Exploring the Power of PySpark: A Guide to Using foreach and foreachPartition Actions

Maximizing Efficiency and Performance in PySpark Jobs through foreach and foreachPartition Actions

Ahmed Uz Zaman Follow 3 min read · Mar 3, 2023

23

Photo by Gabriel Vasiliu on Unsplash

foreach

`foreach` is a PySpark RDD (Resilient Distributed Datasets) action that applies a function to each element of an RDD. It is used to perform some side-effecting operations, such as writing output to a file, sending data to a database, or printing data to the console. The function passed to `foreach` should have a void return type, meaning it does not return anything.

The syntax for ` `foreach` ` is:

```
rdd.foreach(func)
```

Where `rdd` is the RDD on which you want to apply the function `func`.

Example:

```
rdd = sc.parallelize([1, 2, 3, 4, 5])
def print_num(num):
    print(num)
rdd.foreach(print_num)

# OUTPUT
1
2
3
4
5
```

foreachPartition

`foreachPartition` is similar to `foreach`, but it applies the function to each partition of the RDD, rather than each element. This can be useful when you want to perform some operation on a partition as a whole, rather than on each element individually. The function passed to `foreachPartition` should have a void return type.

The syntax for `` `foreachPartition` `` is:

```
rdd.foreachPartition(func)
```

Where `rdd` is the RDD on which you want to apply the function `func`.

Example:

```
rdd = sc.parallelize([1, 2, 3, 4, 5], 2)
def print_partition(iter):
    for num in iter:
        print(num)
rdd.foreachPartition(print_partition)

# OUTPUT
1
2
3
4
5
```

In this example, the RDD is partitioned into two partitions, and the function `print_partition` is applied to each partition. Each partition is printed on a separate line.

Use Cases — foreach and foreachPartition

1. Writing data to external systems: `foreach` and `foreachPartition` are often used to write the output of a PySpark job to an external system such as a file, database, or message queue. For example, you could use `foreach` to write each element of an RDD to a file, or use `foreachPartition` to write each partition to a separate file.
2. Sending data to an external service: Similarly, `foreach` and `foreachPartition` can be used to send data to an external service for

element of an RDD to a web service, or use `foreachPartition` to send each partition to a separate service.

3. Performing custom processing: `foreachPartition` can be useful when you want to perform some custom processing on each partition of an RDD. For example, you might want to calculate some summary statistics for each partition or perform some machine learning model training on each partition separately.
4. Debugging and logging: `foreach` and `foreachPartition` can be used for debugging and logging purposes. For example, you could use `foreach` to print the output of each element to the console for debugging purposes, or use `foreachPartition` to log each partition to a separate file for debugging and monitoring purposes.
5. Performing complex side-effecting operations: Finally, `foreach` and `foreachPartition` can be used to perform complex side-effecting operations that cannot be expressed using built-in PySpark transformations. For example, you could use `foreach` to perform some custom analysis on each element of an RDD, or use `foreachPartition` to perform some complex transformation on each partition.

It's worth noting that `foreach` and `foreachPartition` are actions, meaning they trigger the execution of the computation on the RDD. Therefore, they should be used sparingly, as they can result in significant overhead and slow down the computation. It's usually better to use transformations like `map`, `filter`, and `reduceByKey` to perform operations on RDDs, and only use `foreach` and `foreachPartition` when necessary.

Conclusion

We discussed how to use `foreach` and `foreachPartition` in PySpark and what are some of its use cases and limitations. So use it wisely for your development. Do check out my PySpark 101 series and other articles related to Spark. Follow for more PySpark related content. Happy reading.

Spark | Data Engineering | Software Engineering | Data | Data Science

Written by Ahmed Uz Zaman

3K followers · 15 following

Lead QA Engineer | ETL Test Engineer | PySpark | SQL | AWS | Azure | Improvising Data Quality through innovative technologies | linkedin.com/in/ahmed-uz-zaman/

Follow

No responses yet

A Aaditya Adhikari

What are your thoughts?

More from Ahmed Uz Zaman

Ahmed Uz Zaman

How to use `where()` and `filter()` in a DataFrame with...

Filtering Rows in a Spark DataFrame: Techniques and Tips

Jan 31, 2023 95 1

Ahmed Uz Zaman

PySpark Date & Time Functions: A Comprehensive Guide

10 Essential PySpark Date and Time Functions for Data Analysis

Mar 16, 2023 109 2

In Geek Culture by Ahmed Uz Zaman

Pandas vs PySpark..!

Key differences, when to use either, free resources

Jan 22, 2023 222 2

In Plumbers Of Data Scie... by Ahmed Uz Zam...

Exploring the Different Join Types in Spark SQL: A Step-by-Step...

Understand the Key Concepts and Syntax of Cross, Outer, Anti, Semi, and Self Joins

Feb 3, 2023 91 1

See all from Ahmed Uz Zaman

Recommended from Medium

ıture	Benefit
ːlarative CDC	No complex logic or orchestration needed
ↄ Type 2 Support	Keeps full historical records
ıl-Time Updates	Enables up-to-date dashboards and ML mod
ҳibility in Handling Deletes/Truncates	Meets varying data semantics
ıplified Code	Declarative syntax vs. thousands of lines of c

In Towards Data Engine... by THE BRICK LEAR...

Databricks 2025 : Simplifying Real-Time Data Pipelines with ...

In today's world of ever-changing data, Change Data Capture (CDC) is no longer a...

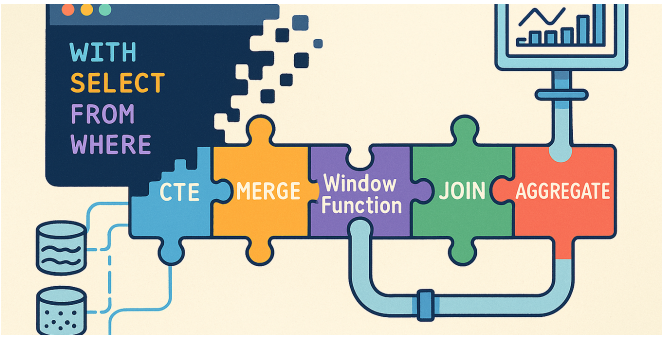
Jun 13 5

Victor Oketch Sabare

Stop Learning Pandas: The Spark 4.0 Features Nobody's Telling Y...

Apache Spark 4.0 finally lets Python-first data folks skip the "collect-to-Pandas"...

Jun 10 25 2



The Data Engineering Digest

SQL for Data Engineering: 10 Advanced Tricks Every Enginee...

From ETL Magic to Performance Power—with Real-Life Analogies and Examples

Jun 16 21 3

Ganesh Chandrasekaran

Databricks: Using Python logging module in notebooks

One common problem many developers face when using the Python logging librar...

Jan 12 30 2

Sriw World of Coding

🚀 Processing 10 TB Data in Apache Spark in 10 Minutes: Ho...

🔍 Introduction

May 6 39 2

In Python in Plain English by Siddharth Ghosh

How Can You Optimize Spark SQL Queries?—An Interview Guide f...

Whether you're getting ready for a big data engineering interview or trying to improve...

Jun 14

See more recommendations