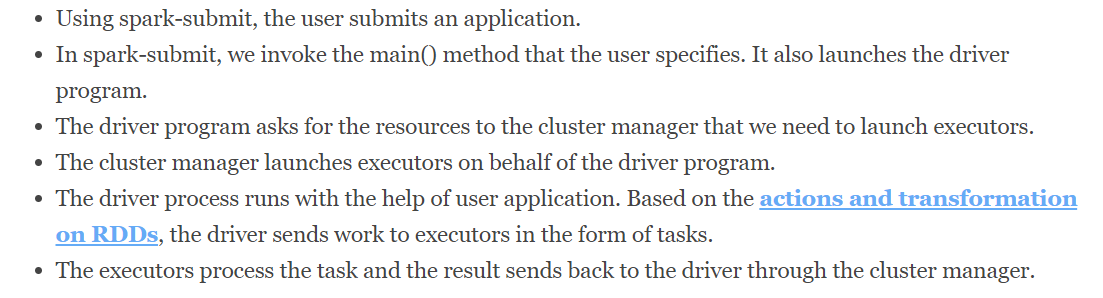
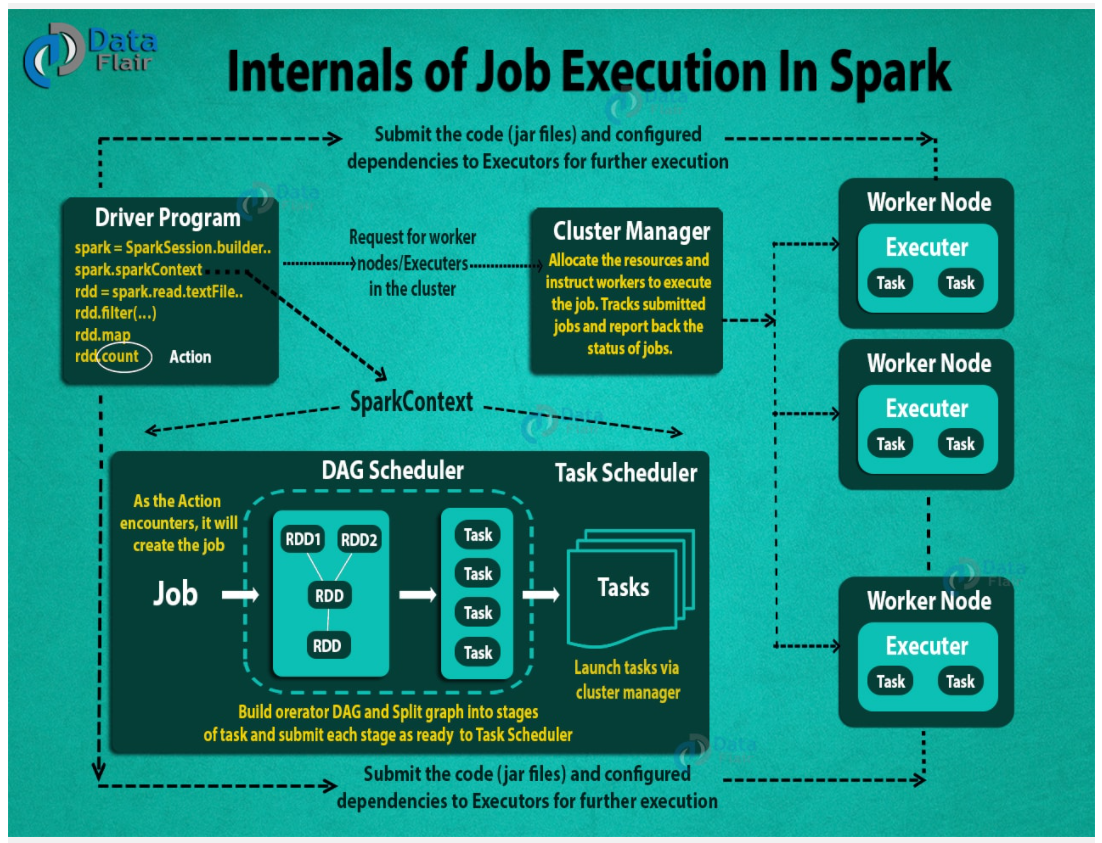
Spark SQL, Spark Streaming, MLlib, GraphX, SparkR and the Core API component.

1. Walk me through the spark architecture.

The Apache Spark framework uses a master–slave architecture that consists of a driver, which runs as a master node, and many executors that run across as worker nodes in the cluster.



1. What are the internals of job execution in spark? Or what is the job flow in spark?



1. What is a driver?

The spark driver is the program that declares the transformations and actions on RDDs of data and submits such requests to the master. In practical terms, the driver is the program that creates the SparkContext, connecting to a given Spark Master

Spark Driver contains various components – DAGScheduler, TaskScheduler, BackendScheduler and BlockManager. They are responsible for the translation of user code into actual Spark jobs executed on the cluster.

1. What is a transformation?

Spark Transformation is a function that produces new RDD from the existing RDDs. It takes RDD as input and produces one or more RDD as output. Each time it creates new RDD when we apply any transformation.

1. What is an action?

RDD actions are operations that return the raw values, In other words, any RDD function that returns other than RDD[T] is considered as an action in spark programming. Spark uses lazy evaluation which is only triggered when action is called.

1. Difference between action and transformation.

<https://blog.knoldus.com/deep-dive-into-apache-spark-transformations-and-action/>

<https://intellipaat.com/blog/interview-question/apache-spark-interview-questions/#10>

<https://www.besanttechnologies.com/pyspark-interview-questions-and-answers>

<https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/>

1. What is fold?

It takes function as an input which has two parameters of the same type and outputs a single value of the input type.

sc.parallelize([1,25,8,4,2]).fold(0,add) 🡪 40

sc.parallelize([1,25,8,4,2], 2).fold(10,add) 🡪(40+10) +2\*10(for each partition)

1. What are closures?

It is basically a conceptual entity meaning encapsulation of all variables and functions defining the scope of the variable

1. Practical use of coalesce

When you want to reduce the number of partitions and the data is not too huge such that it can reside in the memory, you can use coalesce instead of repartition to avoid high latency due to shuffle operations performed in repartition. It forces to partition based on the number provided and doesn’t shuffle and move extra data to the last partition.

Repartition is used for increasing or decreasing the number of partitions but internally it calls coalesce with shuffle = True. All partitions would be of same size.

1. What is a spark RDD?

It is spark’s core abstraction that stands for resilient distributed dataset

1. Why is RDD resilient?

Any RDD in Spark has a lineage graph and in case of any failure if we lose the data and we have a lineage we can recreate it.

1. Why is RDD immutable?

Immutable objects by definition are thread safe, meaning multiple processes can use same object without worrying about something updating the value of it. RDD is immutable so that it can be shared across the several nodes of the cluster to get the advantage of full cluster performance and multi-threading. Also for spark in memory computation it is good to have immutable objects to avoid frequent cache invalidation.

1. What is lineage and how it works in RDD and Dataframe?

Whenever a series of transformations are performed on an RDD, they are not evaluated immediately, but lazily(Lazy Evaluation). When a new RDD is created from an existing RDD, that new RDD contains a pointer to the parent RDD. Similarly, all the dependencies between the RDDs will be logged in a graph, rather than the actual data. This graph is called the lineage graph.

1. Difference between lineage and DAG.

RDD Lineage is just a portion of a DAG(one or more operations) that lead to the creation of that particular RDD. So one DAG(one Spark program) might create multiple RDDs, and each RDD will have its lineage - path in your DAG that lead to that RDD.

1. What is staging in spark?

A stage is a physical unit of execution that you can see in the spark job ui, which is just a set/series of tasks and which can execute in parallel.

1. What is the intention behind lazy evaluation?

In lazy evaluation data is not loaded until it is necessary. Transformations on RDD are lazy in nature and do not execute immediately.

By lazy evaluation, users can organize their Apache Spark program into smaller operations. It reduces the number of passes on data by grouping operations.

if you executed every transformation eagerly, what does that mean? Well, it means you will have to materialize that many intermediate datasets in memory. This is evidently not efficient -- for one, it will increase your GC costs. (Because you're really not interested in those intermediate results as such. Those are just convenient abstractions for you while writing the program.) So, what you do instead is -- you tell Spark what is the eventual answer you're interested and it figures out best way to get there.

1. I have given you an RDD . how will you convert it to paired RDD uisng its first element as key?

RDD2=RDD1.map(lambda x:(x[1], x))

1. What is DAG?

DAG in Apache Spark is a combination of Vertices as well as Edges. In DAG vertices represent the RDDs and the edges represent the Operation to be applied on RDD. Every edge in DAG is directed from earlier to later in a sequence.When we call an Action, the created DAG is submitted to DAG Scheduler which further splits the graph into the stages of the task.

1. Difference between DAG and lineage

Lineage is a logical graph that shows dependency of an RDD on other RDDs which was formed by what transformations, whereas DAG is a physical graph.

When an action is submitted, the lineage is submitted to the catalyst optimizer and the CO generates a final plan and submit to DAG and DAG generates a physical plan to identify RDDs and Stages and decides what can run in parallel or serial.

1. What are the steps followed in optimization?

Various ways to optimize the flow. It includes

* Spark caching
* Memory tuning
* Spark windowing function
* Using accumulators
* Hive bucketing performance
* Optimizing memory overhead

1. How to decide when to increase executors and when to increase memory? What things from logs will help in deciding this?  
   Assigning executors with a large number of virtual cores leads to a low number of executors and reduced parallelism. Assigning a low number of virtual cores leads to a high number of executors, causing a larger number of I/O operations. Based on historical data, we suggest that you have five virtual cores for each executor to achieve optimal results in any sized cluster.

<https://aws.amazon.com/blogs/big-data/best-practices-for-successfully-managing-memory-for-apache-spark-applications-on-amazon-emr/>

1. What are the memory errors in AWS that may rise with incorrect configuration of spark jobs?  
   In AWS, below are the errors that are raised in case of incorrect configuration of spark parameters
2. Out of Memory Error, Java Heap space

WARN TaskSetManager: Loss was due to

java.lang.OutOfMemoryError

java.lang.OutOfMemoryError: Java heap space

1. Out of Memory Error, Exceeding Physical Memory

Error: ExecutorLostFailure Reason: Container killed by YARN for exceeding limits.

12.4 GB of 12.3 GB physical memory used.

Consider boosting spark.yarn.executor.memoryOverhead.

Error: ExecutorLostFailure Reason: Container killed by YARN for exceeding limits.

4.5GB of 3GB physical memory used limits.

Consider boosting spark.yarn.executor.memoryOverhead

1. Out of Memory Error, Exceeding Virtual Memory

Container killed by YARN for exceeding memory limits.

1.1gb of 1.0gb virtual memory used. Killing container.

1. Out of Memory Error, Exceeding Executor Memory

Required executor memory (1024+384 MB) is above

the max threshold (896 MB) of this cluster! Please check the values of 'yarn.scheduler.maximum-allocation-mb'

and/or 'yarn.nodemanager.resource.memory-mb

These issues occur for various reasons, some of which are listed following:

When the number of Spark executor instances, the amount of executor memory, the number of cores, or parallelism is not set appropriately to handle large volumes of data.

When the Spark executor’s physical memory exceeds the memory allocated by YARN. In this case, the total of Spark executor instance memory plus memory overhead is not enough to handle memory-intensive operations. Memory-intensive operations include caching, shuffling, and aggregating (using reduceByKey, groupBy, and so on). Or, in some cases, the total of Spark executor instance memory plus memory overhead can be more than what is defined in yarn.scheduler.maximum-allocation-mb.

The memory required to perform system operations such as garbage collection is not available in the Spark executor instance.

For memory-intensive applications, prefer R type instances over the other instance types. For compute-intensive applications, prefer C type instances. For applications balanced between memory and compute, prefer M type general-purpose instances.

1. Difference between executor and executor core

Executor is a java process/yarn container running on the slave machine, where your code executes, it has vCPUs and RAM of its own.

Executor core is the number of threads/number of cores(vCPUs) started on the executors.

It defines the number of tasks each executor can run at the same time.

1. Difference between persist and cache

If you want to put your RDD in main memory use these options.

Cache : Data is stored in deserialized java objects, saves only in memory.

RDD.cache saves to Memory only and Spark DF or Dataset saved to Memory and disk because recomputing the in-memory columnar representation of the underlying table is expensive.

Cache on dataset internally calls persist.

Persist : You can define how you want to store, default is memory and disk, but you can chose from memory only, memory and disk, memory only ser, memory and disk ser, disk only, memory only 2 and memory and disk 2

Persist gives more control to store the data cache stores directly in memory.

1. Narrow vs wide transformation

Narrow transformations are the result of map() and filter() functions and these compute data that live on a single partition meaning there will not be any data movement between partitions to execute narrow transformations.

Functions such as map(), mapPartition(), flatMap(), filter(), union() are some examples of narrow transformation. Coalesce is also a narrow transformation as it does not include any shuffling.

Wider transformations are the result of groupByKey() and reduceByKey() functions and these compute data that live on many partitions meaning there will be data movements between partitions to execute wider transformations. Since these shuffles the data, they also called shuffle transformations.

Functions such as groupByKey(), aggregateByKey(), aggregate(), join(), repartition() are some examples of a wider transformations.

1. What are shared variables and its uses

Shared variables are the variables that are required to be used by many functions & methods in parallel. Shared variables can be used in parallel operations. Spark segregates the job into the smallest possible operation, a closure, running on different nodes and each having a copy of all the variables of the Spark job.

Sometimes, a variable need to be shared across tasks, or between tasks and the driver program. Spark supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only “added” to, such as counters and sums.

1. What are broadcast variables?

Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used, for example, to give every node a copy of a large input dataset in an efficient manner.

1. What are accumulator variables?

PySpark Accumulators are shared variables that can be updated by executors and propagates back to driver program. These variables are used to add sum or counts and final results can be accessed only by driver program.

1. How to define custom accumulator

<https://towardsdatascience.com/custom-pyspark-accumulators-310f63ca3c8c>

You just need to inherit class AccumulatorParam and the 2 methods therein as shown below.

class SetAccumulator(AccumulatorParam):

def zero(self, init\_value: set):

return init\_value

def addInPlace(self, v1: set, v2: set):

return v1.union(v2)

zero defines zero value of the accumulator type and addInPlace defines how two values of accumulator type are added together.

1. Difference between RDDs, DF and Datasets

RDD:

Type safe

Developer has to take care of optimizations

Not as good as datasets in performance

Not memory efficient

Dataframe: (tabular form built on top of RDD)

Not type safe

Auto optimization with catalyst optimizer

Not as good as datasets in performance

Not memory efficient

Datasets:

Type safe

Auto optimization

Better performance

More memory efficient

1. Create a UDF in pyspark

<https://sparkbyexamples.com/pyspark/pyspark-udf-user-defined-function/>

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, udf

from pyspark.sql.types import StringType

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

columns = ["Seqno","Name"]

data = [("1", "john jones"),

("2", "tracey smith"),

("3", "amy sanders")]

df = spark.createDataFrame(data=data,schema=columns)

df.show(truncate=False)

def convertCase(str):

resStr=""

arr = str.split(" ")

for x in arr:

resStr= resStr + x[0:1].upper() + x[1:len(x)] + " "

return resStr

""" Converting function to UDF """

convertUDF = udf(lambda z: convertCase(z))

df.select(col("Seqno"), \

convertUDF(col("Name")).alias("Name") ) \

.show(truncate=False)

def upperCase(str):

return str.upper()

upperCaseUDF = udf(lambda z:upperCase(z),StringType())

df.withColumn("Cureated Name", upperCaseUDF(col("Name"))) \

.show(truncate=False)

""" Using UDF on SQL """

spark.udf.register("convertUDF", convertCase,StringType())

df.createOrReplaceTempView("NAME\_TABLE")

spark.sql("select Seqno, convertUDF(Name) as Name from NAME\_TABLE") \

.show(truncate=False)

spark.sql("select Seqno, convertUDF(Name) as Name from NAME\_TABLE " + \

"where Name is not null and convertUDF(Name) like '%John%'") \

.show(truncate=False)

""" null check """

columns = ["Seqno","Name"]

data = [("1", "john jones"),

("2", "tracey smith"),

("3", "amy sanders"),

('4',None)]

df2 = spark.createDataFrame(data=data,schema=columns)

df2.show(truncate=False)

df2.createOrReplaceTempView("NAME\_TABLE2")

spark.udf.register("\_nullsafeUDF", lambda str: convertCase(str) if not str is None else "" , StringType())

spark.sql("select \_nullsafeUDF(Name) from NAME\_TABLE2") \

.show(truncate=False)

spark.sql("select Seqno, \_nullsafeUDF(Name) as Name from NAME\_TABLE2 " + \

" where Name is not null and \_nullsafeUDF(Name) like '%John%'") \

.show(truncate=False)

1. Create a hive UDF in spark

**CREATE** **TEMPORARY** **FUNCTION** testUDF **AS** 'org.apache.hadoop.hive.ql.udf.generic.GenericUDFAbs';

**CREATE** **TEMPORARY** **FUNCTION** hiveUDTF

**AS** 'org.apache.hadoop.hive.ql.udf.generic.GenericUDTFExplode';

**CREATE** **TEMPORARY** **FUNCTION** hiveUDAF

**AS** 'org.apache.hadoop.hive.ql.udf.generic.GenericUDAFSum';

If you use your own programmed function, you need to add a JAR containing it into a classpath.

1. How to decide various parameter values in spark-submit

To decide the parameters in spark-submit, you need to know the functionality as in how many executor cores will be needed, how many executor threads per executor will be needed, what will be the executor memory etc.

Simply increasing memory isn’t enough, because each task in a single core for an executor utilizes the entire memory assigned to it. Increasing number of executors will increase the number of tasks but there wont be any benefit with just increasing number of executors, as memory will need to be increased alongwith it. If that is not enough either, more number of cores can be defined to have even more number of tasks, but again parameters need to be changed accordingly. If that is not enough either, try fiddling with parallelism parameter.

1. What is data skew?

Data skew happens when for one reason or another, a small percentage of partitions get most of the data being processed.

In pyspark, you can use skew optimization hints as a work around for tables with skewed data. The hint works like below:

-- table with skew

SELECT /\*+ SKEW('orders') \*/ \* FROM orders, customers WHERE c\_custId = o\_custId

-- subquery with skew

SELECT /\*+ SKEW('C1') \*/ \*

FROM (SELECT \* FROM customers WHERE c\_custId < 100) C1, orders

WHERE C1.c\_custId = o\_custId

-- single column

SELECT /\*+ SKEW('orders', 'o\_custId') \*/ \*

FROM orders, customers

WHERE o\_custId = c\_custId

-- multiple columns

SELECT /\*+ SKEW('orders', ('o\_custId', 'o\_storeRegionId')) \*/ \*

FROM orders, customers

WHERE o\_custId = c\_custId AND o\_storeRegionId = c\_regionId

-- single column, single skew value

SELECT /\*+ SKEW('orders', 'o\_custId', 0) \*/ \*

FROM orders, customers

WHERE o\_custId = c\_custId

-- single column, multiple skew values

SELECT /\*+ SKEW('orders', 'o\_custId', (0, 1, 2)) \*/ \*

FROM orders, customers

WHERE o\_custId = c\_custId

-- multiple columns, multiple skew values

SELECT /\*+ SKEW('orders', ('o\_custId', 'o\_storeRegionId'), ((0, 1001), (1, 1002))) \*/ \*

FROM orders, customers

WHERE o\_custId = c\_custId AND o\_storeRegionId = c\_regionId

1. What is skew join and how it works

Skew join is a hive concept used to join tables with higher skewed data that can cause queries to run longer and may cause skewed data across partitions.

hive.optimize.skewjoin=true;

set hive.skewjoin.key=100000;

1. Why should we use group by transformations in spark
2. How to do mapside join in spark
3. If we have 50 GB memory and 100 GB data how will spark process it

Spark's operators spill data to disk if it does not fit in memory, allowing it to run well on any sized data. Likewise, cached datasets that do not fit in memory are either spilled to disk or recomputed on the fly when needed, as determined by the RDD's storage level.

1. Challenges you faced in spark project
2. Explain map, flatmap, mappartition, foreach, foreachpartition

* PySpark map (map()) is an RDD transformation that is used to apply the transformation function (lambda) on every element of RDD/DataFrame and returns a new RDD.

rdd.map(lambda x: (x,1))

or

rdd.map(lambda x: func1(x))

To use it on a dataframe, convert the dataframe to rdd first.

Df.rdd.map(lambda x: func1(x))

* PySpark flatMap() is a transformation operation that flattens the RDD/DataFrame (array/map DataFrame columns) after applying the function on every element and returns a new PySpark RDD/DataFrame. This is similar to dataframe’s explode function.
* Foreach is similar to map as it applies a function to all elements in the rdd. The important difference between them is that map accumulates all of the results into a collection, whereas foreach returns nothing. map is usually used when you want to transform a collection of elements with a function, whereas foreach simply executes an action for each element.
* Foreachpartition applies a function to each partition of an RDD. The difference between foreachPartition and mapPartition is that foreachPartition is a Spark action while mapPartition is a transformation. This means the code being called by foreachPartition is immediately executed and the RDD remains unchanged while mapPartition can be used to create a new RDD.

1. What is pair rdd and when to use them

Paired RDD is a distributed collection of data with the key-value pair. It is a subset of Resilient Distributed Dataset So it has all the features of RDD and some new feature for the key-value pair. There are many transformation operations available for Paired RDD.

It is much like python dictionary where you store key value pairs.

Use pair rdds when you want to perform operations in parallel on keys. There are transformations in pyspark which work on rdd keys like groupByKey(), reduceByKey(), combineByKey(), mapValues(), keys(), values(), sortByKey()

Actions like : countByKey(), collectAsMap(), lookup(key)

1. Difference between cluster and client mode

Spark application can be submitted in two different ways – cluster mode and client mode. In cluster mode, the driver will get started within the cluster in any of the worker machines. So, the client can fire the job and forget it. In client mode, the driver will get started within the client. So, the client has to be online and in touch with the cluster. So, if the client machine is “far” from the worker nodes then it makes sense to use cluster mode. If our application is in a gateway machine quite “close” to the worker nodes, the client mode could be a good choice.

1. What happens if a worker node is dead

In case if the worker fails, the executors in that worker node will be killed, along with the data in their memory. Using the lineage graph, those tasks will be accomplished in any other worker nodes.

1. How to handle bad records in spark and those types?

While reading a filesystem, spark has a mode called datareadmode that has 3 options: permissive(default), dropmalformed and failfast.

Permissive allows the corrupt data

Dropmalformed removes the corrupt data

Failfast never used, fails execution

You can use a column to capture corrupt data with permissive mode with option columnNameOfCorruptRecord.

Df=spark.read.schema(df\_schema).option(“columnNameOfCorruptRecord”, “\_corrupt\_record”).option(“mode”,”PERMISSIVE”).csv(‘filename’, header=True, inferSchema=True)

OR

Use badRecordsPath option of spark.read to store rejected records in external location

1. How to get all dataframes in spark

From pyspark.sql import DataFrame

Print([k for (k,v) in globals().items() if isinstance(v, DataFrame)])

1. How to track add source file name in one of the columns in dataframe

From pyspark.sql.functions import input\_file\_name

Df=spark.read.csv(“filename”, header=True)

Df.withColumn(“filename”, input\_file\_name())

1. Get number of rows on each file in a dataframe

From pyspark.sql.functions import input\_file\_name

Df=spark.read.csv(“filename”, header=True)

Df.withColumn(“filename”, input\_file\_name()).groupBy(“filename”).count()

1. How to add partitionId in dataframe

From pyspark.sql.functions import spark\_partition\_id

Df=spark.read.csv(“filename”, header=True)

Df.withColumn(“partition\_id”, spark\_partition\_id().select(“partition\_id”).distinct())

1. How to get row count by partitioned in df

From pyspark.sql.functions import spark\_partition\_id

Df=spark.read.csv(“filename”, header=True)

Df.withColumn(“partition\_id”, spark\_partition\_id().select(“partition\_id”).count())

1. How to add sequence generated surrogate key as column in df

From pyspark.sql.functions import monotonically\_increasing\_id

Df.withColumn(“id”, monotonically\_increasing\_id())

OR

Using MD5

Where you can get duplicates

From pyspark.sql.functions import md5

Df.withColumn(“id”, md5(EMPNO))

OR

Using SHA2

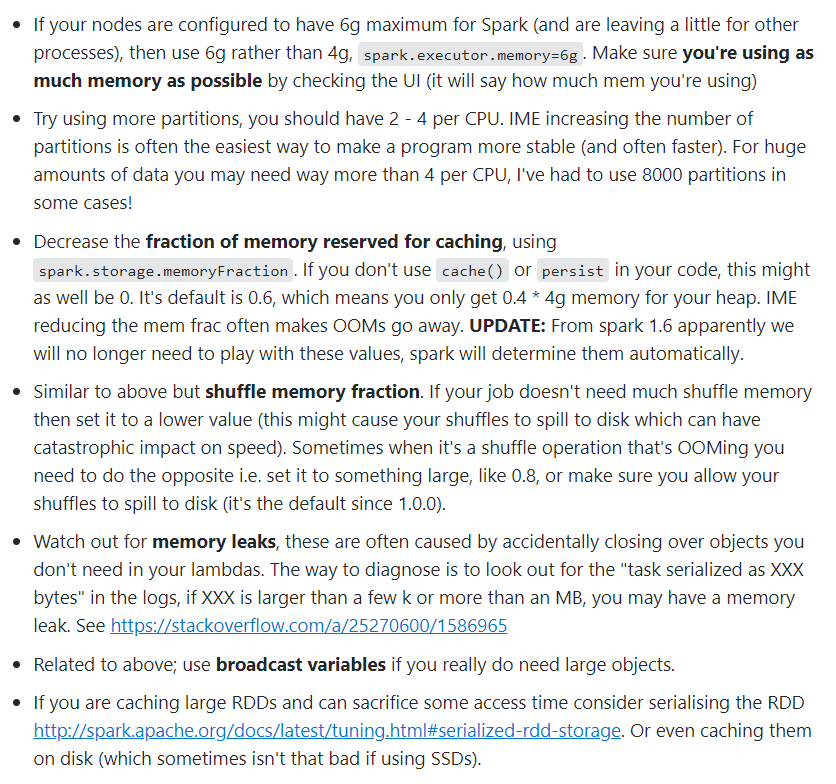
From pyspark.sql.functions import sha2

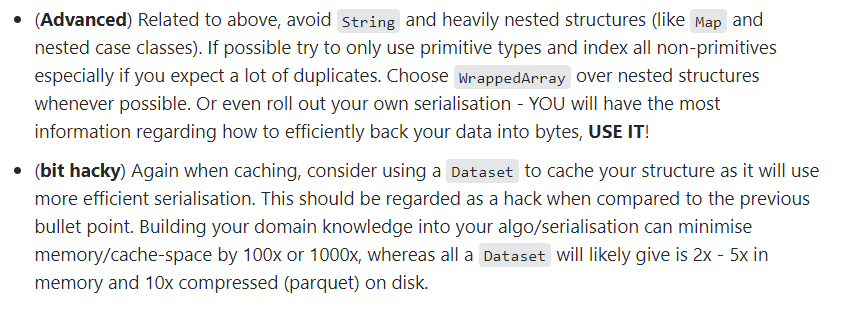
Df.withColumn(“id”, sha2(EMPNO, 256))

1. What is a global temporary view and temporary view.

Global temp view can be shared at server level and temp view at user level

1. How to handle out of memory exception?





1. How do you join spark dataframes?

<https://techvidvan.com/tutorials/apache-spark-terminologies/>

1. What is PySpark SparkContext?

PySpark SparkContext can be seen as the initial point for entering and using any Spark functionality. The SparkContext uses py4j library to launch the JVM, and then create the JavaSparkContext. By default, the SparkContext is available as ‘sc’.

1. Difference between SparkContext and SparkSession

In Spark 1.x, three entry points were introduced: SparkContext, SQLContext and HiveContext. Since Spark 2.x, a new entry point called SparkSession has been introduced that essentially combined all functionalities available in the three aforementioned contexts.

1. How do you create a spark session?

From pyspark.sql import SparkSession

spark = SparkSession.builder.appName(‘App’).getOrCreate()

1. What is the difference between inferSchema and header options in spark?

inferSchema is used to infer the schema of each and every column in the csv file. This will require an extra pass over the csv file and likely reduce the reading speed but will ensure the schema is in line with the csv file being read.

How to do it?

multiline\_df = spark.read.option("multiline","true").option("inferSchema","true").json("test.json")

Header option is used to declare that the csv being read has the first row as header and the headers used in csv become columns of the dataframe, which will be \_c0, \_c1 etc. otherwise.

1. How do you add an id column to an existing dataframe?

from pyspark.sql.functions import monotonically\_increasing\_id

df = df.select("\*").withColumn("id", monotonically\_increasing\_id())

If you want to start from 1

from pyspark.sql.functions import desc, row\_number, monotonically\_increasing\_id

from pyspark.sql.window import Window

df\_with\_seq\_id = df.withColumn('index\_column\_name', row\_number().over(Window.orderBy(monotonically\_increasing\_id())))

1. How do you create sqlContext from sparkSession?

From pyspark.sql import SparkSession

from pyspark import SparkContext

from pyspark.sql import SQLContext

spark = SparkSession.builder.appName().getOrCreate()

sc = spark.sparkContext

sqlContext = SQLContext(sparkContext=sc, sparkSession=spark)

SQLContext needs to be explicitly imported unlike sparkContext which is like an attribute of spark session.

1. How to read a dictionary into a dataframe?

from pyspark.sql import Row

spark.createDataFrame(Row(\*\*x) for x in mylist).show(truncate=False)

1. How do you select specific columns from a dataframe?

From pyspark.sql.functions import \*

Df.select(col(“Emp\_id”)).show()

1. How do you rename a column in a df?

df=df.withColumnRenamed("\_2","Age")

1. How do you cast a column in df to IntegerType

From pyspark.sql.types import IntegerType

df.withColumn("Age",col("Age").cast(IntegerType()))

1. How can you get the count of distinct of a column?

From pyspark.sql.functions import countDistinct

df.select(countDistinct("b")).show()

1. Write code to get the maximum and minimum salary of an employee in a department.

From pyspark.sql.window import Window

From pyspark.sql.functions import \*

windowfunc1=Window.partitionBy(“Dept\_Id”).orderBy(desc(“Sal”))

windowfunc2=Window.partitionBy(“Dept\_Id”).orderBy(asc(“Sal”))

df \

.withColumn(“MaxSalaryInDept”,max(“Sal”).over(windowfunc1)) \

. withColumn(“MinSalaryInDept”,min(“Sal”).over(windowfunc2)) \

orderBy is optional

If partition by is not required,

maxDF = df.select(max(col("Salary").cast(DoubleType())))

df.crossJoin(maxDF).show()

1. Give some examples of data transformations you did in spark
2. Validation of email

from validate\_email import validate\_email

from pyspark.sql.types import BooleanType

from pyspark.sql.functions import udf

valid\_email = udf(lambda x: validate\_email(x), BooleanType())

df.withColumn('is\_valid', valid\_email('EmailAddress')).show()

Created a python udf for validating filename for quarantined files and updated a flag column with the same.

1. Replacing null values with -1 in a few foreign key columns

df=df.fillna("-1")

This is needed when we have tags in xml which are having minOccurs=1 and they still have null values.

1. What are the different datatypes in pyspark.sql?

Null, String, Binary, Boolean, Date, Timestamp, Decimal, Double, Float, Byte, Integer, Long, Short, Array, Map, Struct, StructField.

1. How can you read values from a database and save them in your env variables?

Consider reading values from database with pandas read\_sql module into a pandas df.

Df=read\_sql(‘select key,value from config\_table’,connection)

Now you can read from the dataframe in following 2 ways

1. Using pandas

For ind in Df.index:

Os.environ[Df[‘key’][ind]]=Df[‘value’][ind]

1. Using spark dataframes

Sdf=spark.createDataFrame(Df)

For row in Sdf.rdd.toLocalIterator():

Os.environ[row.key]=row.value

1. How to read avro files

df = spark.read.format("avro").load("examples/src/main/resources/users.avro")

df.select("name", "favorite\_color").write.format("avro").save("namesAndFavColors.avro")

1. How to use parquet file in pyspark?

peopleDF = spark.read.json("examples/src/main/resources/people.json")

# DataFrames can be saved as Parquet files, maintaining the schema information.

peopleDF.write.parquet("people.parquet")

# Read in the Parquet file created above.

# Parquet files are self-describing so the schema is preserved.

# The result of loading a parquet file is also a DataFrame.

parquetFile = spark.read.parquet("people.parquet")

# Parquet files can also be used to create a temporary view and then used in SQL statements.

parquetFile.createOrReplaceTempView("parquetFile")

teenagers = spark.sql("SELECT name FROM parquetFile WHERE age >= 13 AND age <= 19")

teenagers.show()

1. How do you read json files?

Spark.read.json() however, each line must be a valid json object

1. When to use parquet files over avro?

Parquet is a columnar format which means columns are stored in sequential blocks rather than a record, hence retrieving columns at once becomes faster than any row based file format in which all records will be loaded, parsed for columns and then columns will be returned.

1. How do you rename columns of a spark df?

df.withColumnRenamed("dob","DateOfBirth").printSchema()

Or directly in spark.sql use as alias for columns

from pyspark.sql.functions import \*

df.select(col("name.firstname").alias("fname"), \

col("name.middlename").alias("mname"), \

col("name.lastname").alias("lname"), \

col("dob"),col("gender"),col("salary")) \

.printSchema()

Or

from pyspark.sql.functions import \*

df4 = df.withColumn("fname",col("name.firstname")) \

.withColumn("mname",col("name.middlename")) \

.withColumn("lname",col("name.lastname")) \

.drop("name")

df4.printSchema()

or

newColumns = ["newCol1","newCol2","newCol3","newCol4"]

df.toDF(\*newColumns).printSchema()

1. Difference between flatten and explode

Explode will return 1 row for each exploded element from the array

Flatten will combine all elements and return single row.

1. Difference between explode and explode\_outer?

When column to explode contains null/None data, explode will just omit the record, while explode\_outer will keep the record while exploding rest of the data.

1. Difference between pandas and spark explode function?

Pandas explode function is like sparks’ explode\_outer function.

1. What is explode alternative for rdd?

flatMap

1. What are globalTempViews?

These are spark views created by createGlobalTempView function and can be accessed in different session unlike temporary local views

1. How do you read xml into a spark dataframe?

Spark.read.format(“com.databricks.spark.xml”).option().load()

1. How do you write spark dataframe to redshift database ?

Df.write.

format(“com.databricks.spark.redshift”)

.option(“url”,”redshift\_url”)

.option(“dbtable”,”table\_name”)

.option(“driver”, “com.amazon.redshift.jdbc42.Driver”)

.option(“tempdir”,”temp\_out\_dir”)

.option(“aws\_iam\_role”,”role”)

.option(“tempformat”, “CSV GZIP”)

.option(“csv\_separator”, “,”)

.option(“preactions”,”actions to be performed before writin to the table”)

.option(“postactions”, ”actions to be performed after writin to the table”)

.mode(“append”)

.save()

1. What are different modes while writing to redshift db?

Append

Overwrite

Error : Throw an error if data already exists

CreateIfNotExists

1. What option can you use if your S3 bucket and Redshift cluster are in different AWS regions

Use .option(“extracopyoptions”,”region eu-west-1”)

Or

.option(“awsregion”, “eu-west-1”)

1. What is the difference between collect and glom.collect?

When you create an rdd with partitions, collect will coalesce all elements in all partitions into a single list, but glom will collect all elements from all partitions and group them within partitions and return list of list.

Example:

Rdd=sc.parallelize([1,2,3,4],2)

Rdd.collect()

[1,2,3,4]

Rdd.glom().collect()

[[1,2],[3,4]]

1. What to do when you have Long-running Spark query that hangs indefinitely even though the corresponding Redshift operation is done

Use tcpKeepAlive JDBC flag in spark.read.format().options() and TCPKeepAliveMinutes to a low value.

1. What is the default format in which data is written in temp directory while writing to redhisft?

Default is AVRO, but CSV is faster.

If there are bigger files, we can use LZO compression for better performance.

https://www.stitchdata.com/blog/redshift-database-benchmarks-copy-performance-with-compressed-files/

1. How have you used spark dataframes in your project?

We had stored sql queries and config details in the database as key and value columns, to read that we used pandas read\_sql function and stored results in a var lets call it sqldetails.

Post that we used sd=spark.createDataFrame(sqldetails) and then iterated(row) over it to set those values in the environment variables using sd.rdd.toLocalIterator() and os.environ[row.key]=row.value

1. Which property would you use to avoid using show on each dataframe in jupyter?

Set the param spark.sql.repl.eagerEval.enabled to True and set the value of spark.sql.repl.eagerEval.maxNumRows to control the number of rows.

1. How do you fetch rows based on some conditions?

Df.filter(df.a==’abc’)

1. How to create pandas DF from Spark DF?

Sdf=sc.parallelize().toDF()

Pdf=sdf.toPandas()

1. Have you used pandas UDFs?

No, but to use those we need to use the decorator @pandas\_udf or to apply pandas function defined in your program, you can use mapInPandas or applyInPandas on your spark dataframe.

1. Difference between mapInPandas and applyInPandas.

Apply works on grouped data, map yields generators.

1. Difference between merge and merge\_asof

Merge is used to perform inner join merge\_asof for left join and match is done based on nearest keys rather than exact keys

1. How do you read file from S3 bucket?

Using spark.sparkContext.wholeTextFiles()

Or spark.read.text(s3path, wholetext=True).select(input\_file\_name(), “value”).rdd

1. Difference between multiple read functions in spark

sparkContext.textFile() and sparkContext.wholeTextFiles() methods to read into RDD and spark.read.text() and spark.read.textFile() methods to read into DataFrame from local or HDFS file.

1. Can you use python or pandas function as udf in spark sql

Yes, for that you need to register the udf with spark.udf.register

def convertCase(str):

resStr=""

arr = str.split(" ")

for x in arr:

resStr= resStr + x[0:1].upper() + x[1:len(x)] + " "

return resStr

spark.udf.register("convertUDF", convertCase,StringType())

df.createOrReplaceTempView("NAME\_TABLE")

spark.sql("select Seqno, convertUDF(Name) as Name from NAME\_TABLE") \

.show(truncate=False)

Or

Using sqlContext.registerFunction

sqlContext.registerFunction("stringLengthString", lambda x: len(x))

sqlContext.sql("SELECT stringLengthString('test')").collect()

1. DataFrame and Spark SQL share the same execution engine so they can be interchangeably used seamlessly. Give an example of this

Df.createOrReplaceTempView(“tablename”)

Now you can run spark.sql on this tablename.

1. Difference between coalesce and repartition

Repartition can be used to increase or decrease the number of partitions, whereas coalesce can only be used to decrease.

1. How can you sort on a dataframe?

Use df.sort or df.orderby, asc for ascending and desc otherwise. Default is asc.

1. How can you use pyspark sql functions?

By importing **pyspark.sql.functions**

1. What are discretized streams?

Discretized Stream or DStream is the basic abstraction provided by Spark Streaming. It represents a continuous stream of data, either the input data stream received from source, or the processed data stream generated by transforming the input stream.

1. What are all the options available with diff formats while reading in spark?

<https://spark.apache.org/docs/2.0.2/api/java/org/apache/spark/sql/DataFrameReader.html>

1. How do you connect to redshift or aurora?

Using python’s mysql.connector for aurora

And

Psycopg2 for redshift

1. How to handle encryption of password in python?

From base64 import standard\_b64decode

Spark Q&A

1. What all is required to get spark up and running?

Python – used anaconda distribution for that.

JDK 8.0 – spark not available for 9+

Apache spark – from Apache.com

1. Sparkcontext object is automatically created in the pyspark or scala shell.
2. What is Maven?

Maven is a popular package management tool for Java-based languages that lets you link to libraries in public repositories.

1. How do you run python standalone applications for spark?

By using spark-submit script which includes spark dependencies in Python.

1. How do you create a spark session?

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

1. How can you control the number of rows that you can show on a jupyter notebook with pyspark dataframe?

By using variables like :

spark.conf.set('spark.sql.repl.eagerEval.enabled', True)

This will set the eager evaluation to true and when the following value is set to a specific number only those number of lines will be shown.

spark.sql.repl.eagerEval.maxNumRows

1. How to collect the dataframe data from all executors at the driver?

Use df.collect()

1. Convert a spark df to pandas df

Df.toPandas()

1. How can you create a new column in existing df?

Use the withColumn function as below:

df.withColumn('upper\_c', upper(df.c)).show()

1. How to apply conditions to filter data from a dataframe?

df.fliter(df.a == 1).show()

1. How can you run pandas functions directly on your spark dataframes?

Use mapInPandas() function

def pandas\_filter\_func(iterator):

for pandas\_df in iterator:

yield pandas\_df[pandas\_df.a == 1]

sdf.mapInPandas(pandas\_filter\_func, schema=sdf.schema).show()