

Directed Works TD 3 – Working with RDDs and Transformations in Apache Spark

Introduction – Reminder

Apache Spark is a distributed computing framework that processes data in memory. It uses a Directed Acyclic Graph (DAG) to represent all transformations before execution.

Question 1: Explain in your own words what lazy evaluation means in Spark:

Question 2: Fill in the blanks:

In Spark, transformations are _____ (they don't execute immediately), while actions are _____ (they trigger the execution).

Creating RDDs

Spark can create RDDs from existing collections or external datasets.

Example:

```
data = [10, 20, 30, 40]
rdd = spark.sparkContext.parallelize(data)
```

Question 3: If we execute the line `rdd.count()`, what happens internally?

- Spark immediately creates all partitions.
- Spark builds a DAG and executes it because `count()` is an action.
- Spark executes line by line without optimization.

Transformations

Transformations are operations applied on RDDs that create new RDDs. They are lazy, meaning they only define the lineage of operations.

Transformation	Description	Example
<code>map()</code>	Applies a function to each element	<code>rdd.map(lambda x: x*2)</code>
<code>filter()</code>	Keeps elements that satisfy a condition	<code>rdd.filter(lambda x: x > 20)</code>
<code>flatMap()</code>	Similar to map but can return multiple results per input	<code>rdd.flatMap(lambda x: x.split(" "))</code>
<code>reduceByKey()</code>	Merges values with the same key	<code>rdd.reduceByKey(lambda a,b: a+b)</code>

Question 4: Complete the following sentence:

`map()` always produces _____ output element(s) per input, while `flatMap()` can produce _____.

Actions

Actions trigger the actual execution of the DAG.

Examples:

- `collect()` : returns all elements to the driver
- `count()` : returns number of elements
- `first()` : returns the first element
- `take(n)` : returns the first n elements

- `saveAsTextFile(path)` : saves results to storage

Question 5: When would you not use `collect()`?

.....

Mini-Exercise :

We have a dataset representing student grades:

Student	Subject	Grade
Ali	Math	15
Sarah	Physics	17
Ali	Physics	13
John	Math	12
Sarah	Math	14

We represent this as:

```
grades = [  
  ("Ali", "Math", 15),  
  ("Sarah", "Physics", 17),  
  ("Ali", "Physics", 13),  
  ("John", "Math", 12),  
  ("Sarah", "Math", 14)  
]
```

Tasks (conceptual):

- Create an RDD from the list above: `rdd = spark.sparkContext.parallelize(grades)`
- Transform the RDD to get (student, grade) pairs
- Use `reduceByKey()` to compute the average grade per student (conceptually).

Question 6: If one node fails during computation, how does Spark recover the missing data?

.....

Reflection

Question 7: Explain the difference between RDD persistence and re-computation:

.....

Question 8: Why is `reduceByKey` preferred over `groupByKey` in most cases?

.....

Optional Challenge

Given the RDD:

```
data = ["big data", "spark data", "big analytics", "data processing"]
```

Write the sequence of transformations (in plain text, not code) that counts the frequency of each word.