**PySpark Broadcast Variables**

In PySpark RDD and DataFrame, Broadcast variables are read-only shared variables that are cached and available on all nodes in a cluster in-order to access or use by the tasks. Instead of sending this data along with every task, PySpark distributes broadcast variables to the workers using efficient broadcast algorithms to reduce communication costs.

**Use case**

Let me explain with an example when to use broadcast variables, assume you are getting a two-letter country state code in a file and you wanted to transform it to full state name, (for example CA to California, NY to New York e.t.c) by doing a lookup to reference mapping. In some instances, this data could be large and you may have many such lookups (like zip code e.t.c).

Instead of distributing this information along with each task over the network (overhead and time consuming), we can use the broadcast variable to cache this lookup info on each machine and tasks use this cached info while executing the transformations.

**How does PySpark Broadcast work?**

Broadcast variables are used in the same way for RDD, DataFrame.

When you run a PySpark RDD, DataFrame applications that have the Broadcast variables defined and used, PySpark does the following.

* PySpark breaks the job into stages that have distributed shuffling and actions are executed with in the stage.
* Later Stages are also broken into tasks
* Spark broadcasts the common data (reusable) needed by tasks within each stage.
* The broadcasted data is cache in serialized format and deserialized before executing each task.

You should be creating and using broadcast variables for data that shared across multiple stages and tasks.

Note that broadcast variables are not sent to executors with sc.broadcast(variable) call instead, they will be sent to executors when they are first used.

**How to create Broadcast variable**

The PySpark Broadcast is created using the broadcast(v) method of the SparkContext class. This method takes the argument v that you want to broadcast.

**In PySpark shell**

broadcastVar = sc.broadcast(Array(0, 1, 2, 3))

broadcastVar.value

**PySpark RDD Broadcast variable example**

Below is a very simple example of how to use broadcast variables on RDD. This example defines commonly used data (states) in a Map variable and distributes the variable using SparkContext.broadcast() and then use these variables on RDD map() transformation.

import pyspark

from pyspark.sql import SparkSession

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

states = {"NY":"New York", "CA":"California", "FL":"Florida"}

broadcastStates = spark.sparkContext.broadcast(states)

data = [("James","Smith","USA","CA"),

("Michael","Rose","USA","NY"),

("Robert","Williams","USA","CA"),

("Maria","Jones","USA","FL")

]

rdd = spark.sparkContext.parallelize(data)

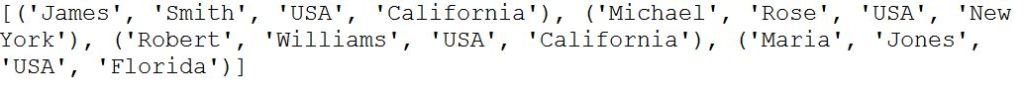
def state\_convert(code):

return broadcastStates.value[code]

result = rdd.map(lambda x: (x[0],x[1],x[2],state\_convert(x[3]))).collect()

print(result)

Yields below output



**PySpark DataFrame Broadcast variable example**

Below is an example of how to use broadcast variables on DataFrame, similar to above RDD example, This also uses commonly used data (states) in a Map variable and distributes the variable using SparkContext.broadcast() and then use these variables on DataFrame map() transformation.

If you are not familiar with DataFrame, I will recommend to learn the DataFrame before proceeding further on this article.

import pyspark

from pyspark.sql import SparkSession

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

states = {"NY":"New York", "CA":"California", "FL":"Florida"}

broadcastStates = spark.sparkContext.broadcast(states)

data = [("James","Smith","USA","CA"),

("Michael","Rose","USA","NY"),

("Robert","Williams","USA","CA"),

("Maria","Jones","USA","FL")

]

columns = ["firstname","lastname","country","state"]

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

df.printSchema()

df.show(truncate=False)

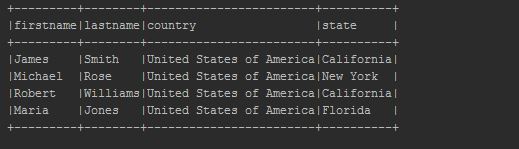
def state\_convert(code):

return broadcastStates.value[code]

result = df.rdd.map(lambda x: (x[0],x[1],x[2],state\_convert(x[3]))).toDF(columns)

result.show(truncate=False)

Above example first creates a DataFrame, transform the data using broadcast variable and yields below output.



You can also use the broadcast variable on the filter and joins. Below is a filter example.

# Broadcast variable on filter

filteDf= df.where((df['state'].isin(broadcastStates.value)))