Ministry of Higher Education and Scientific Research
University of Abdelhafid Boussouf - Mila
Institute of Mathematics and Computer Science
Department of Computer Science
Master 2 I2A – Big Data
2025/2026

TD 3 – Introduction to Apache Spark

Objectives

Understand how Spark works compared to Hadoop, the concept of Resilient Distributed Dataset (RDD)s, and the logic of transformations and actions in Spark.

Introduction

In previous sessions, we studied Hadoop and the MapReduce model, which processes data by reading it from disk for each operation. While this approach is reliable, it can be slow because it constantly writes and reads intermediate results.
To overcome these limitations, Apache Spark was created. Spark keeps most data in memory, allowing much faster computations, especially for iterative algorithms and real-time analytics.
Spark introduces a new concept: the RDD: a distributed collection of data that can be processed in parallel across multiple nodes.

Part 1 – Concept Review

Before diving into RDDs and transformations, let's compare Spark to Hadoop and ensure we understand the new terminology.

Q1. Compare Hadoop MapReduce and Apache Spark according to the following aspects:

Aspect	Hadoop MapReduce	Apache Spark
Execution Model		
Speed and Performance		
Memory Management		
Ease of Programming		

Q2. Complete the following sentences carefully:

1. Spark computations are based on a fundamental structure called the _____, which stands for _____. It represents a distributed dataset that can be manipulated through transformations and actions.
2. Unlike traditional programming models, Spark uses _____ evaluation, meaning that transformations are not executed immediately but are only computed when an _____ is called.
3. The series of transformations applied to an RDD form a logical plan called a _____ graph, which Spark uses to optimize and recover data in case of failure.
4. When Spark runs on a cluster, data is divided into _____, and each partition is processed in _____ on different nodes.

Part 2 – Understanding RDD Transformations

RDDs support two main categories of operations:

- **Transformations:** operations that create a new RDD from an existing one (e.g., map, filter, flatMap, reduceByKey).
- **Actions:** operations that trigger computation and return a result to the driver program (e.g., collect, count, saveAsTextFile).

Analyze the following code step by step:

```
rdd = sc.parallelize([1, 2, 3, 4, 5, 6])
rdd2 = rdd.map(lambda x: x * 2)
rdd3 = rdd2.filter(lambda x: x > 6)
result = rdd3.collect()
```

Q3. Draw the data flow diagram (lineage graph) showing how data moves from one RDD to another.
Q4. Determine what happens at each stage (contents of rdd2, rdd3, result).
Q5. Identify transformations vs. actions and explain why only actions trigger execution.

📊 Part 3 – Working with Key-Value Pairs

Many real-world problems deal with key-value data, such as word counts or aggregating sales per region. Spark provides pair RDDs that allow us to apply transformations like reduceByKey or groupByKey.

```
data = sc.parallelize(['apple', 'banana', 'apple', 'orange', 'banana', 'apple'])
pairs = data.map(lambda x: (x, 1))
counts = pairs.reduceByKey(lambda a, b: a + b)
output = counts.collect()
```

Q6. Write the intermediate contents of pairs.
Q7. Explain how reduceByKey works.
Q8. Predict the final output of output.
Bonus: Explain RDD lineage and how Spark recovers lost partitions.

💭 Part 4 – Reflection and Discussion

Q9. Why does Spark use lazy evaluation?
Q10. Two advantages of Spark's lineage graph compared to Hadoop's intermediate files.
Q11. When might Hadoop MapReduce be preferable to Spark?