**PySpark Accumulator with Example**

The PySpark Accumulator is a shared variable that is used with RDD and DataFrame to perform sum and counter operations similar to Map-reduce counters. These variables are shared by all executors to update and add information through aggregation or computative operations.

In this article, I’ve explained what is PySpark Accumulator, how to create, and using it on RDD and DataFrame with an example.

**What is PySpark Accumulator?**

Accumulators are write-only and initialize once variables where only tasks that are running on workers are allowed to update and updates from the workers get propagated automatically to the driver program. But, only the driver program is allowed to access the Accumulator variable using the **value**property.

**How to create Accumulator variable in PySpark?**

Using **accumulator**() from SparkContext class we can create an Accumulator in PySpark programming. Users can also create Accumulators for custom types using AccumulatorParam class of PySpark.

**Some points to note..**

* **sparkContext.accumulator()**is used to define accumulator variables.
* **add()** function is used to add/update a value in accumulator
* **value** property on the accumulator variable is used to retrieve the value from the accumulator.

We can create Accumulators in PySpark for primitive types **int** and **float**. Users can also create Accumulators for custom types using AccumulatorParam class of PySpark.

**Creating Accumulator Variable**

Below is an example of how to create an accumulator variable “**accum**” of type int and using it to sum all values in an RDD.

accum=sc.accumulator(0)

rdd=spark.sparkContext.parallelize([1,2,3,4,5])

rdd.foreach(lambda x:accum.add(x))

print(accum.value) #Accessed by driver

Here, we have created an accumulator variable **accum** using **spark.sparkContext.accumulator(0)** with initial value 0. Later, we are iterating each element in an rdd using foreach() action and adding each element of rdd to accum variable. Finally, we are getting accumulator value using **accum.value**property.

Note that, In this example, rdd.foreach() is executed on workers and accum.value is called from PySpark driver program.

Let’s see another example of an accumulator, this time will do with a function.

accuSum=spark.sparkContext.accumulator(0)

def countFun(x):

global accuSum

accuSum+=x

rdd.foreach(countFun)

print(accuSum.value)

We can also use accumulators to do a counters.

accumCount=spark.sparkContext.accumulator(0)

rdd2=spark.sparkContext.parallelize([1,2,3,4,5])

rdd2.foreach(lambda x:accumCount.add(1))

print(accumCount.value)

**PySpark Accumulator Example**

Below is a complete RDD example of using different accumulators that I was able to run on my environment.

import pyspark

from pyspark.sql import SparkSession

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

accum=spark.sparkContext.accumulator(0)

rdd=spark.sparkContext.parallelize([1,2,3,4,5])

rdd.foreach(lambda x:accum.add(x))

print(accum.value)

accuSum=spark.sparkContext.accumulator(0)

def countFun(x):

global accuSum

accuSum+=x

rdd.foreach(countFun)

print(accuSum.value)

accumCount=spark.sparkContext.accumulator(0)

rdd2=spark.sparkContext.parallelize([1,2,3,4,5])

rdd2.foreach(lambda x:accumCount.add(1))

print(accumCount.value)