Apache Spark SQL - DataFrames

DVS Technologies, Opp to Home Town, Beside Biryani Zone, Marathahalli, Bangalore
Web : www.dvstechnologies.in | Ph : 080-4209 1111 | Mob: 8892499499, 8123001123

3 V’s of Big data

- **Volume**
  - TB’s and PB’s of files
  - Driving need for batch processing systems
- **Velocity**
  - TB’s of stream data
  - Driving need for stream processing systems
- **Variety**
  - Structured, semi structured and unstructured
  - Driving need for SQL, graph processing systems

Why care about structured data?

- Isn’t big data is all about unstructured data?
- Most real world problems work with structured / semi-structured data 80% of the time
- Sources
  - JSON from API data
  - RDBMS input
  - NoSQL db inputs
- ETL process convert from unstructured to structured

Structured data in M/R world

- Both structured and unstructured data treated as same text file
- Even higher level frameworks like Pig/Hive interprets the structured data using user provided schema
- Let’s take an example of processing csv data in spark in Map/Reduce style.
- Ex: CsvInRDD

Challenges of structured data in M/R

- No uniform way of loading structured data, we just piggyback on input format
- No automatic schema discovery
- Adding a new field or changing field sequencing is not that easy
- Even Pig JSON input format just limits for record separating
- No high level representation of structured data even in Spark as there is always RDD[T]
- No way to query RDD using sql once we have constructed structured output

Spark SQL library

- **Data source API**

  Universal API for Loading/Saving structured data

- **DataFrame API**

  Higher level representation for structured data

- **SQL Interpreter and optimizer**

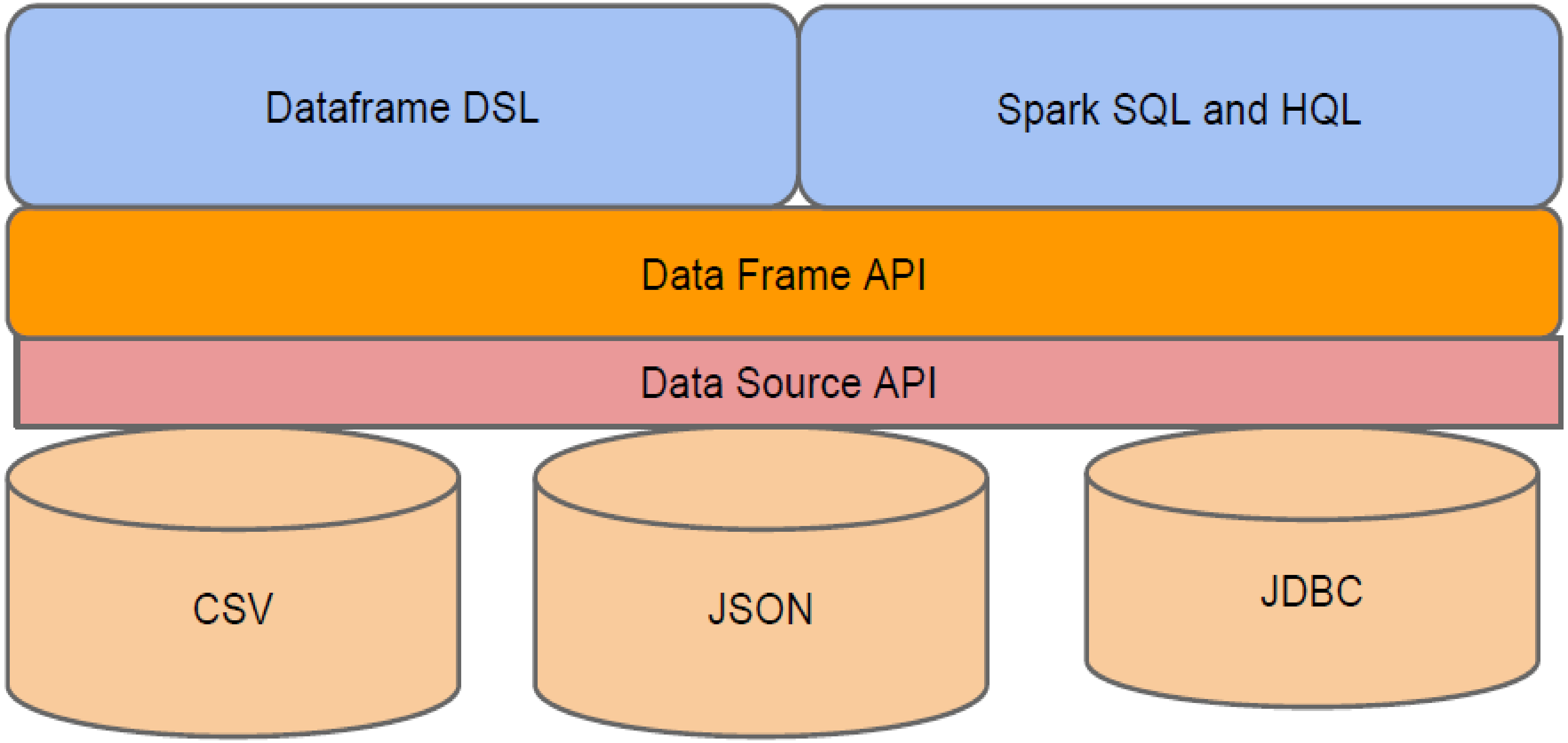
  Express data transformation in SQL

- **SQL service**

  Hive thrift service

Spark SQL	Apache Hive
Library	Framework
Optional metastore	Mandatory metastore
Automatic schema inference	Explicit schema declaration using DDL
API - DataFrame DSL and SQL	HQL
Supports both Spark SQL and HQL	Only HQL
Hive Thrift server	Hive thrift server

Architecture of Spark SQL



Data source API

- Universal API for loading/saving structured data
- Built in support for Hive, Avro, Json, JDBC, Parquet
- Third party integration through spark-packages
- Support for smart sources
- Third parties already supporting
  - CSV
  - MongoDB
  - Cassandra etc.

Data source API examples

- SQLContext for accessing data source API's
- sqlContext.read is way to load from given source
- Examples
  - Loading CSV file – CSVFile.scala
  - Loading JSON file – JsonFile.scala
- Can we mix and match sources having same schema ?
  - Example : MixSources.scala

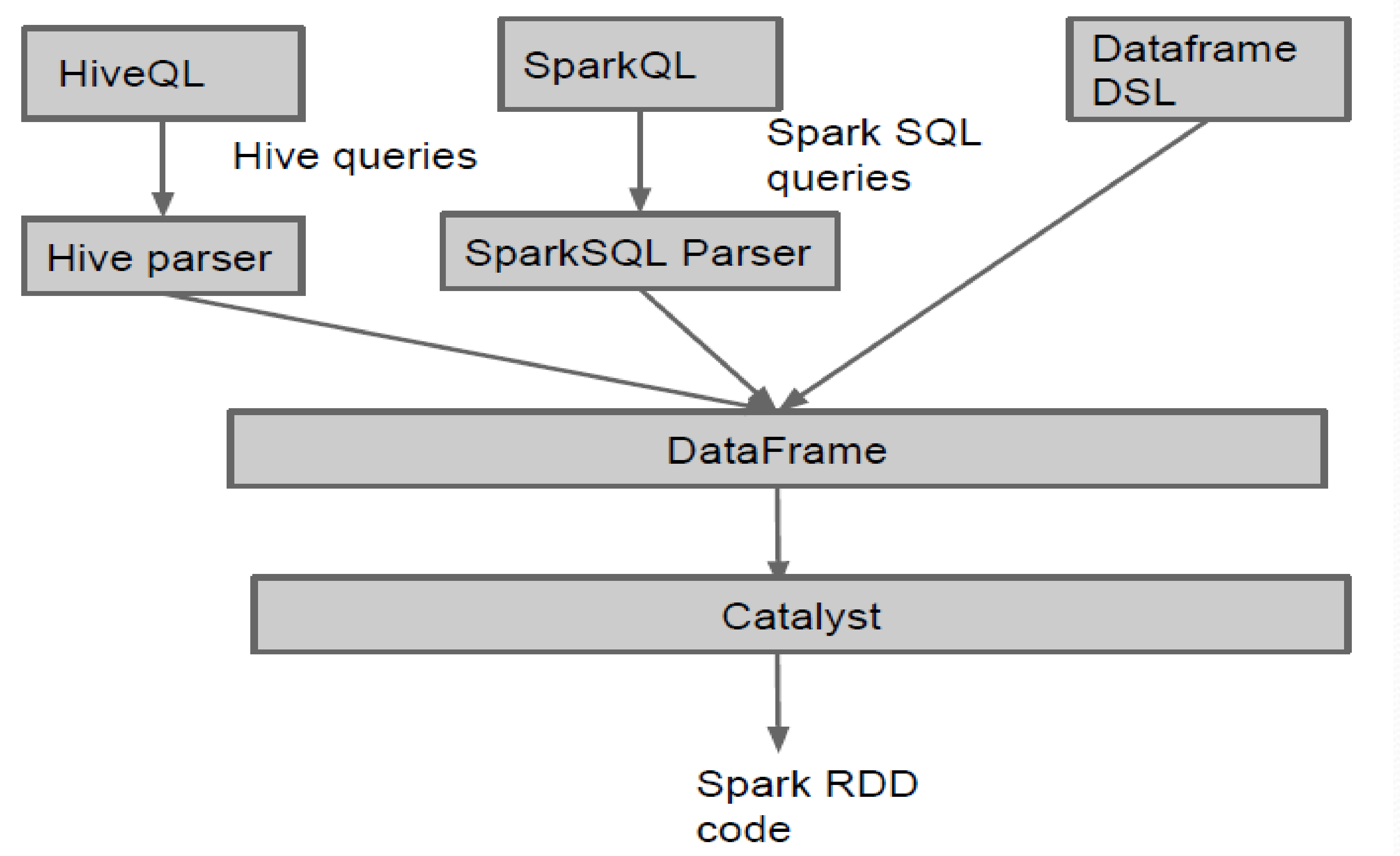
DataFrame

- Single abstraction for representing structured data in Spark
- DataFrame = RDD + Schema (aka SchemaRDD)
- All data source API's return DataFrame
- Introduced in 1.3
- Inspired from R and Python panda
- .rdd to convert from dataframe to RDD representation resulting in RDD[Row]
- Support for DataFrame DSL in Spark

Need for new abstraction

- Single abstraction for structured data
  - Ability to combine data from multiple sources
  - Uniform access from all different language API's
  - Ability to support multiple DSL's
- Familiar interface to Data scientists
  - Same API as R/Panda
  - Easy to convert from R local data frame to Spark
  - New SparkR is built around it

Spark SQL pipeline

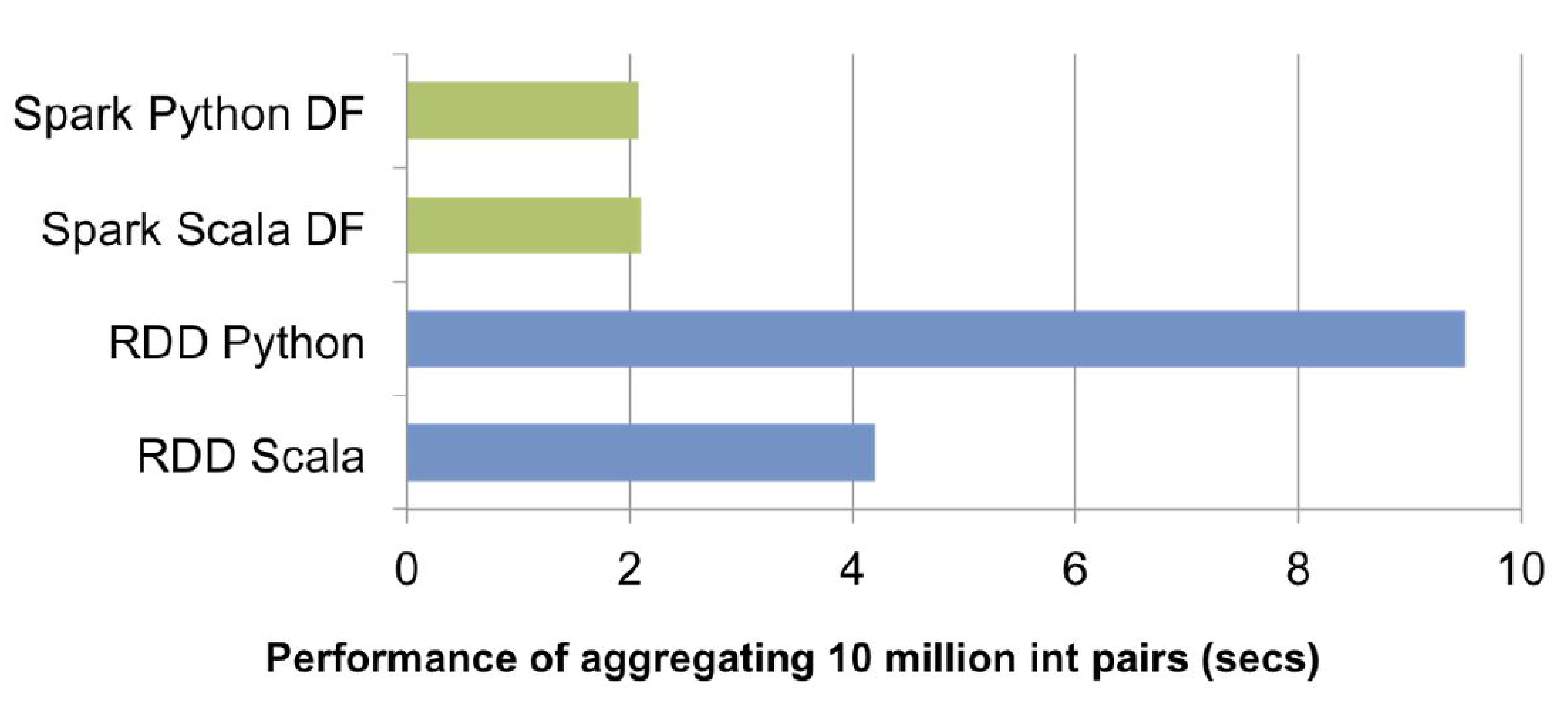


Querying data frames using SQL

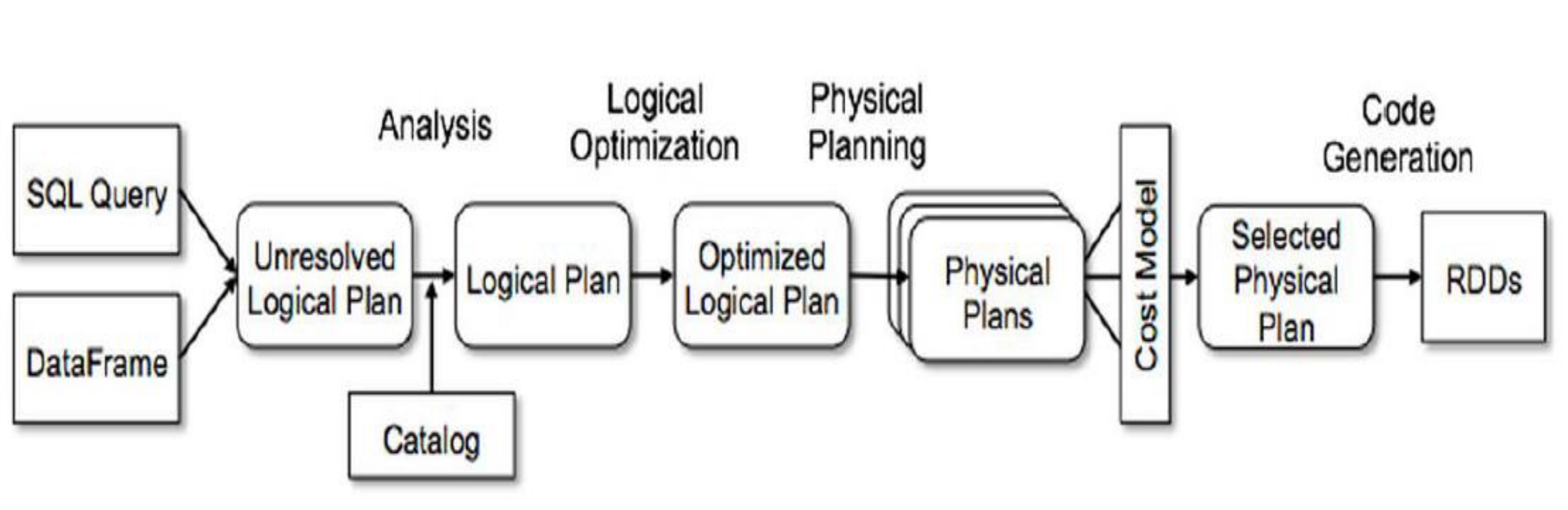
- Spark-SQL has a built in Spark sql interpreter and optimizer similar to Hive
- Support both Spark SQL and Hive dialect
- Support for both temporary and hive metastore
- All ideas like UDF, UDAF, Partitioning of Hive is supported

RDD transformations	Data Frame transformation
Actual transformation is shipped on cluster	Optimized generated transformation is shipped on cluster
No schema need to be specified	Mandatory Schema
No parser or optimizer	SQL parser and optimizer
Lowest API on platform	API built on SQL which is intern built on RDD
Don’t use smart source capabilities	Make effective use of smart souces
Different performance in different language API’s	Same performance across all different languages

Performance



Why so fast?


SQL Query
DataFrame
Analysis
Unresolved Logical Plan
Catalog
Logical Optimization
Logical Plan
Physical Planning
Optimized Logical Plan
Physical Plans
Cost Model
Selected Physical Plan
Code Generation
RDDs