“What is the role and purpose of SparkContext in PySpark?

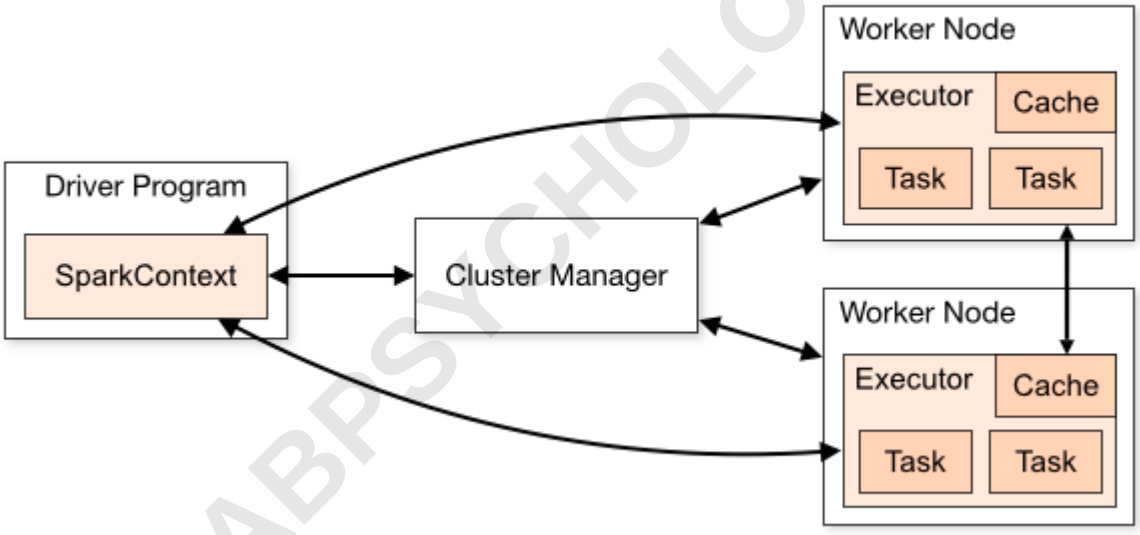
Authored by

stats writer

June 24, 2024

SparkContext is a crucial component of the PySpark framework, responsible for establishing a connection between the application and the Spark cluster. It acts as the entry point for any Spark functionality and manages the execution of jobs on the cluster. Its primary purpose is to coordinate the parallel execution of tasks and distribute data across the cluster efficiently. It also provides access to various Spark features, such as RDDs, DataFrames, and streaming capabilities. In summary, the role of SparkContext is to facilitate the efficient and seamless interaction between a PySpark application and the Spark cluster.

`pyspark.SparkContext` is an entry point to the PySpark functionality that is used to communicate with the cluster and to create an RDD, accumulator, and broadcast variables. In this article, you will learn how to create PySpark SparkContext with examples. Note that you can create only one SparkContext per JVM, in order to create another first you need to stop the existing one using `stop()` method.



Source: spark.apache.org

The Spark driver program creates and uses SparkContext to connect to the cluster manager to submit PySpark jobs, and know what resource manager (YARN, Mesos, or Standalone) to communicate to. It is the heart of the PySpark application.

How to get current SparkContext & its configurations in Spark

1. SparkContext in PySpark shell

Be default PySpark shell creates and provides `sc` object, which is an instance of SparkContext

class. We can directly use this object where required without the need of creating.

```
>>sc.appName
```

Similar to the PySpark shell, in most of the tools, notebooks, and Azure Databricks, the environment itself creates a default SparkContext object for us to use so you don't have to worry about creating a PySpark context.

2. Create SparkContext in PySpark

Since PySpark 2.0, Creating a SparkSession creates a SparkContext internally and exposes the `sparkContext` variable to use.

At any given time only one `SparkContext` instance should be active per JVM. In case you want to create another you should stop existing SparkContext using `stop()` before creating a new one.

```
# Create SparkSession from builder
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local")
.appName('SparkByExamples.com')
.getOrCreate()
print(spark.sparkContext)
print("Spark App Name : "+ spark.sparkContext.appName)
```

Outputs
#<SparkContext master=local appName=SparkByExamples.com>
#Spark App Name : SparkByExamples.com

As I explained in the SparkSession article, you can create any number of SparkSession objects however, for all those objects underlying there will be only one SparkContext.

3. Stop PySpark SparkContext

You can stop the SparkContext by calling the `stop()` method. As explained above you can have only one SparkContext per JVM. If you wanted to create another, you need to shutdown it first by using stop() method and create a new SparkContext.

```
# SparkContext stop() method
```

```
spark.sparkContext.stop()
```

When PySpark executes this statement, it logs the message ***INFO SparkContext: Successfully stopped SparkContext*** to console or to a log file.

When you try to create multiple SparkContext you will get the below error.

ValueError: Cannot run multiple SparkContexts at once;

4. Creating SparkContext prior to PySpark 2.0

You can create SparkContext by programmatically using its constructor, and pass parameters like master and appName at least as these are mandatory params. The below example creates context with a master as `local` and app name as `Spark_Example_App`.

```
# Create SparkContext
from pyspark import SparkContext
sc = SparkContext("local", "Spark_Example_App")
print(sc.appName)
```

You can also create it using `SparkContext.getOrCreate()`. It actually returns an existing active SparkContext otherwise creates one with a specified master and app name.

```
# Create Spark Context
from pyspark import SparkConf, SparkContext
conf = SparkConf()
conf.setMaster("local").setAppName("Spark Example App")
sc = SparkContext.getOrCreate(conf)
print(sc.appName)
```

5. Create PySpark RDD

Once you have a SparkContext object, you can create a PySpark RDD in several ways, below I have used the `range()` function.

```
# Create RDD
rdd = spark.sparkContext.range(1, 5)
print(rdd.collect())
```

```
# Output
#
```

6. SparkContext Commonly Used Variables

`applicationId` - Returns a unique ID of a PySpark application.

`version` - Version of PySpark cluster where your job is running.

`uiWebUrl` - Provides the Spark Web UI url that started by SparkContext.

7. SparkContext Commonly Used Methods

`accumulator`(value) - It creates an pyspark accumulator variable with initial specified value. Only a driver can access accumulator variables.

`broadcast(value)` - read-only PySpark broadcast variable. This will be broadcast to the entire cluster. You can broadcast a variable to a PySpark cluster only once.

`emptyRDD()` - Creates an empty RDD

`getOrCreate()` - Creates or returns a SparkContext

`hadoopFile()` - Returns an RDD of a Hadoop file

`newAPIHadoopFile()` - Creates an RDD for a Hadoop file with a new API InputFormat.

`sequenceFile()` - Get an RDD for a Hadoop SequenceFile with given key and value types.

`setLogLevel()` - Change log level to debug, info, warn, fatal, and error

`textFile()` - Reads a text file from HDFS, local or any Hadoop supported file systems and returns an RDD

`union()` - Union two RDDs

`wholeTextFiles()` - Reads a text file in the folder from HDFS, local or any Hadoop supported file systems and returns an RDD of Tuple2. The first element of the tuple consists file name and the second element consists context of the text file.

8. Conclusion

In this PySpark Context article, you have learned what is SparkContext, how to create it, stop it, and usage with a few basic examples. As you learned SparkContext is an entry point to the PySpark execution engine which communicates with the cluster. Using this you can create a RDD, Accumulators and Broadcast variables.

Related Articles

Reference

Happy Learning !!