DASCI

Instituto Andaluz Interuniversitario
en Ciencia de Datos e
Inteligencia Computacional

Ciencia de Datos a través del Big Data

Diego García (djgarcia@ugr.es)
Isaac Triguero (isaaktriguero@ugr.es)

Financiado por
la Unión Europea
NextGenerationEU

GOBIERNO
DE ESPAÑA

MINISTERIO
PARA LA TRANSFORMACIÓN DIGITAL
Y DE LA FUNCIÓN PÚBLICA

SECRETARÍA DE ESTADO
DE DIGITALIZACIÓN
E INTELIGENCIA ARTIFICIAL

Plan de
Recuperación,
Transformación
y Resiliencia

UNIVERSIDAD
DE GRANADA

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

REPSOL

CAMBRIDGE UNIVERSITY PRESS

Chapter 5

Spark SQL I

Learning Outcomes

- The importance of structured data to optimize Big Data processing [KU]
- The concept of DataFrames to work with structured data in Spark [KU]
- How to operate with DataFrames: understanding the differences with RDDs [KU, IS, PPS]
- How to use DataFrames in practice for problem solving [IS, PPS]

Outline

- Recap – Spark RDDs
- Brief intro to Spark SQL
  - Introduction and motivation
  - DataFrames and Datasets
  - Internal working

Recap

- **RDDs API**
  - Transformations vs actions
  - RDDs of key-value pairs
  - RDD lineage – fault tolerance for RDDs
  - Caching RDDs
  - Driver vs workers – operations are happening in the workers

SparkSQL: Introduction and motivation

CAMBRIDGE UNIVERSITY PRESS

- **RDDs are characterized by:**
  - **Dependencies** between them
  - We act on **Partitions** of the data
    - We learnt that Spark optimizes the execution of the parallel tasks on RDDs, but this is only possible for **narrow transformations**
  - We apply **functions to a partition** and return another: Partition => Iterator[T]
    - Transformations are 'black-boxes' for Spark
    - **T is also unknown for Spark**
    - They are very general, which makes them difficult to be optimized!

SparkSQL: Introduction and motivation

- **Spark SQL**
  - **Key Idea**: Impose structure to the data to optimize the execution
  - This means to organise the data – similar to Pandas in Python
  - Spark SQL is a module to process structured data
  - It allows further optimisation as Spark knows more about the underlying data
    - However, RDDs are still being used underneath!
  - We can apply SQL queries

Limiting what you can express allows developers to perform optimizations

SparkSQL: DataFrames and Datasets

CAMBRIDGE UNIVERSITY PRESS

- **Structured APIs**
  - **DataFrames (>= Spark 1.3)**
    - Idea: **RDDs of rows with columns that can be accessed by their names**
    - Similar to **Pandas** in Python (dataframes in R)
    - **Avoid Java serialization** performed by RDDs
    - API natural for developers familiar with building query plans (SQL)
    - Introduced as a part of **Tungsten** project
      - Efficient memory management
    - Concept of **schema** to describe data

SparkSQL: DataFrames and Datasets

CAMBRIDGE UNIVERSITY PRESS

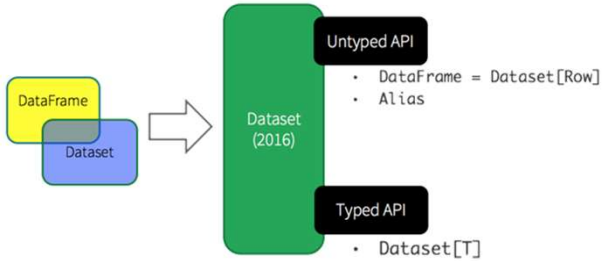
- Structured data vs. **Non-structured data**: DataFrames vs RDDs
  - **RDDs**
    - **Advantages**
      - Easy to understand and very **flexible**
      - Object-oriented programming (e.g., `data.map()`)
      - **Compile-time type-safety** (obviously not in Python!)
    - **Main weaknesses**
      - **Performance in some cases**
      - Data distribution over network or storing implies → **serialisation**
        - Java Serialization or Kryo
        - Overhead for each object (storage and send it)
        - Overhead of the Garbage Collector

SparkSQL: DataFrames and Datasets

CAMBRIDGE UNIVERSITY PRESS

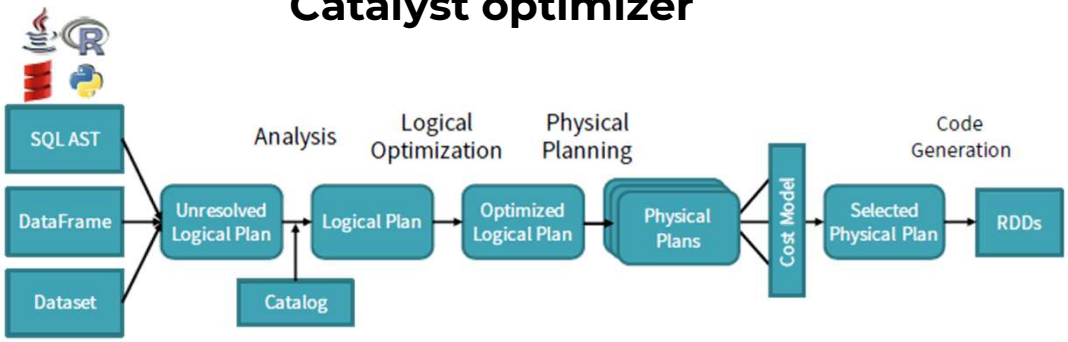
- **Structured APIs**
  - **Datasets (>= Spark 1.6)**
    - Idea: **Strongly typed RDDs**
    - **Functional transformations** (`map`, `flatMap`, `filter`)
    - **Best of both RDDs and DataFrames**
      - Object-oriented programming
      - Compile-time type safety
      - Catalyst optimisation
      - Tungsten off-heap memory optimisation
    - Only for Scala and Java

SparkSQL: DataFrames and Datasets

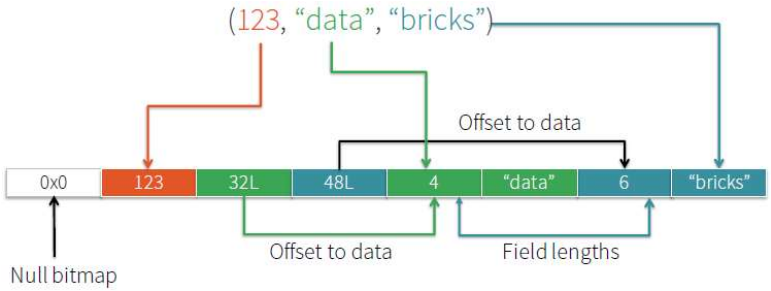


- **Structured APIs**
  - **DataFrames and Datasets**
  - **Fused in Spark 2.0** (November 2016)
    - A DataFrame is just a Dataset of Rows: Dataset[Row]
  - Both make use of Catalyst and Tungsten projects

Catalyst optimizer



Tugnsten memory optimization



Source: [Databricks]

SparkSQL: RDDs vs DataFrames

CAMBRIDGE UNIVERSITY PRESS

- **Structured data vs. Non-structured** data: DataFrames vs RDDs
  - RDDs

```
rdd.filter(lambda x: x.age > 21)
```

  - DataFrames

SQL style

```
df.filter("age > 21");
```

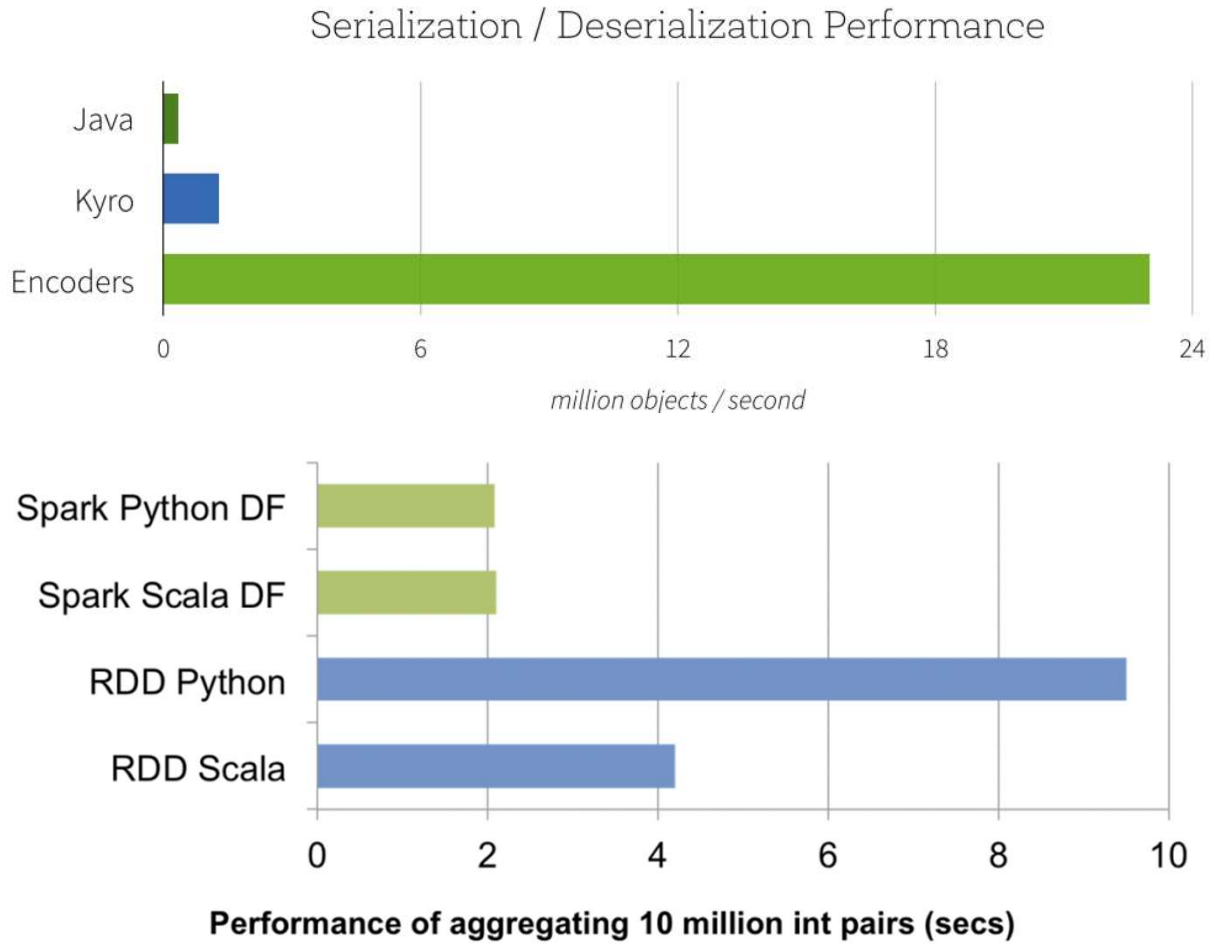
OOP style

```
df.filter(df.col("age").gt(21));

df.filter(df.col("age") > 21);
```

With Catalyst, all is translated to the same Java code, independently of the language

SparkSQL: RDDs vs DataFrames



Source: Databricks

SparkSQL: Internal working

- **Optimization**
  - With **lambda functions** Spark only knows the input and the output
    - It doesn't have information about the computation happening in the workers
  - **Working by columns**
    - Spark can optimize the internal working

	Functions	Columns
You type	`lambda x: x == 1`	`col["x"] == 1`

Spark understands...	`class $anonfun$1 {`
	`  def apply(Int):  Boolean`
	`}`
		`EqualTo(x, Lit(1))`
		OPTIMIZABLE

	A function that takes an integer a returns a boolean (I don't know what's going on inside).	*There is a comparison between the value of a specific variable/attribute and a one.*

SparkSQL: Internal working

CAMBRIDGE UNIVERSITY PRESS

- **Dealing with complex column with functions**
  - Spark provides more than 100 optimized functions and its own SQL types.
    - String manipulation: concat, format_string, lower, lpad
    - Data/Time: current_timestamp, date_format
    - Math: sqrt, randn
    - …
  - You should **always try to use them** to take advantage of the optimizations provided by the projects Tungsten and Catalyst

```
from pyspark.sql.types import * # import data types from Spark
import pyspark.sql.functions as sql_f # import SQL functions
```

http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#module-pyspark.sql.functions

SparkSQL: Summary

CAMBRIDGE UNIVERSITY PRESS

- **Structured data vs. Non-structured** data: DataFrames vs RDDs
  - **RDDs**
    - Allow us to decide **HOW** to do operations
    - Limits the optimisation that Spark can do
  - **DataFrames/Datasets**
    - Allow us to define **WHAT** we want to do
    - Let Spark decides how to do it
- DataFrames have become the primary Machine learning API for Spark
  - RDDs won't be deprecated
  - DataFrames are built on top of RDDs and are interoperable

Take-home message

- RDDs are black-boxes, limiting optimizations
- Imposing a structure allows for further optimizations: DataFrames and Dataset APIs powered by tungsten and catalyst projects

Next lecture:

- Spark SQL – hands on

CAMBRIDGE UNIVERSITY PRESS

Chapter 5

Spark SQL I