Learning Spark

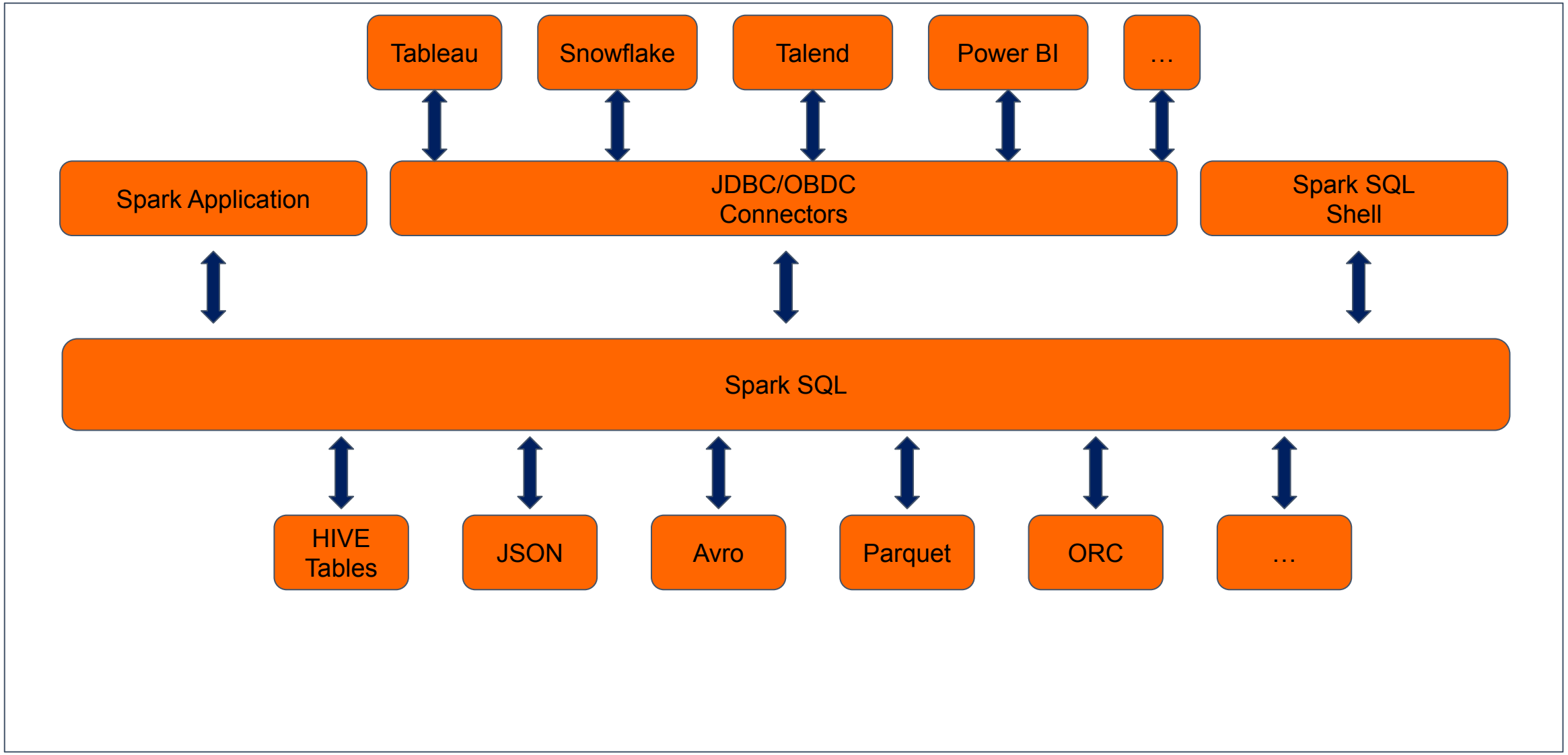
Introduction to Built-in Data Sources

💻 Master's degree, Universitat de Lleida

/bluetab

an IBM Company

Spark SQL connectors and data sources



SQL Tables and Views

- Tables hold data
- Associated with each table in Spark is its relevant metadata, which is information about the table and its data: the schema, description, table name, database name, column names, partitions, physical location where the actual data resides, etc.
- All this is stores in a central metastore.
- Instead of having a separe metastore for Spark tables, Spark by default uses the Apache Hive metastore, located ate /users/hive/warehouse, to persist all the metadata about our tables.
- We can change the default location by setting the Spark config variable spark.sql.warehouse.dir to another location (local or external distributed)
- Spark allows us to create two types of tables (managed or unmanaged)
  - Managed:
    - Spark managed both the metadata and the data in the file store (local FS, HDFS or object store as Amazon S3 or Azure Blob)
    - a SQL command such as DROP TABLE table_name deletes both the metadata and the data
  - Unmanaged:
    - Spark only manages the metadata, while we manage the data ourself, in an external data source such as Cassandra.
    - a SQL command such as DROP TABLE table_name deletes only the metadata, not the actual data

SQL Tables and Views

SQL Database and Tables

- Tables reside within a database.
- By default, Spark creates tables under the default database.
- To create our own database name, we can issue a SQL command from Spark application or notebook.

Creating Views

- In addition, Spark can create views on top of existing tables.
- Can be global (visible across all Spark Sessions on a given cluster) or session-scoped (visible only to a singles SparkSession), and they are temporary.
- Has similar syntax to creating tables within a database.
- Once we create a view, we can query it as a table.
- A temporary view is tied to a single SparkSession within SparkApplication
- Global temporary view is visible across multiple SparkSessions within a Spark Application.

SQL Tables and Views

Viewing Metadata

- Spark manages the metadata associated with each managed or unmanaged table.
- This is captured in the Catalog, a high-level abstraction in Spark SQL for storing metadata.
- LET'S CODE

Data Sources for DataFrames and SQL Tables

- We can interact to Data Sources from Spark through the Data Sources API (DataFrameReader and DataFrameWriter)
- DataFrameReader is the core construct for reading data from a data source into a DataFrame.
  - Recommended pattern of usage:

```
DataFrameReader.format(args).option("key", "value").schema(args).load()
```

  - To instantiate a DataFrameReader, we do it from the SparkSession (“spark.read”)
- DataFrameWriter saves or writes data to a specified built-in data source.
  - Recommended usage patterns examples:

```
DataFrameWriter.format(args)
  .option(args)
  .bucketBy(args)
  .partitionBy(args)
  .save(path)

DataFrameWriter.format(args).option(args).sortBy(args).saveAsTable(table)
```

  - To instantiate a DataFrameWriter, we do it from the DF (“df.write(...)…”)

Data Sources for DataFrames and SQL Tables

Parquet

- We know some typical data sources (files formatted as csv or json, RDBMS such as PostreSQL, No-SQL DB as MongoDB…) but Parquet is the default data source in Spark
- Supported and widely used by many big data processing frameworks and platforms
- Open source columnar file format that offers many I/O optimizations
-

```
_SUCCESS
_committed_1799640464332036264
_started_1799640464332036264
part-00000-tid-1799640464332036264-91273258-d7ef-4dc7-<...>-c000.snappy.parquet
```

- It’s not needed to specify the schema, because it’s in the metadata of parquet

Data Sources for DataFrames and SQL Tables

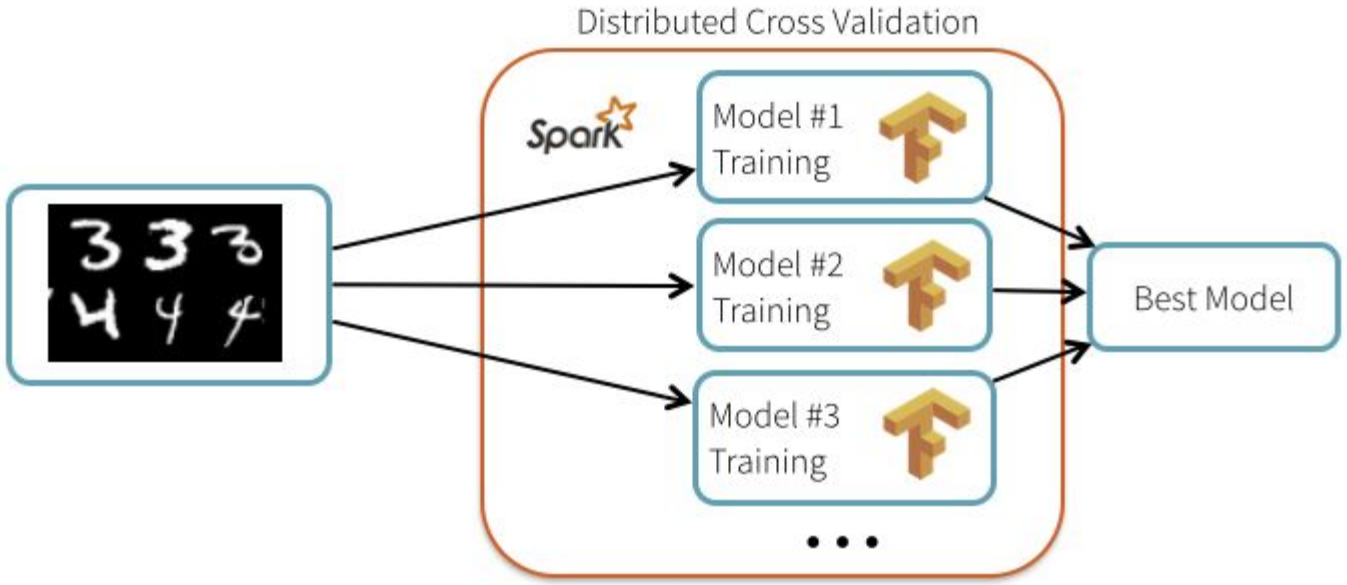
Avro

- Introduced in Spark 2.4
- Avro format is used, for example, by Apache Kafka for message serializing and deserializing
- It offers many benefits:
  - Direct mapping to JSON
  - Speed and efficiency
  - Available for many programming languages

Data Sources for DataFrames and SQL Tables

Images

- Introduced in Spark 2.4 by the community
- To support deep learning and machine learning frameworks such as TensorFlow and PyTorch
- YES, WE CAN INTEGRATE DL FWs WITH SPARK



Data Sources for DataFrames and SQL Tables

Binary Files

- Introduced in Spark 3.0
- The DataFrameReader converts each binary file into a single DataFrame row (record) that contains the raw content and metadata of the file
- The binary file data source produces a DataFrame with the following columns
  - path: StringType
  - modificationTime: TimestampType
  - length: LongType
  - content: BinaryType

/bluetab
an IBM Company

¡Gracias!

alba.lamas@bluetab.net

¡Síguenos!

https://bluetab.net/

https://www.linkedin.com/company/bluetab/