



POLITECNICO
MILANO 1863

Apache Spark

Alessandro Margara

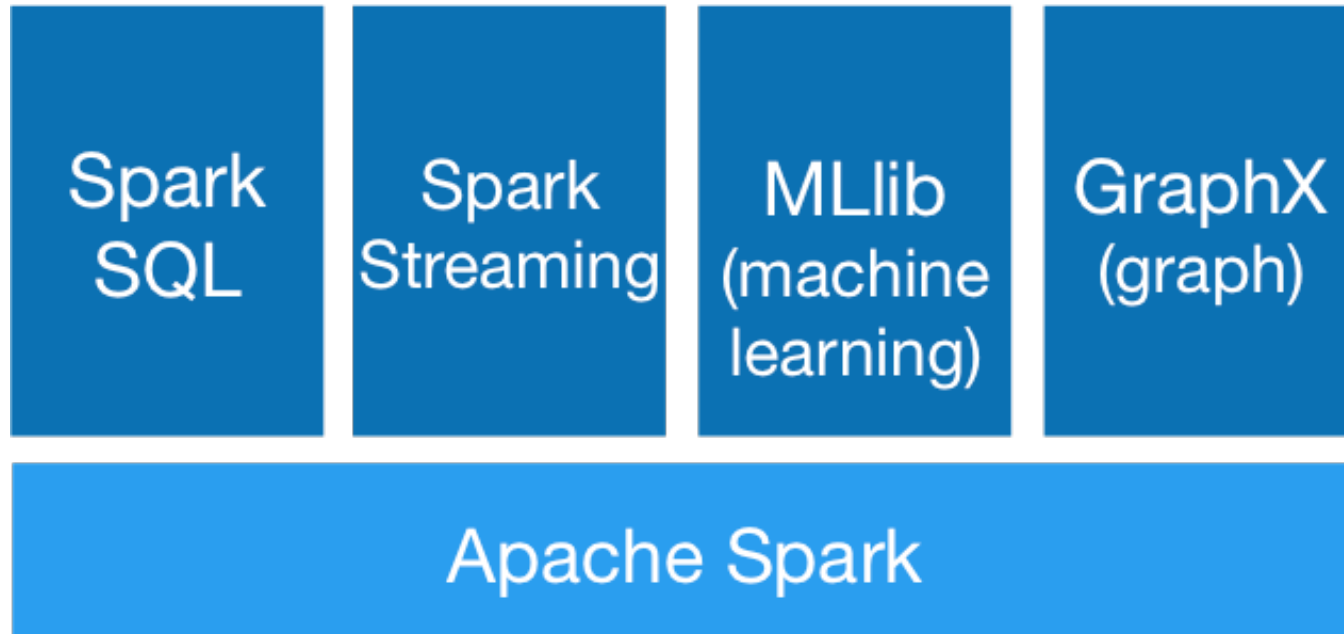
alessandro.margara@polimi.it

<https://margara.faculty.polimi.it>

Spark SQL

SQL, Datasets, and DataFrames

Spark SQL



Spark SQL

- Spark SQL is a Spark module for structured data processing
 - Datasets have a known schema that determines the number, type (and names) of the fields in each data element
 - The API relies on the schema of the datasets under analysis
- The presence of a schema brings twofold advantages
 - Higher-level, declarative API, derived from established database languages
 - Pre-defined operators offer more opportunities for the engine to optimize the computation

Spark SQL

- Spark SQL offers different API/languages
 - SQL
 - DataFrame API
 - Dataset API
- The same Spark SQL engine is used, independently from the API/language adopted
 - It is possible to switch between different APIs depending on the specific needs of the application

SQL API

- The SQL API enables developers to execute standard SQL queries
 - From a programming language
 - From command-line
 - Over JDBC/ODBC
 - ...

Dataset and DataFrames

- Dataset is an interface that provides the level of abstraction of RDDs ...
 - Transformations and actions over large collections
- ... with some additional benefits from Spark SQL optimized execution engine
 - Datasets use a specialized encoder to serialize the objects, processing them, and transmitting over the network
 - By knowing the format, Spark can perform many operations without deserializing the object
 - Filtering, sorting, hashing, ...

Dataset and DataFrames

- DataFrames are a special type of DataSets
 - Implemented as `DataSet<Row>`
 - Represent relational tables, where elements are rows with fixed and known schema
 - Offer the same abstraction as SQL

Spark SQL: SparkSession

- The Spark SQL module exposes its functionalities through the SparkSession class
 - Like SparkContext for the core Spark API
 - Obtained using a builder

```
SparkSession spark = SparkSession
    .builder()
    .appName("App name")
    .master("local")
    .config("option name", "option value")
    .getOrCreate();
```

Spark SQL: DataFrame creation

- `SparkSession.read` provides various sources
 - They create DataFrames

```
spark.read.json("file")
```

```
spark.read.csv("file")
```

```
spark.read.textFile("file")
```

```
spark.read.jdbc(...)
```

Spark SQL: DataFrame creation

- DataFrames can also be created from RDDs
 - By programmatically defining the schema
 - By inferring the schema using reflection

```
List<StructField> fields = new ArrayList<>();  
  
fields.add(DataTypes.createStructField  
    ("name", DataTypes.StringType, false));  
  
fields.add(DataTypes.createStructField  
    ("age", DataTypes.IntegerType, true));  
  
StructType schema = DataTypes  
    .createStructType(fields);
```

Spark SQL: DataFrame

- It is possible to print the content of a DataFrame on the standard output

```
df.show();
```

- Textual representation of the table, ...
- ... very useful for testing and debugging
 - Display the results of transformation on a small dataset

Spark SQL: DataFrame operations

- DataFrames include additional information on their content (e.g., names of columns)
 - Enables more declarative processing
 - Simplified column access using the column name

```
df.filter(col("age").gt(18))  
  .select("name", col("salary") + 10)
```

Spark SQL: DataFrame operations

- Spark SQL also accepts SQL queries as strings

```
employeeDataFrame  
    .createOrReplaceTempView("employee");
```

```
val resultDF = spark.sql  
(  
    "SELECT *  
      FROM employee  
     WHERE salary < 100");
```

Spark SQL: DataFrame operations

- Spark SQL supports all operators coming from the relational algebra
 - selection, projection, filter, join, ...
- It supports common aggregations
 - `count()`, `countDistinct()`, `avg()`, `sum()`, `min()`, `max()`
- It enables developers to write and use their own custom aggregations
 - Extending the `UserDefinedAggregateFunction` class

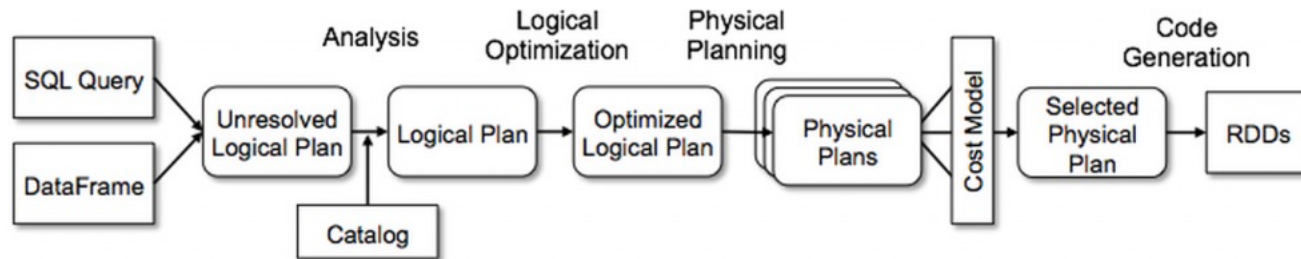
Spark SQL: engine

- Spark SQL can be seen as a scalable relational processing engine ...
- ... to execute expensive analytical queries in a distributed environment
- Developers express relational queries
 - Using SQL / DataFrame / DataSet API
- Queries are compiled to classic operations on RDDs
 - They undergo various optimization steps in the process

Spark SQL: declarative API

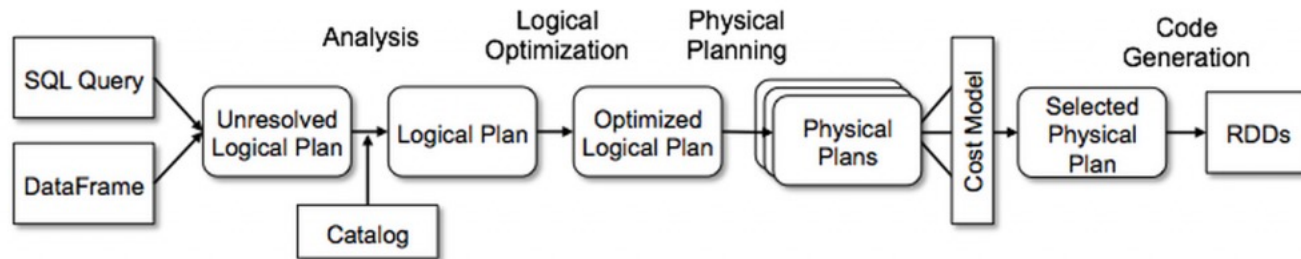
- SQL is totally declarative (queries are strings)
 - Pro: automated optimization
 - Cons: reduced flexibility, syntax errors detected at runtime
- DataFrames are equivalent to SQL, but defined programmatically
 - Pro: automated optimization, syntax errors detected at compile time
 - Cons: reduced flexibility
- Datasets are typed: not Rows, but structures
 - Pro: syntax errors detected at compile time, more flexibility with user-defined lambdas
 - Cons: possibilities for optimizations reduced by the presence of user-defined functions

Spark SQL: optimization



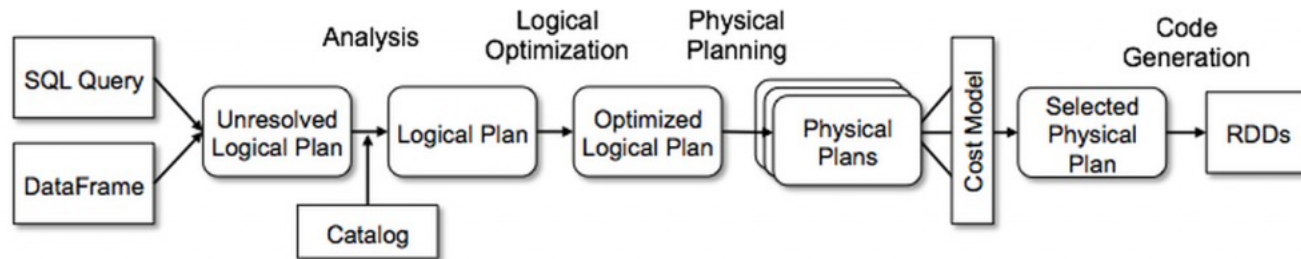
- Logical plan: the engine optimizes the query plans based on well known techniques and heuristics from DBMSs
 - Column pruning, join reordering, filter push down, ...

Spark SQL: optimization



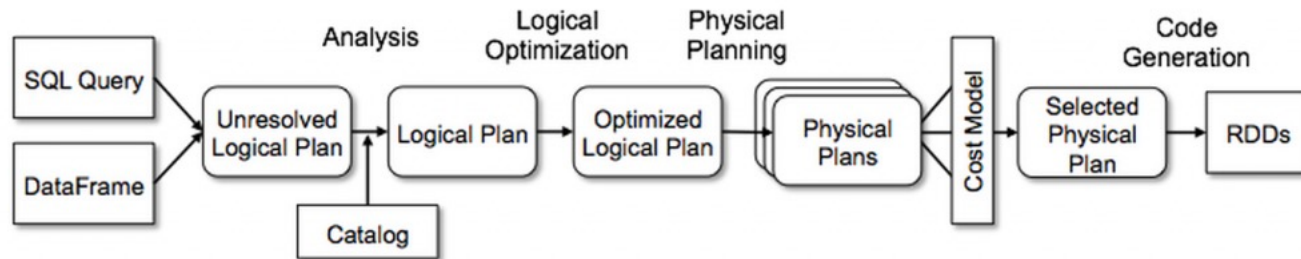
- Physical plans
 - From “what to do” to “how / where to do” it
 - E.g., a join can translate to a broadcast hash join or to a sort merge join

Spark SQL: optimization



- Out of many physical plans one is selected based on the minimization of a cost function
 - The expected cost of a plan is estimated based on the cardinality of the input tables, distribution of data, operators, filtering conditions, ...

Spark SQL: optimization



- Code generation: translates the physical plan into optimized JVM / Spark code
 - Whole stage code generation converts an entire stage into simplified JVM code
 - Adopts binary data format efficient for computation and memory
 - E.g., if a table is persisted / cached, it is stored in columnar format, possibly compressed

Spark SQL: abstraction

- Due to code generation, understanding the execution can be difficult
- Spark offers a SQL visualization tool with the details of the physical plan that is translated to jobs and stages
 - Comparing the two can offer a better insight on the execution and help locating possible bottlenecks