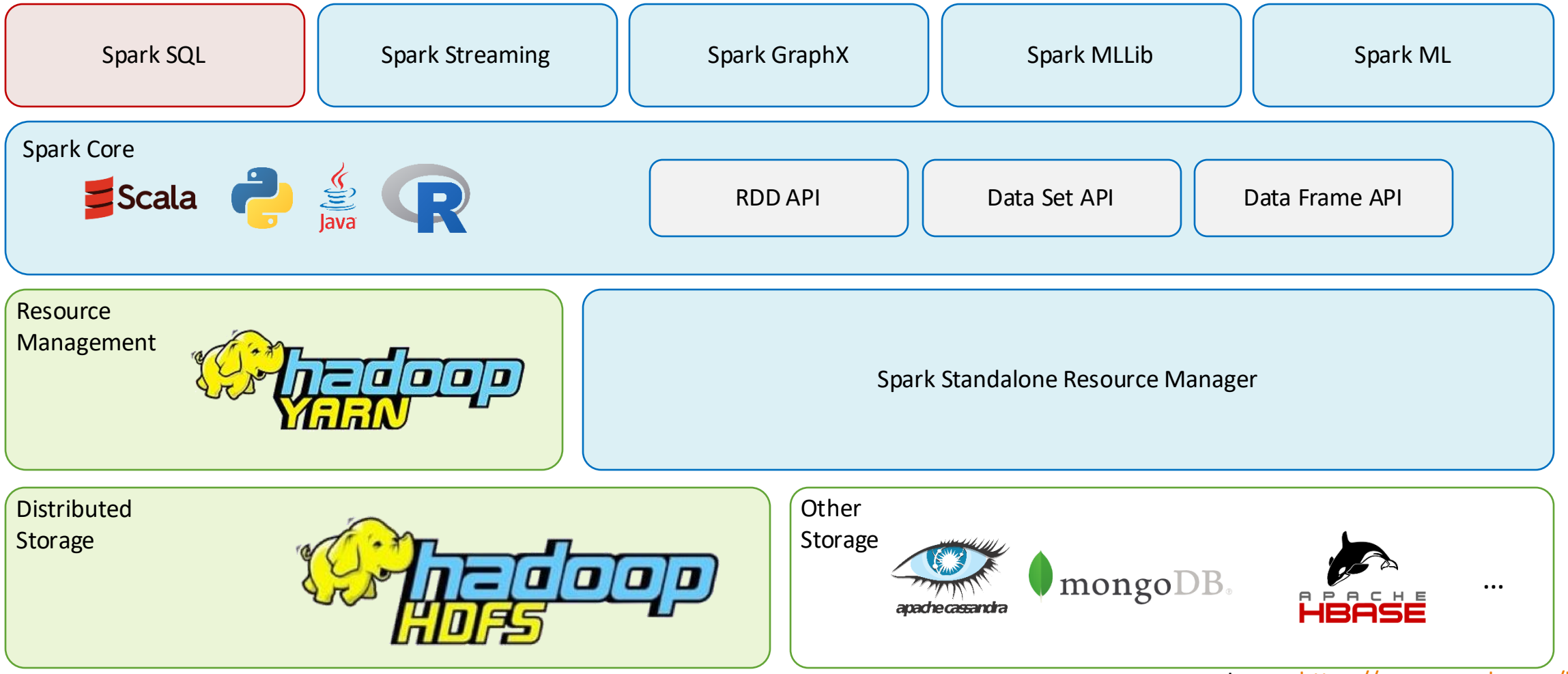
Apache Spark SQL

Abdel Dadouche
DJZ Consulting

adadouche@hotmail.com
@adadouche

Remember the Spark stack …



Logos: https://www.apache.org/logos

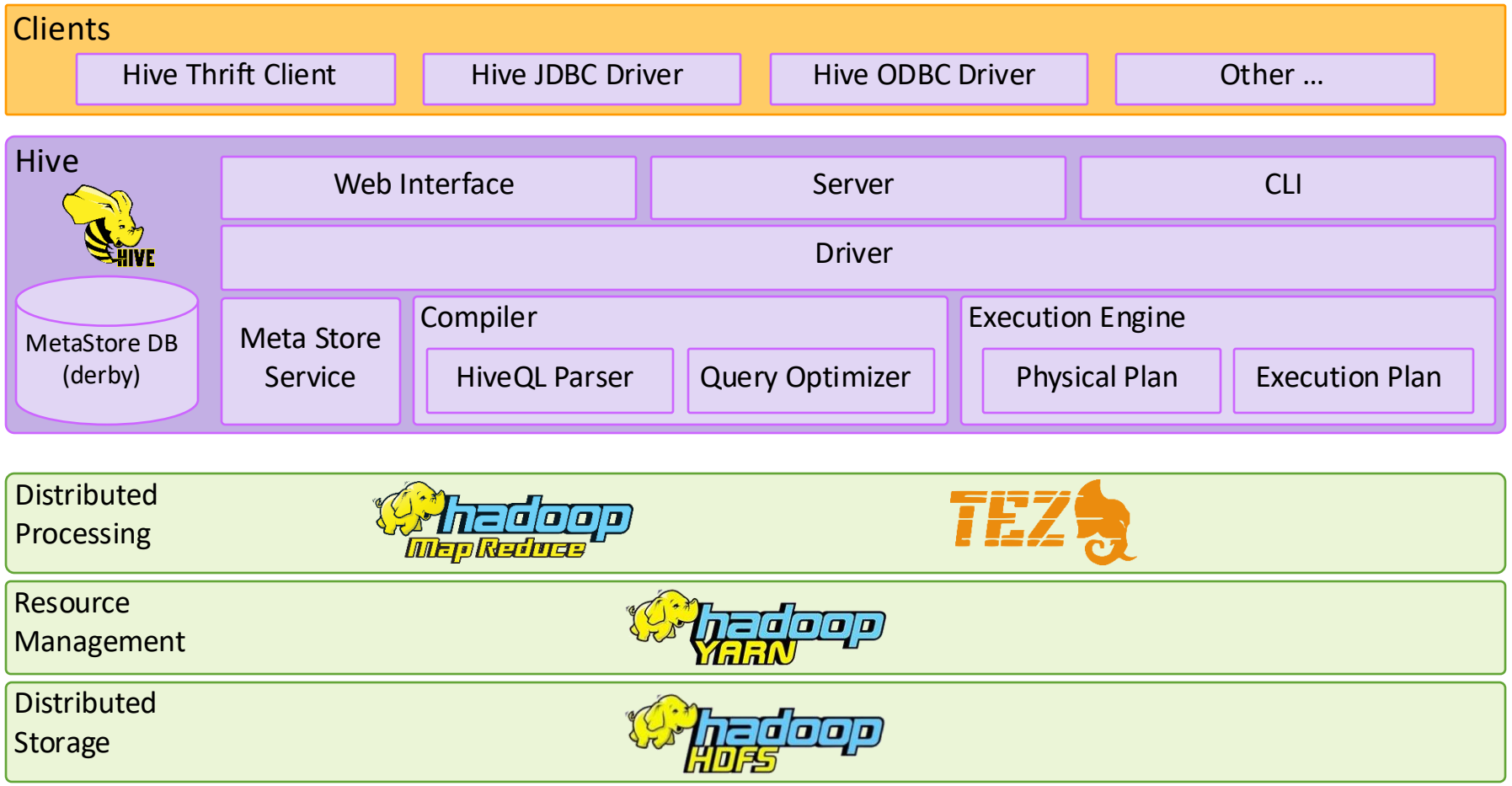
Some History

- **Shark** (not SparkSQL) was initially started as a Apache Spark subproject
- Project timelines
  - started in 2011 (Spark was started in 2009)
  - alpha version available in April 2012
- The intent was to port Apache Hive capabilities to Apache Spark
  - Often referred as “Hive on Spark”
- **Shark was deprecated and replaced by SparkSQL** (July 2014)
- **Hive on Spark** was added in 2015 to Hive to complement the MapReduce/Tez engines

From Hive to Shark to Spark SQL

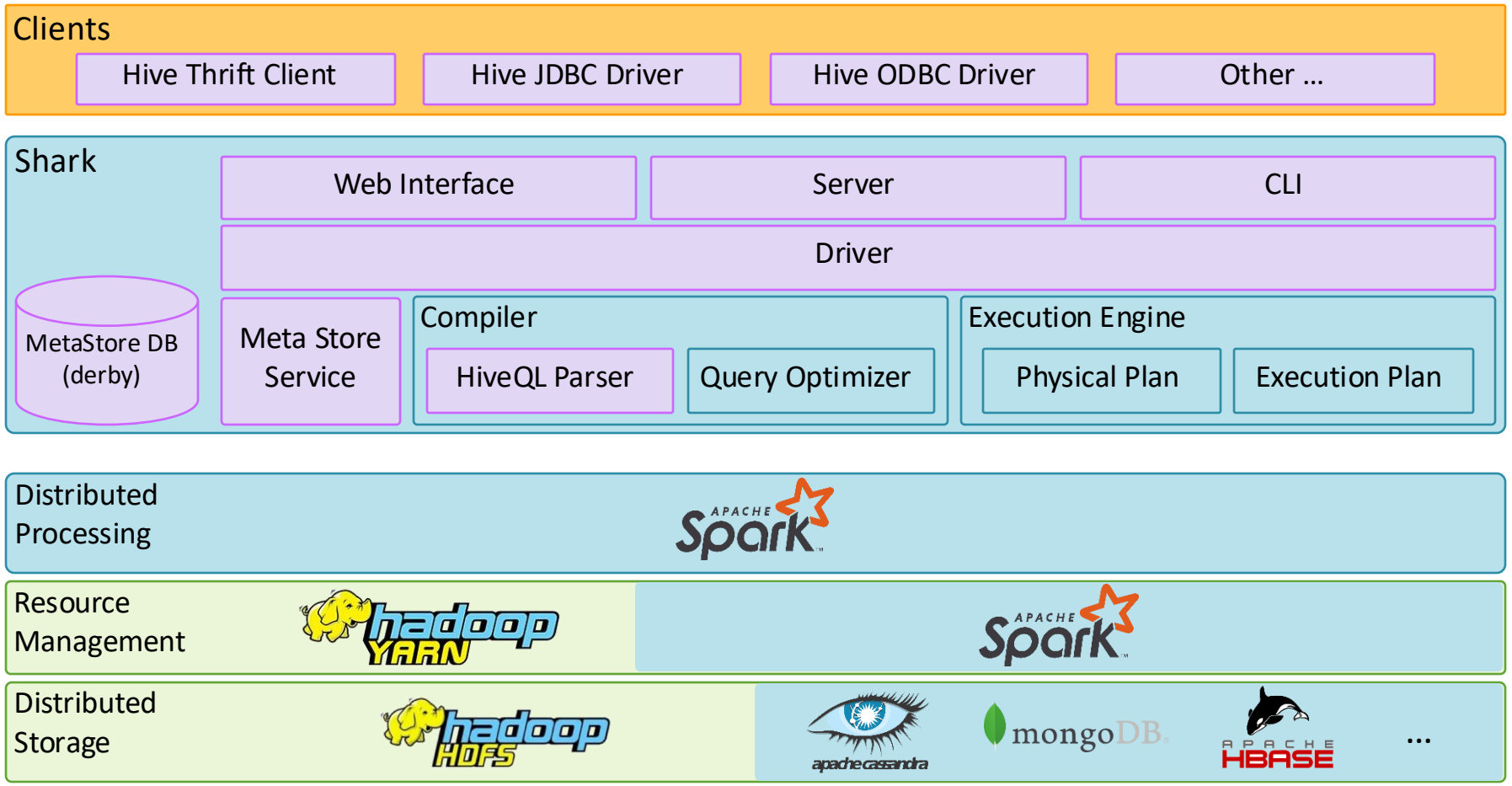
- In a few words, Shark reimplemented most of the Hive execution engine on Spark instead of MapReduce / Tez
- However, Shark inherited the Hive codebase (large & complicated, hard to optimize and maintain)
- SparkSQL was designed from ground-up to leverage the power of Spark based on the lesson learned while building Shark and maintaining performances and compatibility with Shark/Hive (Hive on Spark)

Remember the Hive architecture…



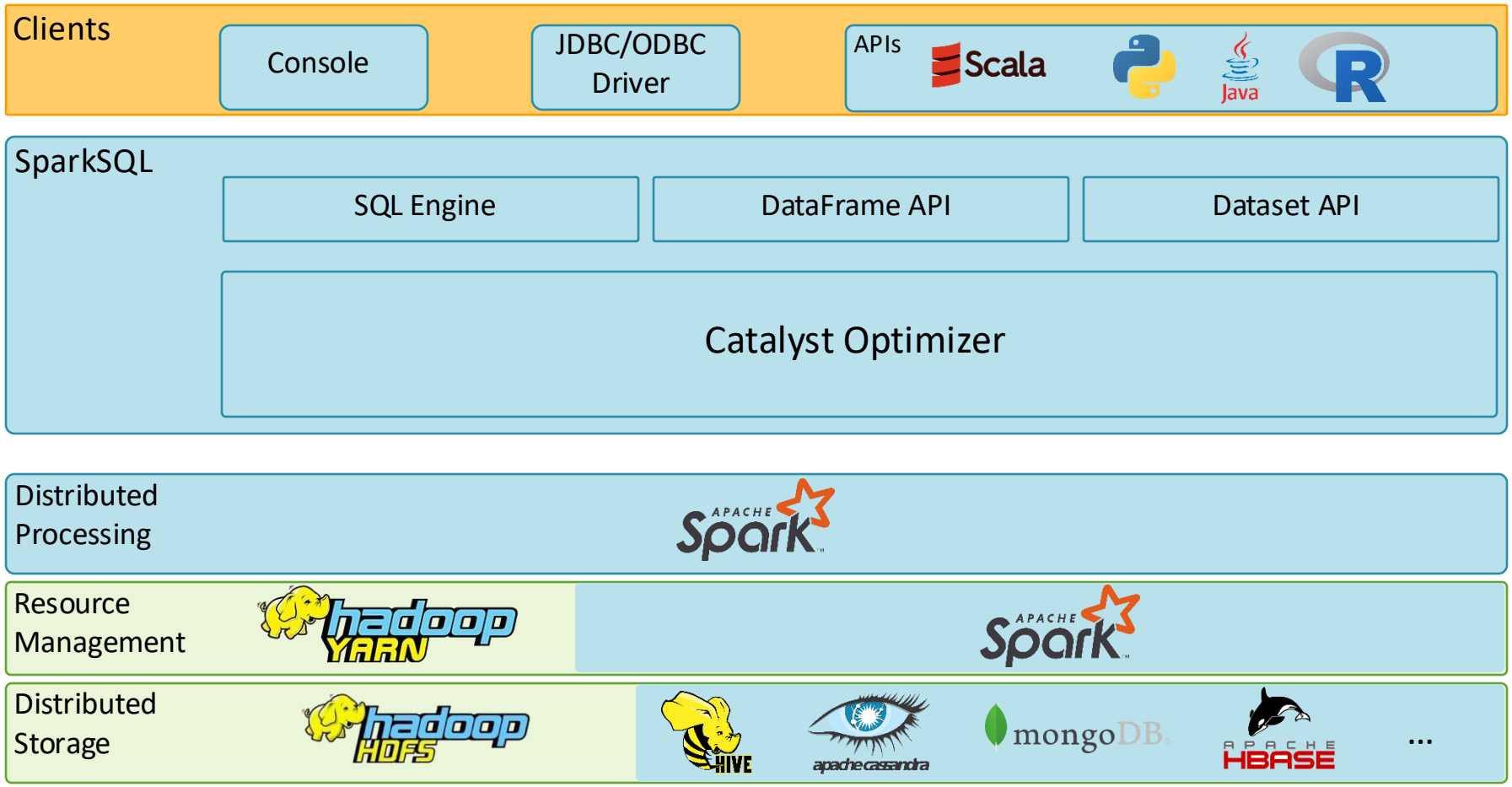
Logos: https://www.apache.org/logos

Then the Shark architecture …



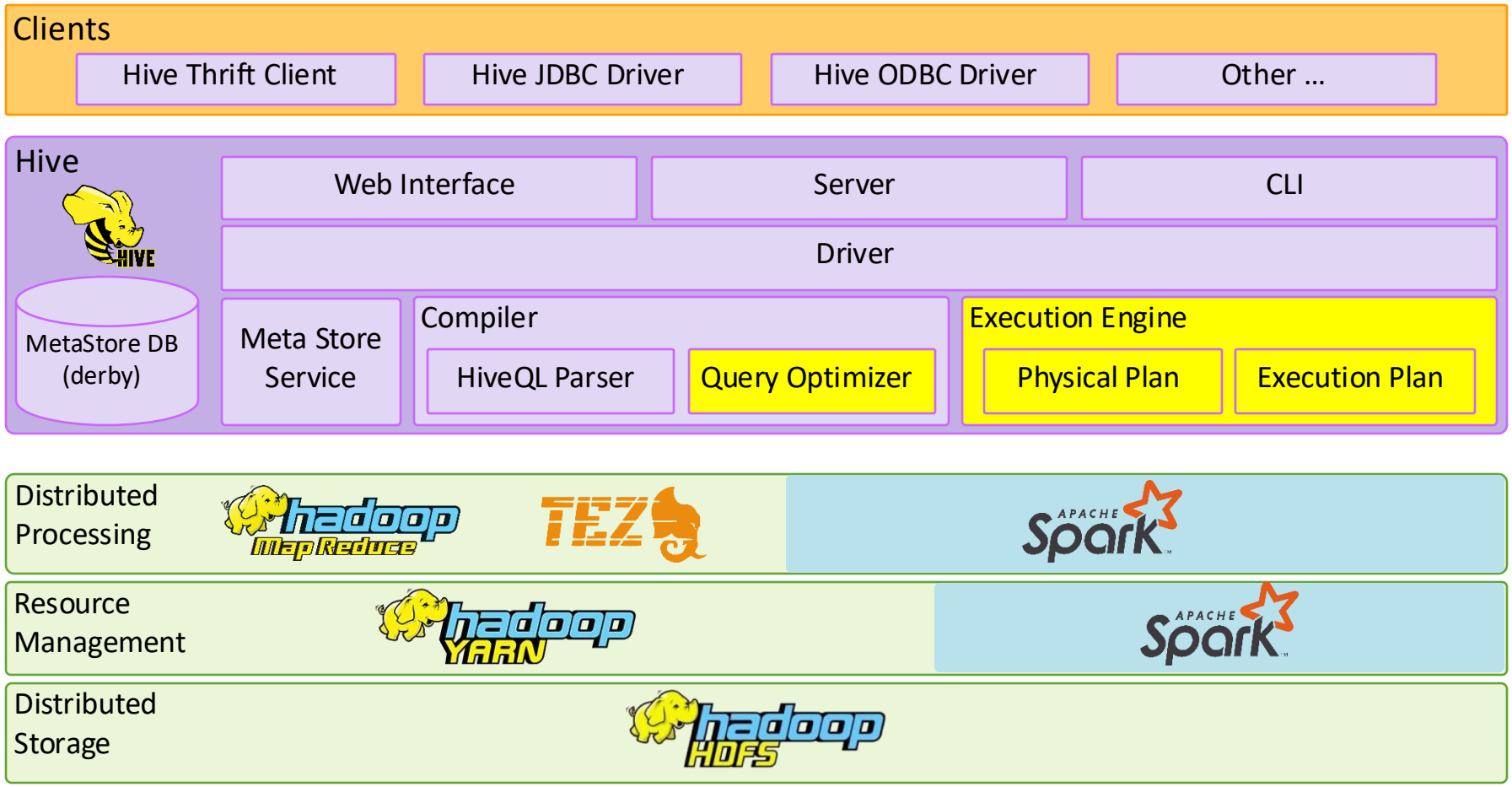
Logos: https://www.apache.org/logos

And finally, the SparkSQL architecture …



Logos: https://www.apache.org/logos

But also the “Hive on Spark” architecture…



More info: https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark
Logos: https://www.apache.org/logos

Why Spark SQL?

It’s more concise, simpler & faster

Spark SQL:

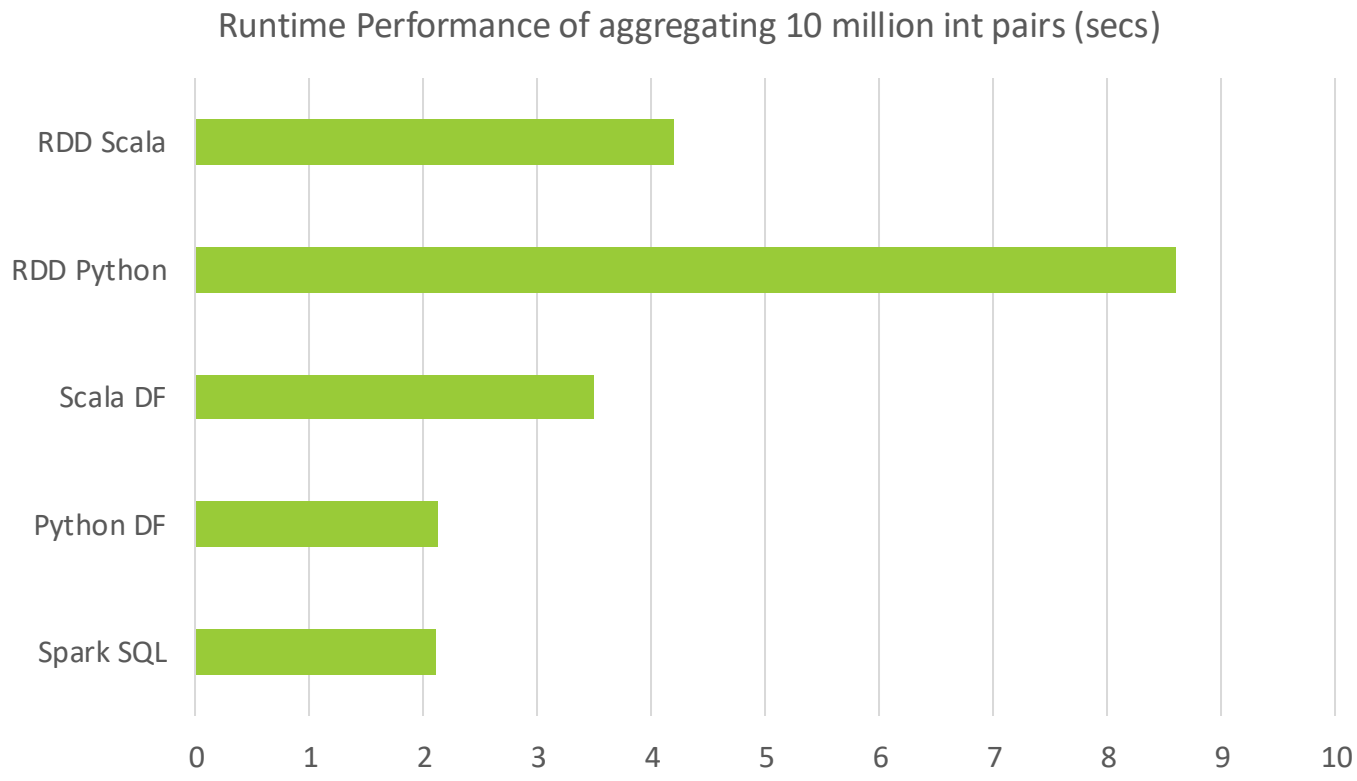
```
select dept, avg(age) from data group by dept
```

DataFrame:

```
data.groupBy("dept").avg(“age")
```

Scala RDD:

```
data.map {
  case (dept, age) => dpt -> (age, 1)
}.reduceByKey {
  case ((a1, c1), (a2, c2)) => ( a1 +a2, c1 + c2)
}.map {
  case (dept, (age, c)) => dpt -> age / c
}
```



More info: https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html

Spark SQL - For **SQL users**

- Provides state-of-the-art SQL performance
- Maintains compatibility with Shark/Hive.
- Spark SQL supports existing
  - Hive data formats
  - user-defined functions (UDF)
  - Hive meta store.
- With features that will be introduced since Apache Spark 1.1.0, Spark SQL beats Shark in TPC-DS performance!

More info: https://databricks.com/blog/2014/07/01/shark-spark-sql-hive-on-spark-and-the-future-of-sql-on-spark.html

Spark SQL - For **Spark users**

- Enable ingestion of data from heterogeneous schema-full/less sources such as JSON, Parquet, Hive, or EDWs
- Ease the manipulation of (semi-) structured data
- Unifies SQL and sophisticated analysis
- Allow users to mix and match SQL and Spark APIs for advanced analytics

More info: https://databricks.com/blog/2014/07/01/shark-spark-sql-hive-on-spark-and-the-future-of-sql-on-spark.html

Spark SQL - For **open source developers**

- Proposes a novel, elegant way of building query planners
- Ease to add new optimizations under this framework
- Strong support and enthusiasm from the open source community, largely thanks to this new design.
- After only three months, over 40 authors have contributed code to it

More info: https://databricks.com/blog/2014/07/01/shark-spark-sql-hive-on-spark-and-the-future-of-sql-on-spark.html