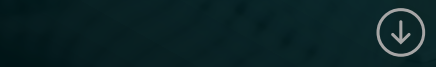
Spark SQL và Big Data File format

- Giới thiệu về Spark SQL.
- Các định dạng dữ liệu trong Spark
- Đọc/ ghi dữ liệu với Dataframer Reader/ Writer
- Làm quen với các API của Spark
- Thực hành Spark SQL trên DataBricks Cluster

Giảng Viên: Nguyễn Chí Thanh



Nguyễn **Chí Thanh**
Big Data Engineer/ Data Architect
Blog: https://karcuta.medium.com

ABOUT **ME**

- Trên 5 năm kinh nghiệm trong lĩnh vực Big Data Engineering.
- Tham gia xây dựng và triển khai hệ thống vBI, Viettel Data Lake cho Viettel Telecom.
- Sở hữu chứng chỉ Quốc tế về Hadoop, Spark do Cloudera, Databricks cấp (CCA 175, CRT020).
- Thiết kế phát triển các hệ thống trên nền tảng Hadoop Ecosytem: Hdfs, Spark, Kafka, Hive...

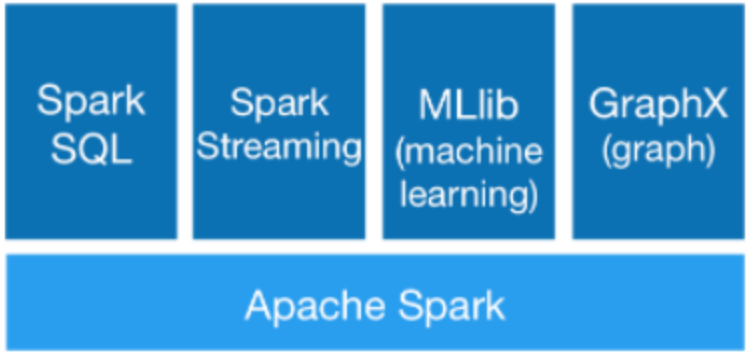
1.
Spark SQL Overview

Spark SQL (1)

Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.

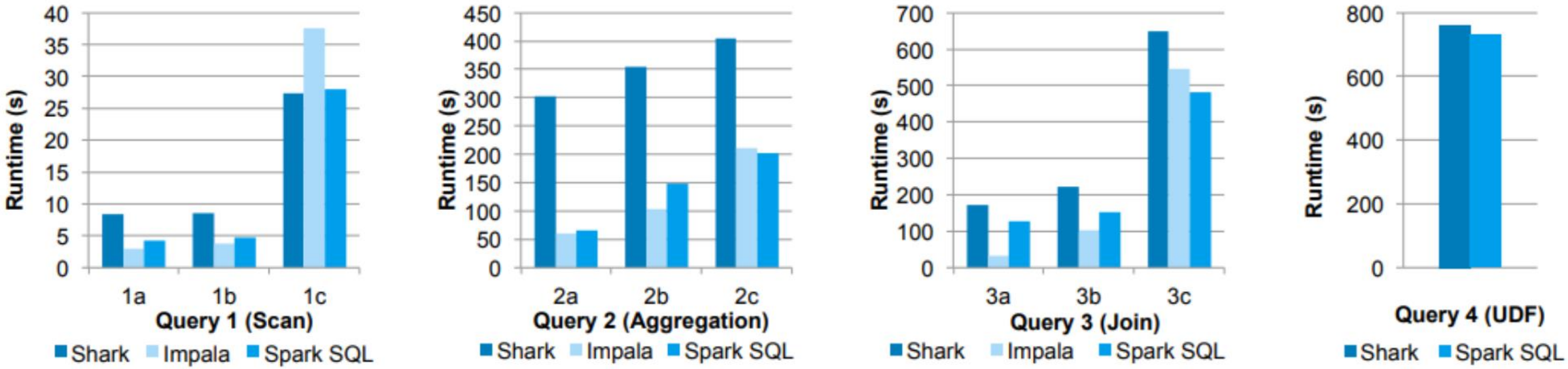


- Là 1 trong các thành phần trong bộ thư viện Apache Spark
- Hỗ trợ thao tác trên dữ liệu sử dụng SQL
- Cung cấp API hoặc có thể query trực tiếp dữ liệu thông qua *SQL Query*

Spark SQL (2)

- Được xây dựng phía trên tầng Spark Core, thừa hưởng tất cả các tính năng mà RDD có.
- Làm việc với tập dữ liệu là DataSet hoặc DataFrame (tập dữ liệu phân tán, có cấu trúc)
- Hiệu năng cao, khả năng mở rộng và chịu lỗi tốt
- Tương tích với các thành phần khác trong tổng thể Spark Framework (như Streaming/ Mllib, GraphX)
- Bao gồm 2 thành phần là DataSet API và Catalyst Optimizer.

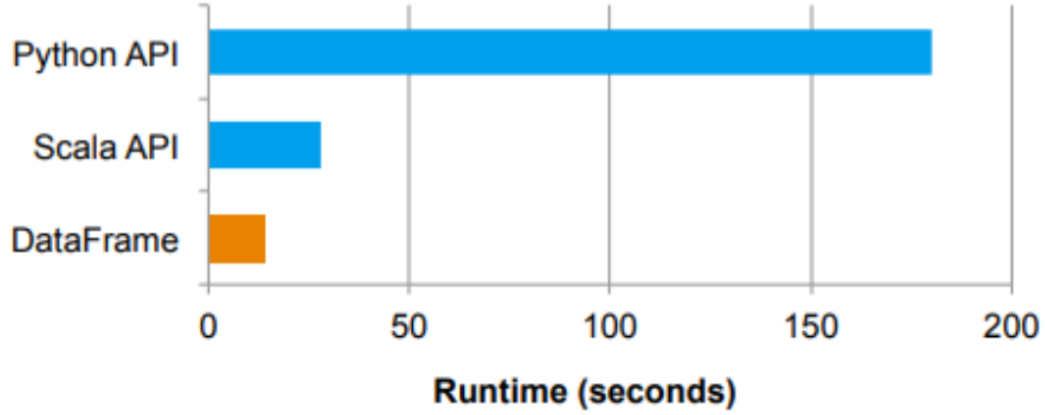
Spark SQL Performance (1)



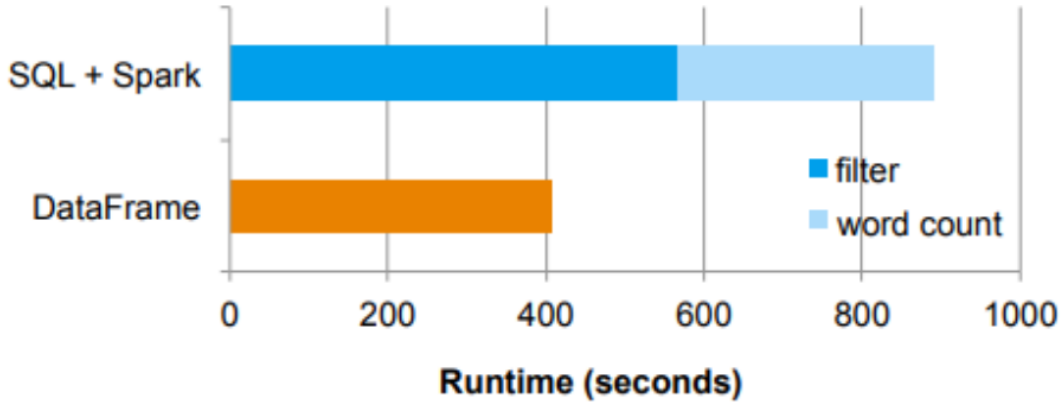
Performance of Shark, Impala and Spark SQL on Big Data benchmark queries

So sánh hiệu năng giữa các SQL Engine: Shark, Impala, SparkSQL

Spark SQL Performance (2)



Performance of an aggregation written using the native Spark Python and Scala APIs versus the DataFrame API.

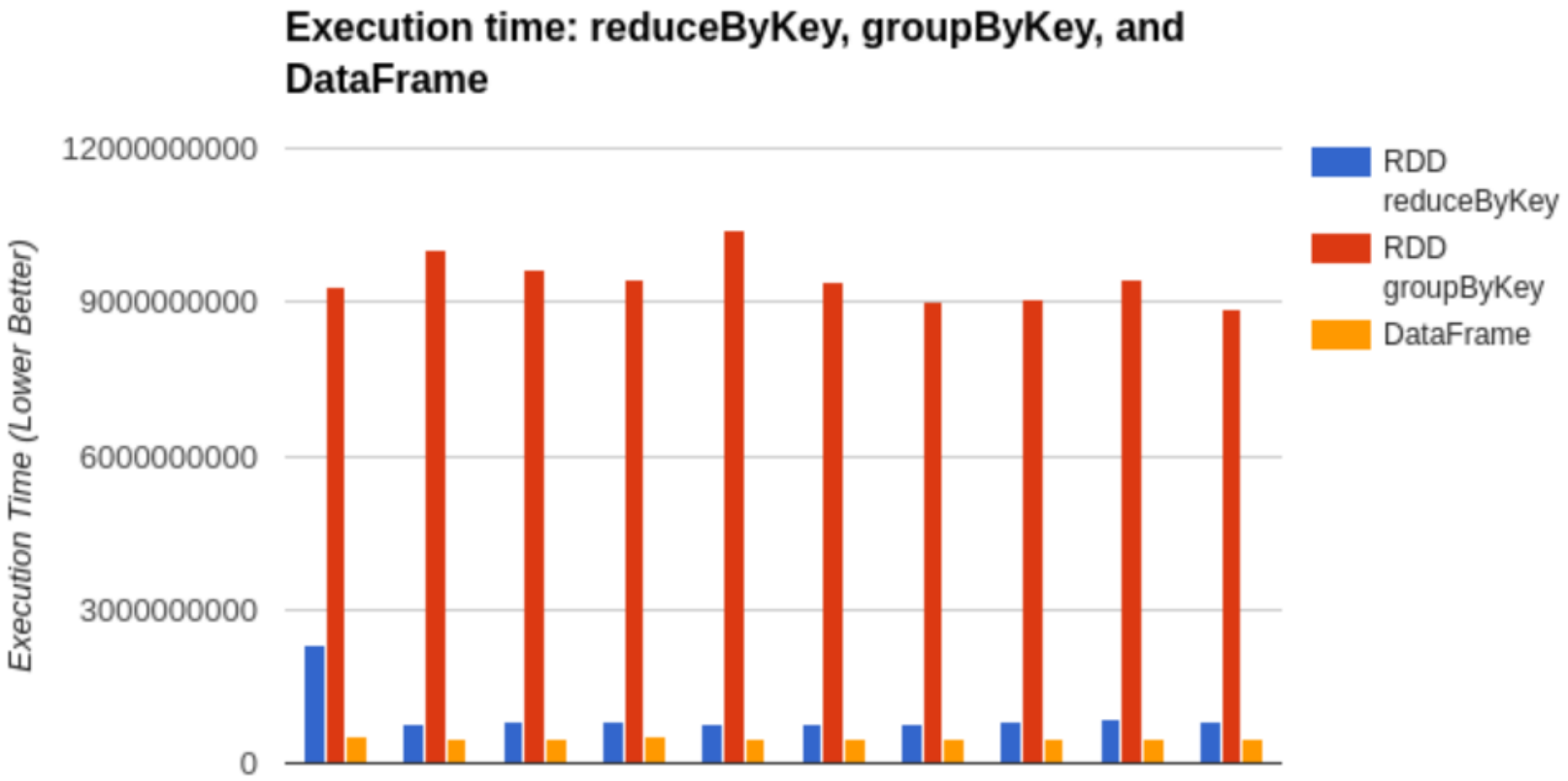


Performance of a two-stage pipeline written as a separate Spark SQL query and Spark job (above) and an integrated DataFrame job (below).

Spark DataFrames vs RDDs and SQL

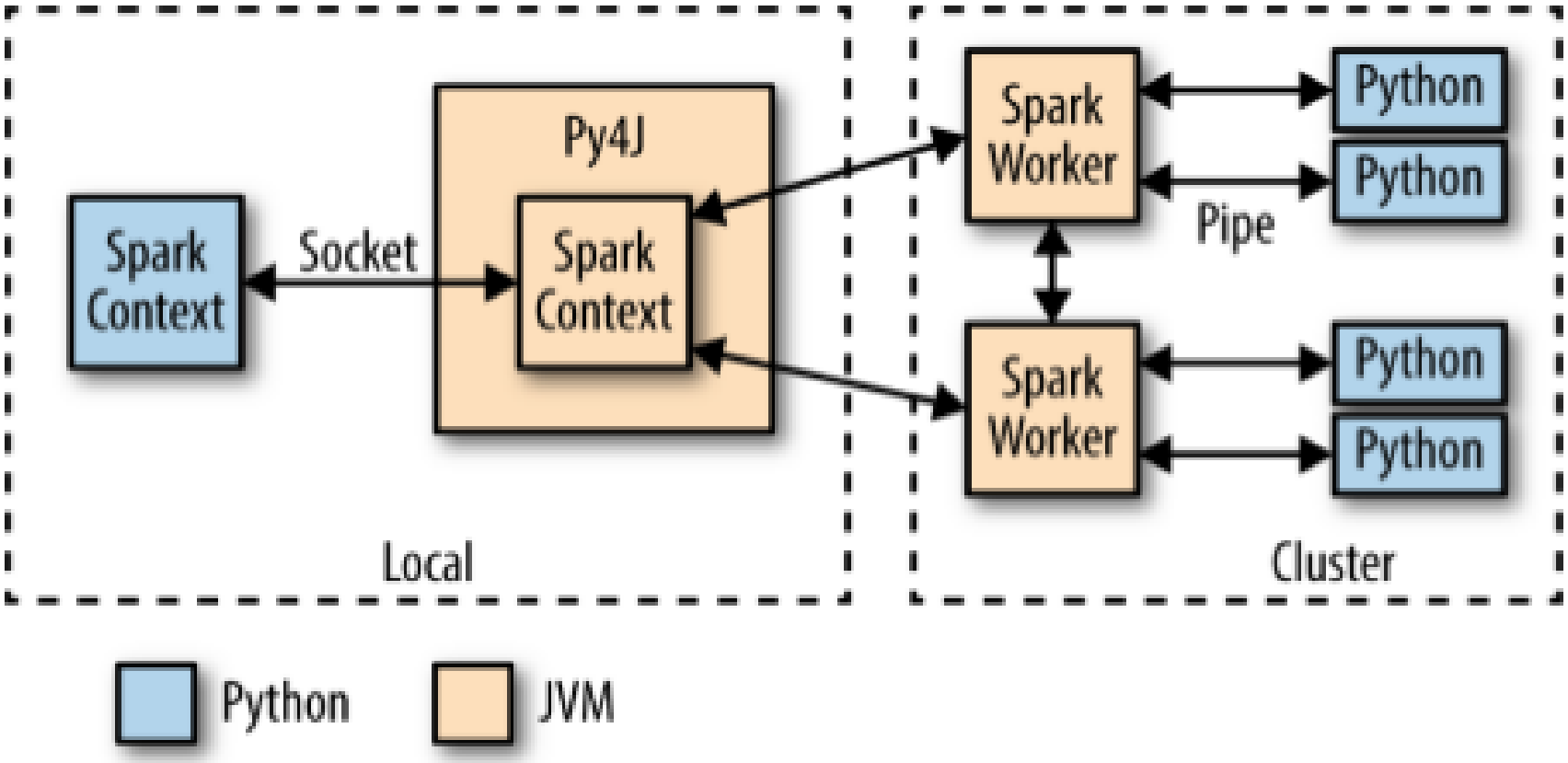
So sánh hiệu năng giữa Spark Core vs Spark SQL với Dataframe

Spark SQL Performance (3)

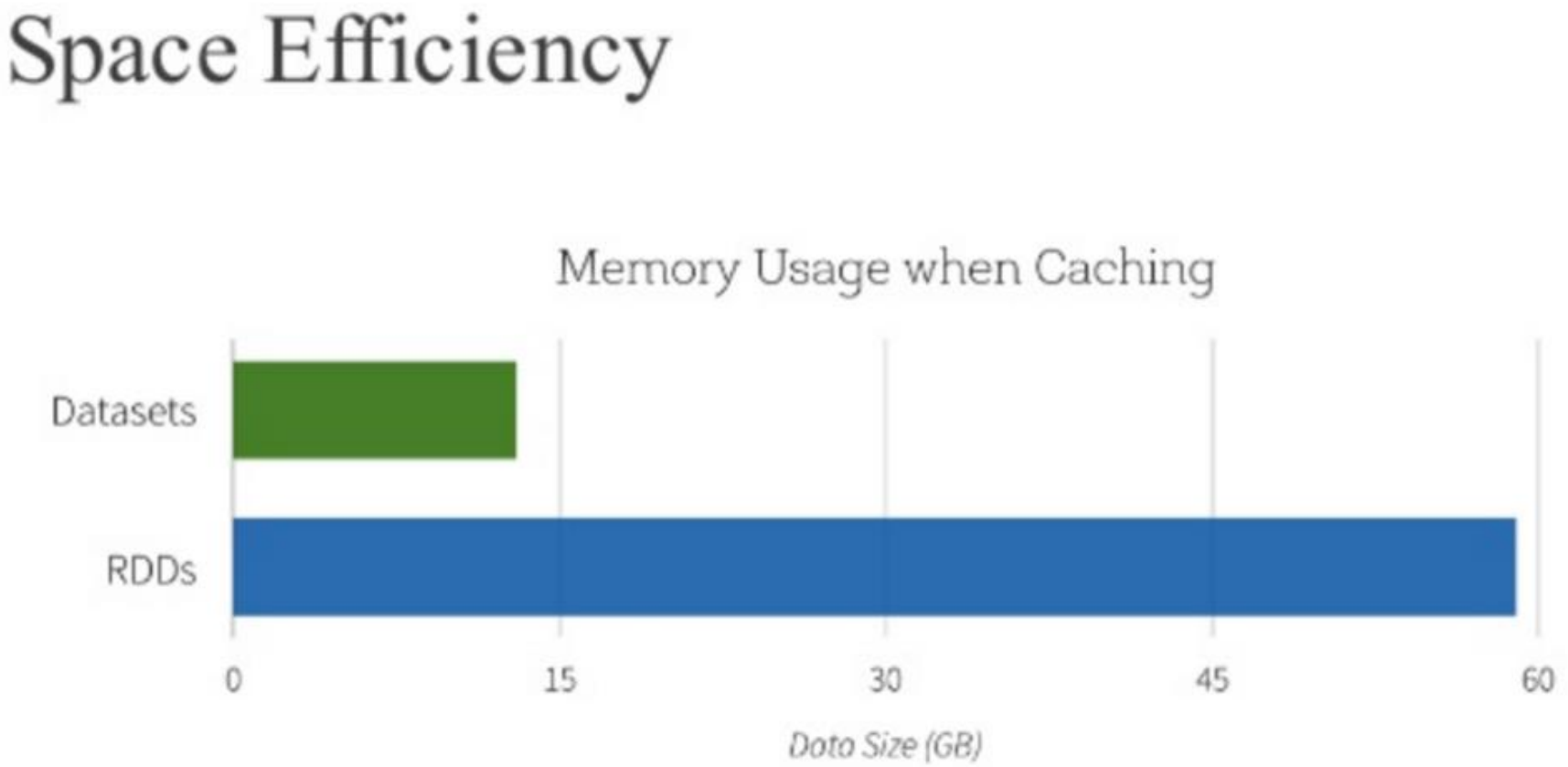


So sánh hiệu năng giữa Spark Core vs Spark SQL với Dataframe theo 1 số transformation thông dụng

Spark SQL Performance (4)



Spark SQL Performance (5)

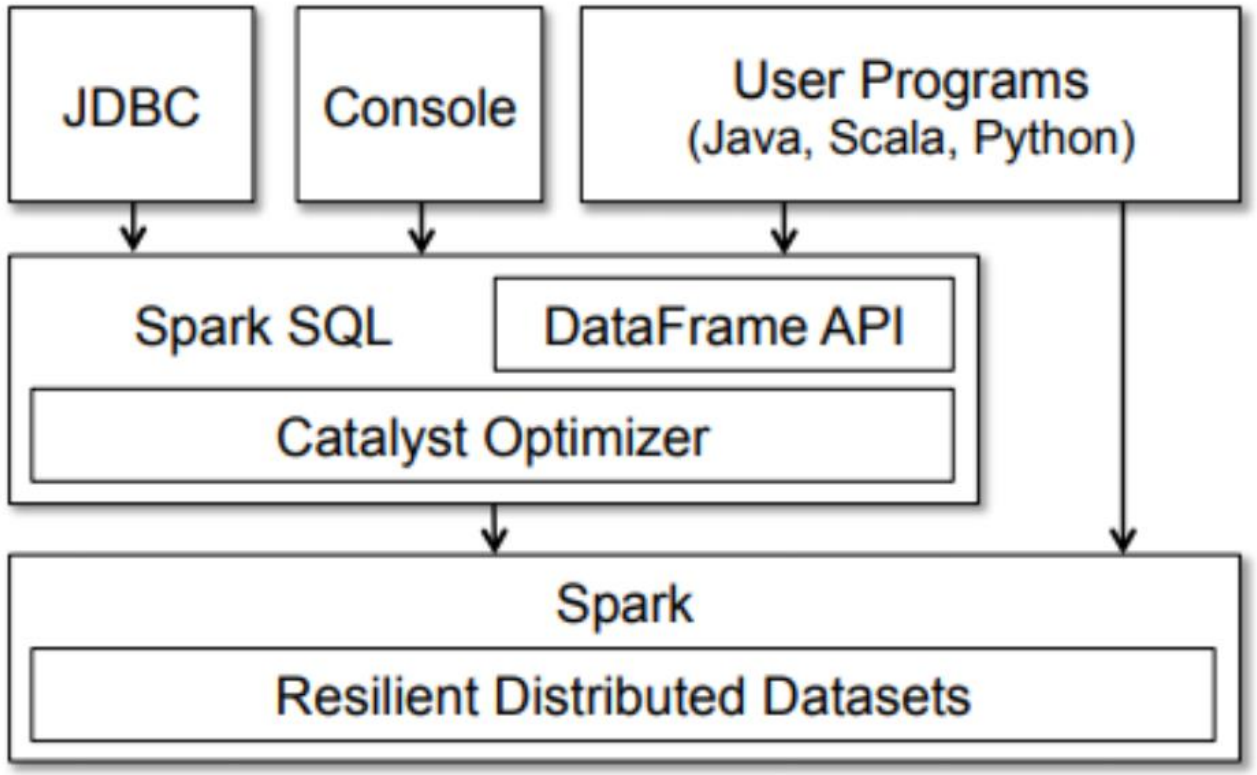


Khả năng sử dụng memory của Spark SQL (Dataset) tốt hơn so với Spark Core RDD

2.
Spark SQL Components

Spark SQL

- Dataframe APIs: Các APIs hỗ trợ tương tác với Dataframe (tập dữ liệu có cấu trúc, phân tán) như select, đọc, ghi, lọc...
- Catalyst Optimizer: Tối ưu hóa các step xử lý trước khi tạo task tính toán



DataFrame (1)

"A *Dataframe* is an immutable, distributed collection of data that is organized into rows, where each one consists a set of columns and each column has a name and an associated type"

DataFrames

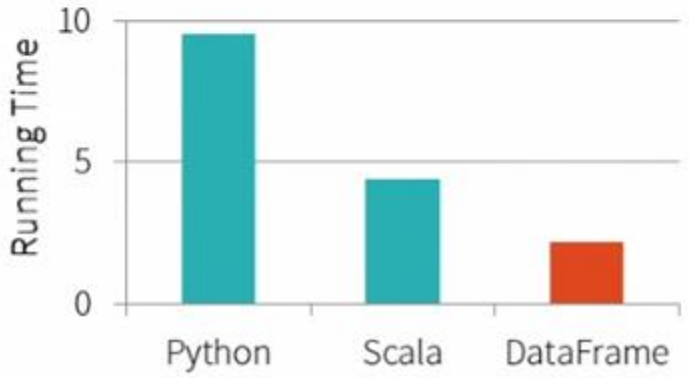
Similar API to data frames in R and Pandas

Automatically optimized via Spark SQL

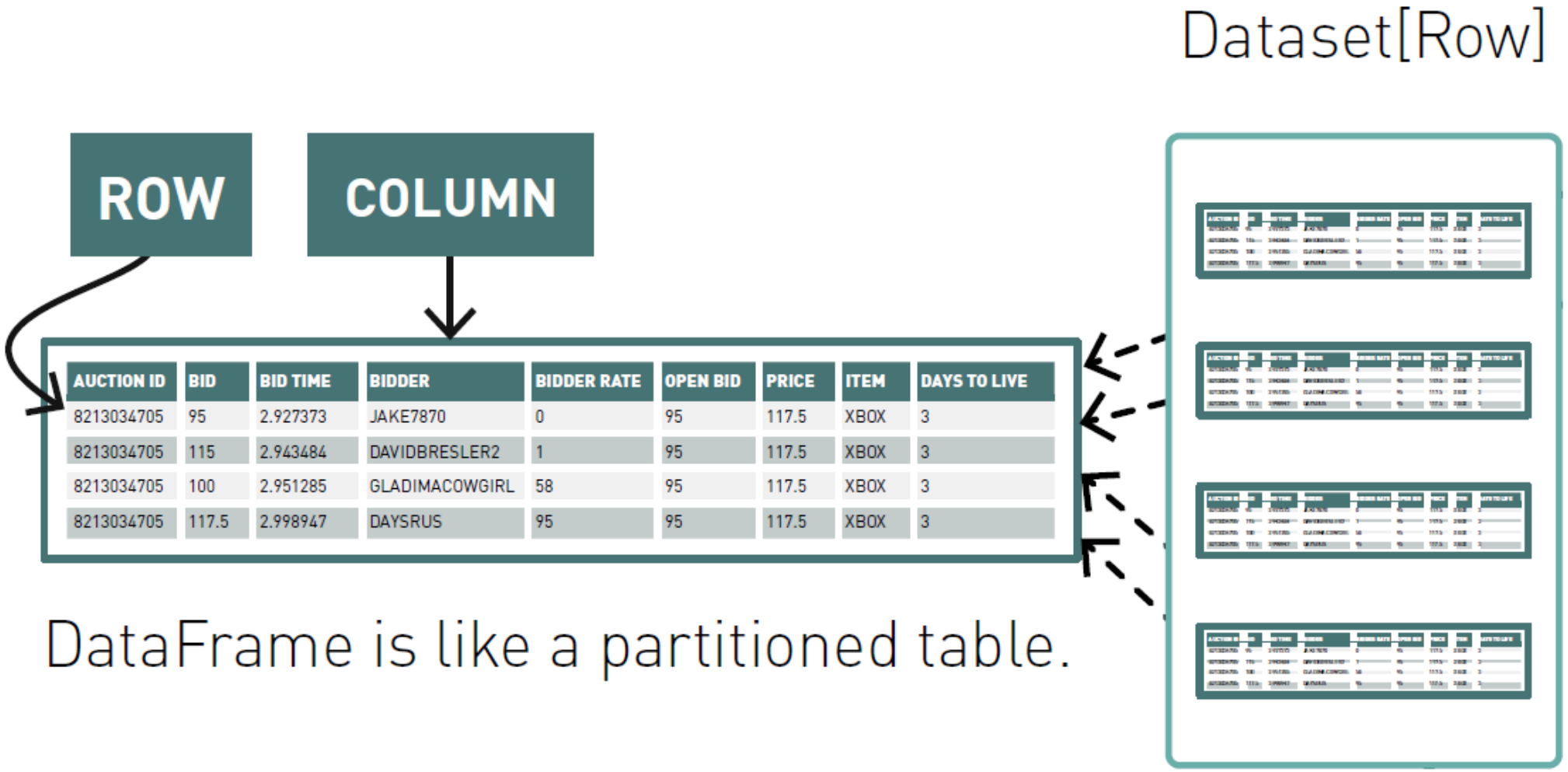
Coming in Spark 1.3

```
df = jsonFile("tweets.json")

df[df["user"] == "matei"]
  .groupBy("date")
  .sum("retweets")
```

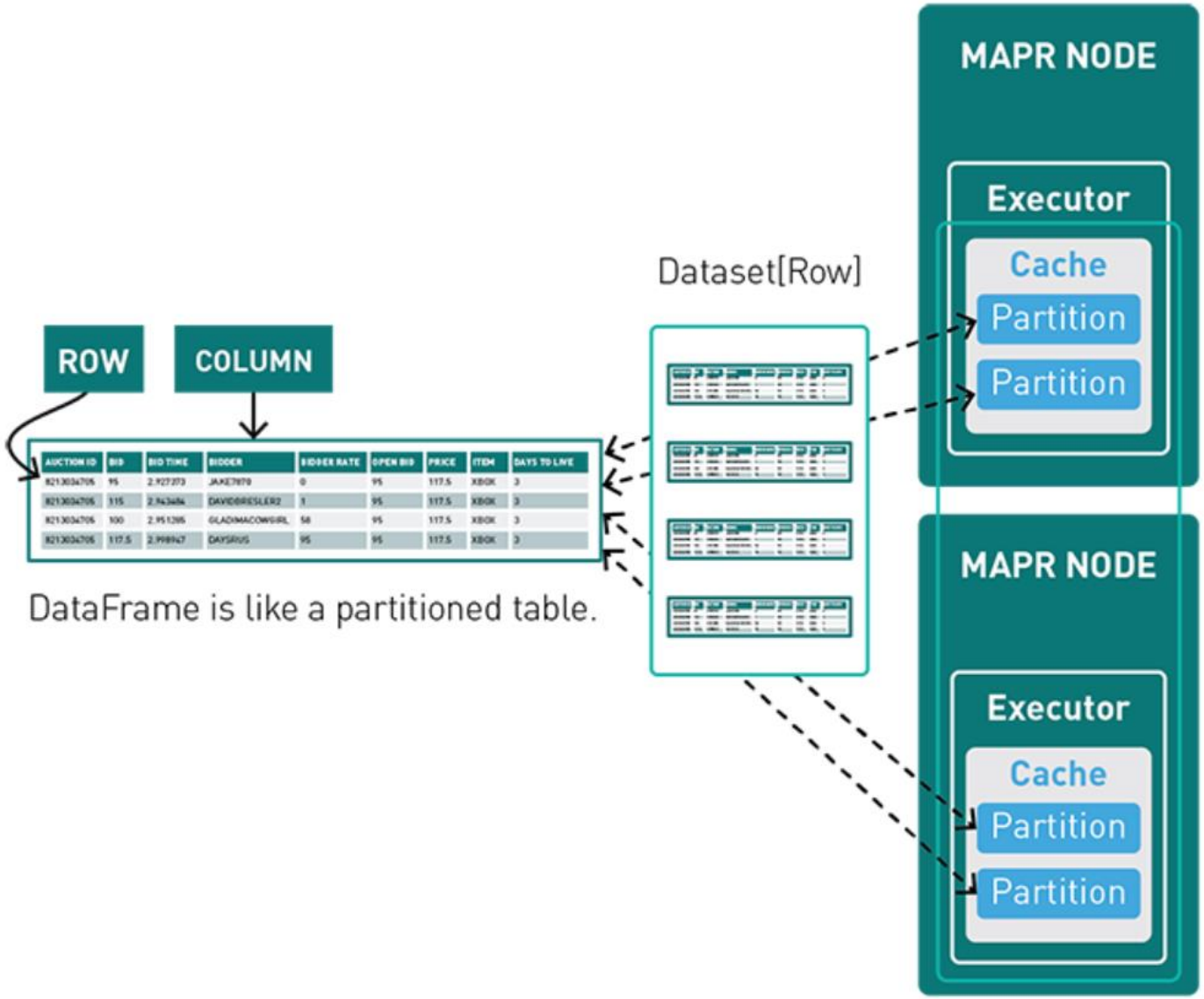


DataFrame (2)


Dataset[Row]
ROW
COLUMN
AUCTION ID | BID | BID TIME | BIDDER | BIDDER RATE | OPEN BID | PRICE | ITEM | DAYS TO LIVE
8213034705 | 95 | 2.927373 | JAKE7870 | 0 | 95 | 117.5 | XBOX | 3
8213034705 | 115 | 2.943484 | DAVIDBRESLER2 | 1 | 95 | 117.5 | XBOX | 3
8213034705 | 100 | 2.951285 | GLADIMACOWGIRL | 58 | 95 | 117.5 | XBOX | 3
8213034705 | 117.5 | 2.998947 | DAYSRUS | 95 | 95 | 117.5 | XBOX | 3
DataFrame is like a partitioned table.

DataFrame (3)

- Mỗi bản ghi (hàng) là 1 đối tượng *Row*
- Dữ liệu được chia thành các partitions
- Mỗi partition chứa 1 phần dữ liệu.
- Các partitions phân tán trong cụm, tương tự RDD



Catalyst Optimizer (1)

```
events =
 sc.read.json("/logs")

stats =
 events.join(users)
  .groupBy("loc","status")
  .avg("duration")

errors = stats.where(
 stats.status == "ERR")
```


```
while(logs.hasNext) {
  e = logs.next
  if(e.status == "ERR") {
    u = users.get(e.uid)
    key = (u.loc, e.status)
    sum(key) += e.duration
    count(key) += 1
  }
}
...
```

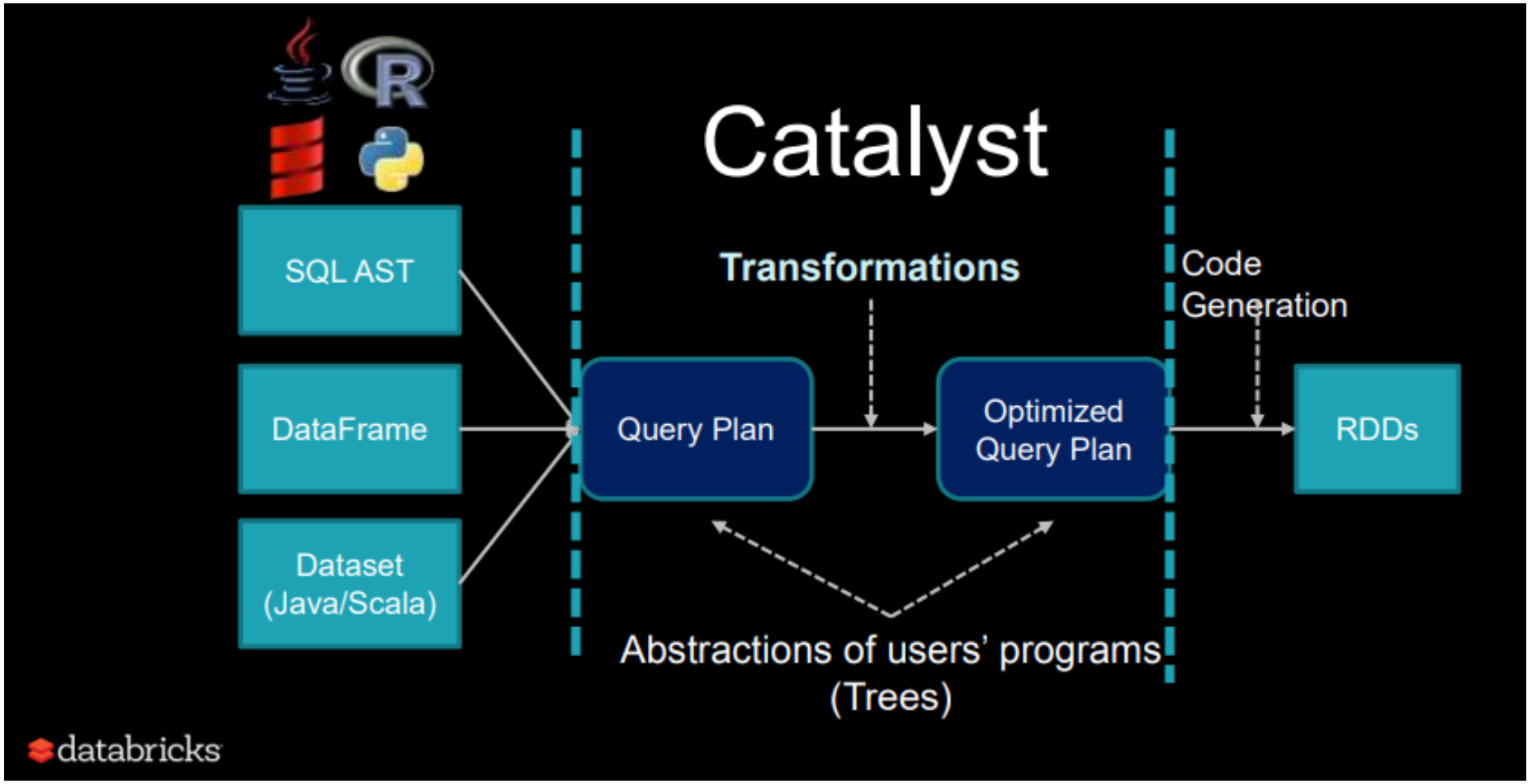
DataFrame API

Optimized Plan

Specialized Code

databricks

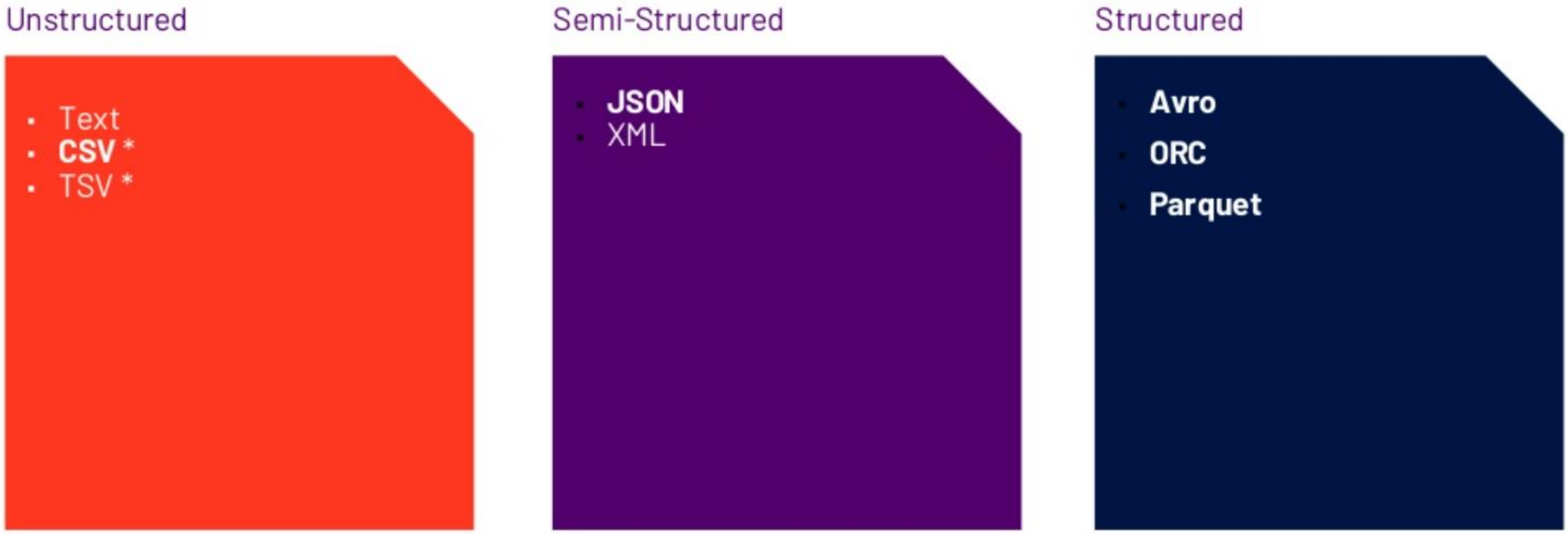
Catalyst Optimizer (2)



3.

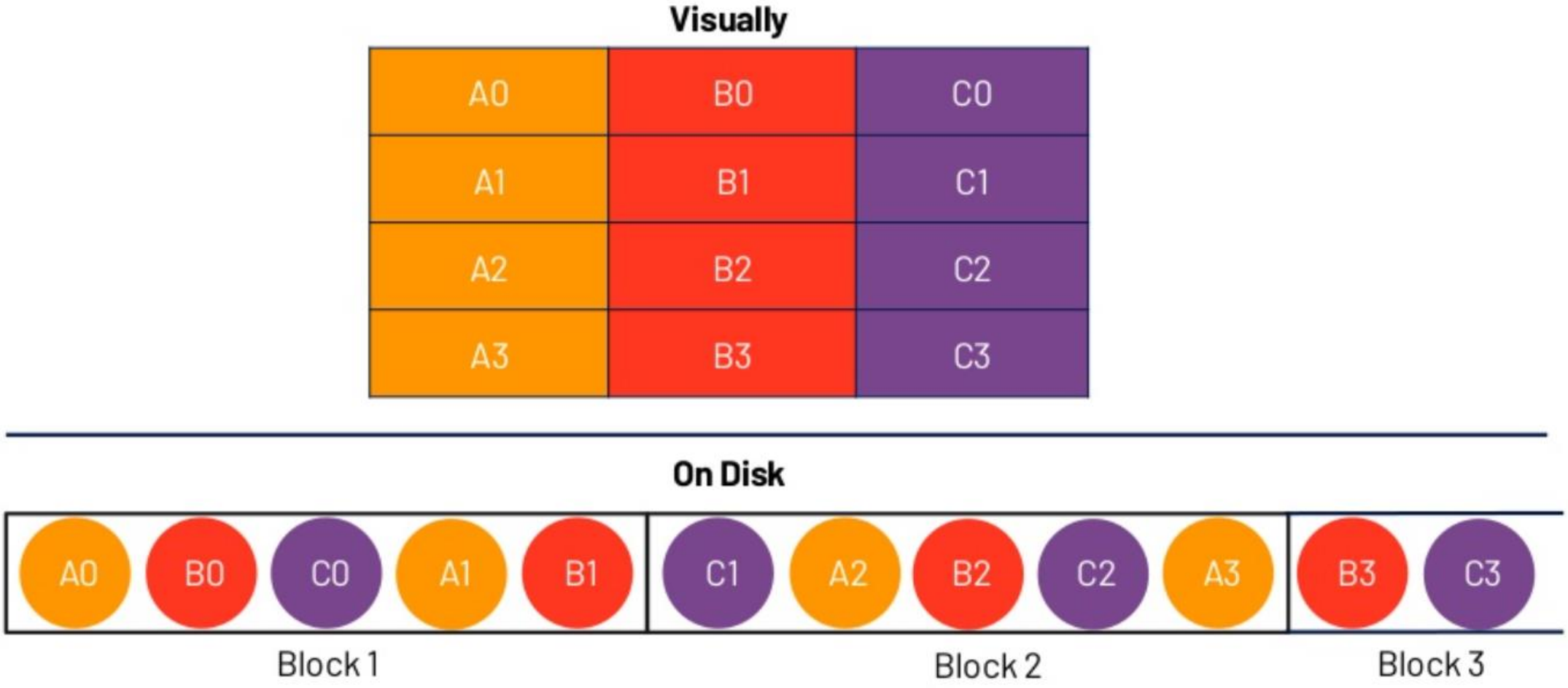
Big Data File Format

File format

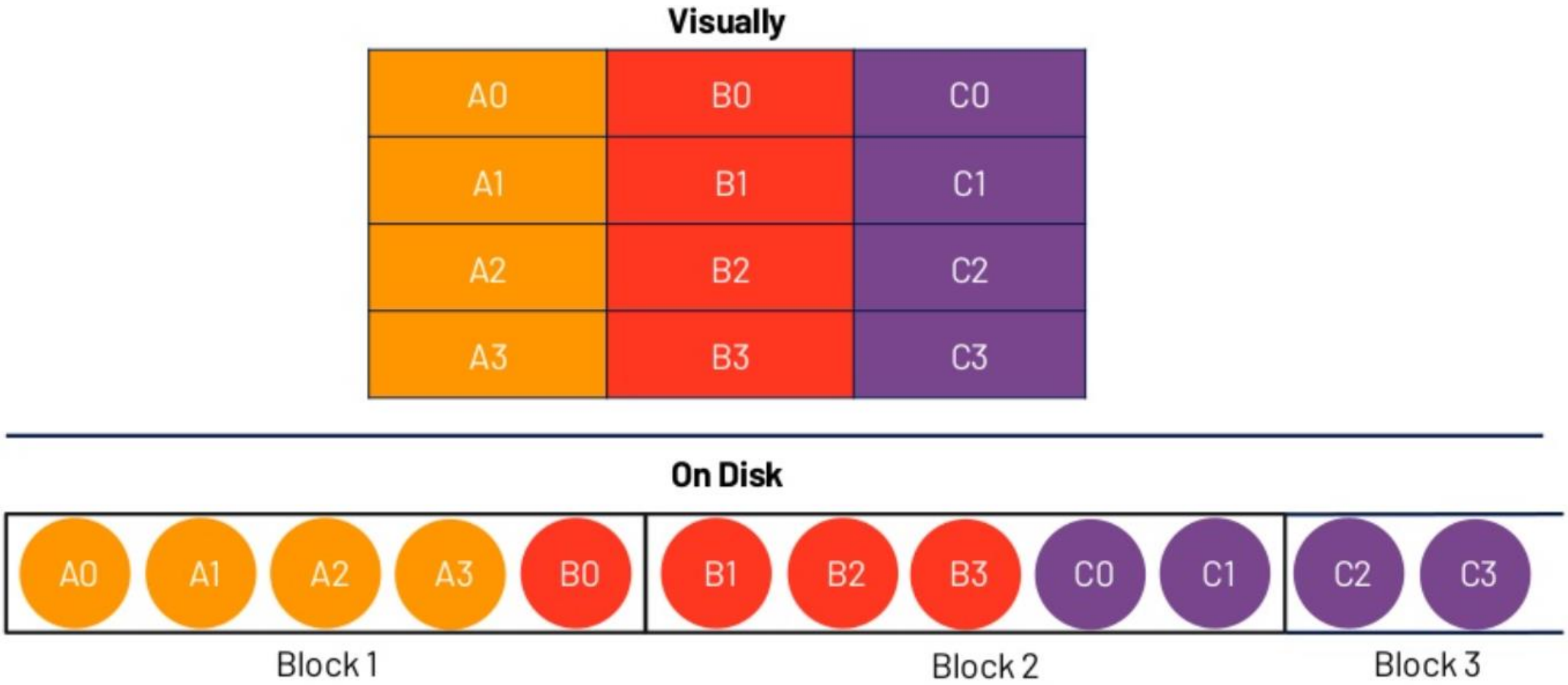

Unstructured
Text
CSV *
TSV *
Semi-Structured
JSON
XML
Structured
Avro
ORC
Parquet

Example Data

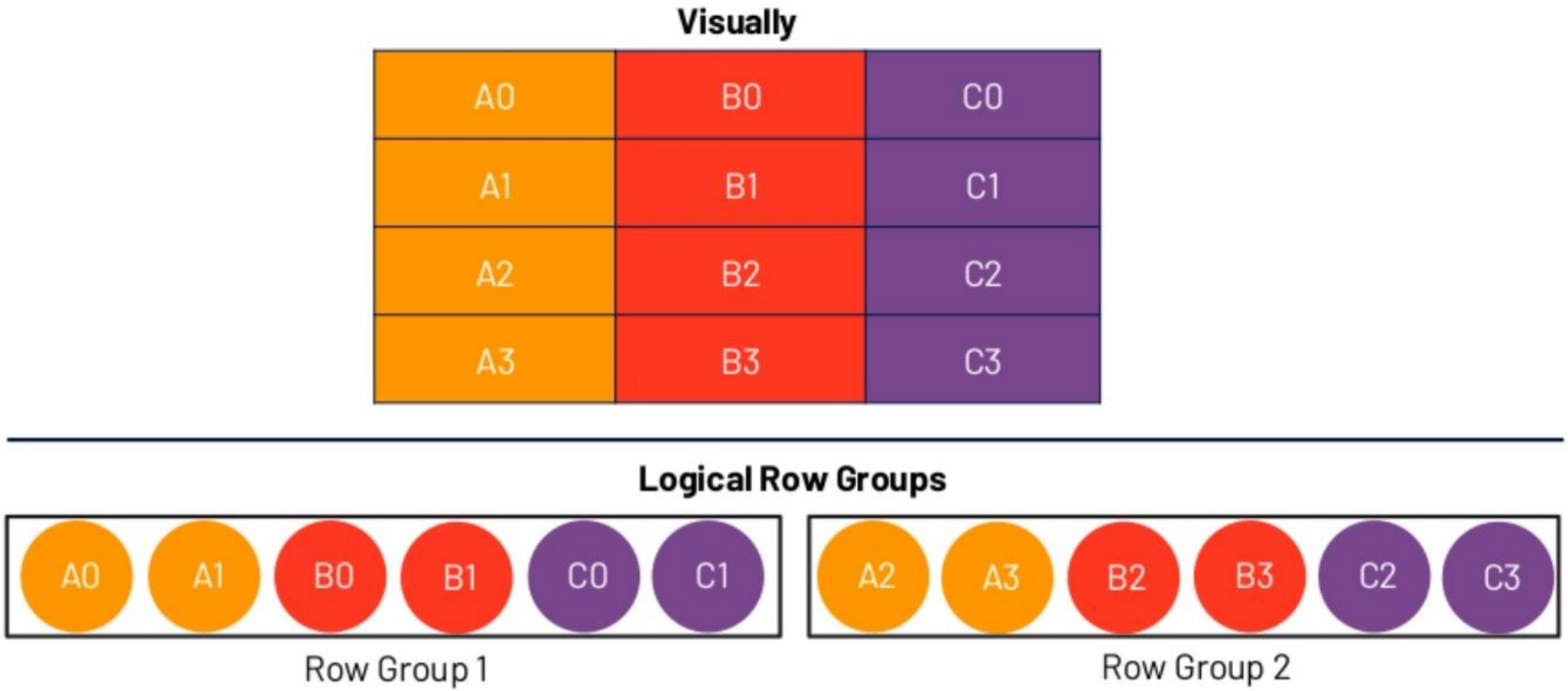
Row Wise


Visually
A0
B0
C0
A1
B1
C1
A2
B2
C2
A3
B3
C3
On Disk
A0
B0
C0
A1
B1
C1
A2
B2
C2
A3
B3
C3
Block 1
Block 2
Block 3

Column Wise

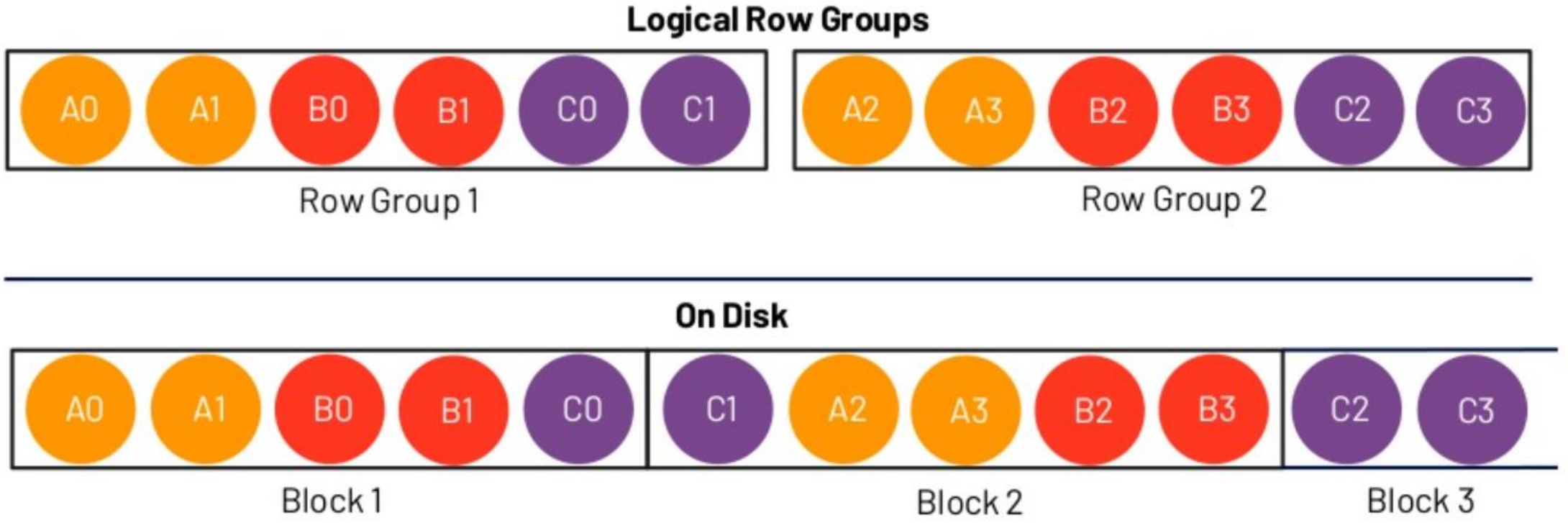

Visually
A0
B0
C0
A1
B1
C1
A2
B2
C2
A3
B3
C3
On Disk
A0
A1
A2
A3
B0
B1
B2
B3
C0
C1
C2
C3
Block 1
Block 2
Block 3

Hybrid (1)


Visually
A0
B0
C0
A1
B1
C1
A2
B2
C2
A3
B3
C3
Logical Row Groups
A0
A1
B0
B1
C0
C1
A2
A3
B2
B3
C2
C3
Row Group 1
Row Group 2

Hybrid (2)

In Parquet – aim to fit one row group in one block



About: CSV

Comma Separated Value (CSV)

CSV developed by IBM in 1972

- Ease of typing CSV lists on punched cards

Flexible (not always good)

Row-based

Human Readable

Compressible

Splittable

- When raw / using spittable format

Supported Natively

Fast (from a write perspective)

*Some formatting applied

```
$ cat myTable.csv
"student_id","subject","score"
71,"math",97.44
33,"history",88.32
101,"geography",73.11
13,"physics",87.78

scala> val table = spark.read.option("header","true")
.option("inferSchema", "true").csv("myTable.csv")
table: org.apache.spark.sql.DataFrame = [student_id: int,
subject: string ... 1 more field]

scala> table.printSchema
root
 |-- student_id: integer (nullable = true)
 |-- subject: string (nullable = true)
 |-- score: double (nullable = true)

scala> table.show
+----------+---------+-----+
|student_id|  subject|score|
+----------+---------+-----+
|        71|     math|97.44|
|        33|  history|88.32|
|       101|geography|73.11|
|        13|  physics|87.78|
+----------+---------+-----+
```

About: Parquet

Originally built by Twitter and Cloudera

Self-Describing

Hybrid-Based (rows grouped by row groups, then column partitioned)

- Optimized for read-intensive applications

Binary Format – Schema stored inside of file

Compressible

Splittable

Supported by natively in Spark

Supports rich data structures

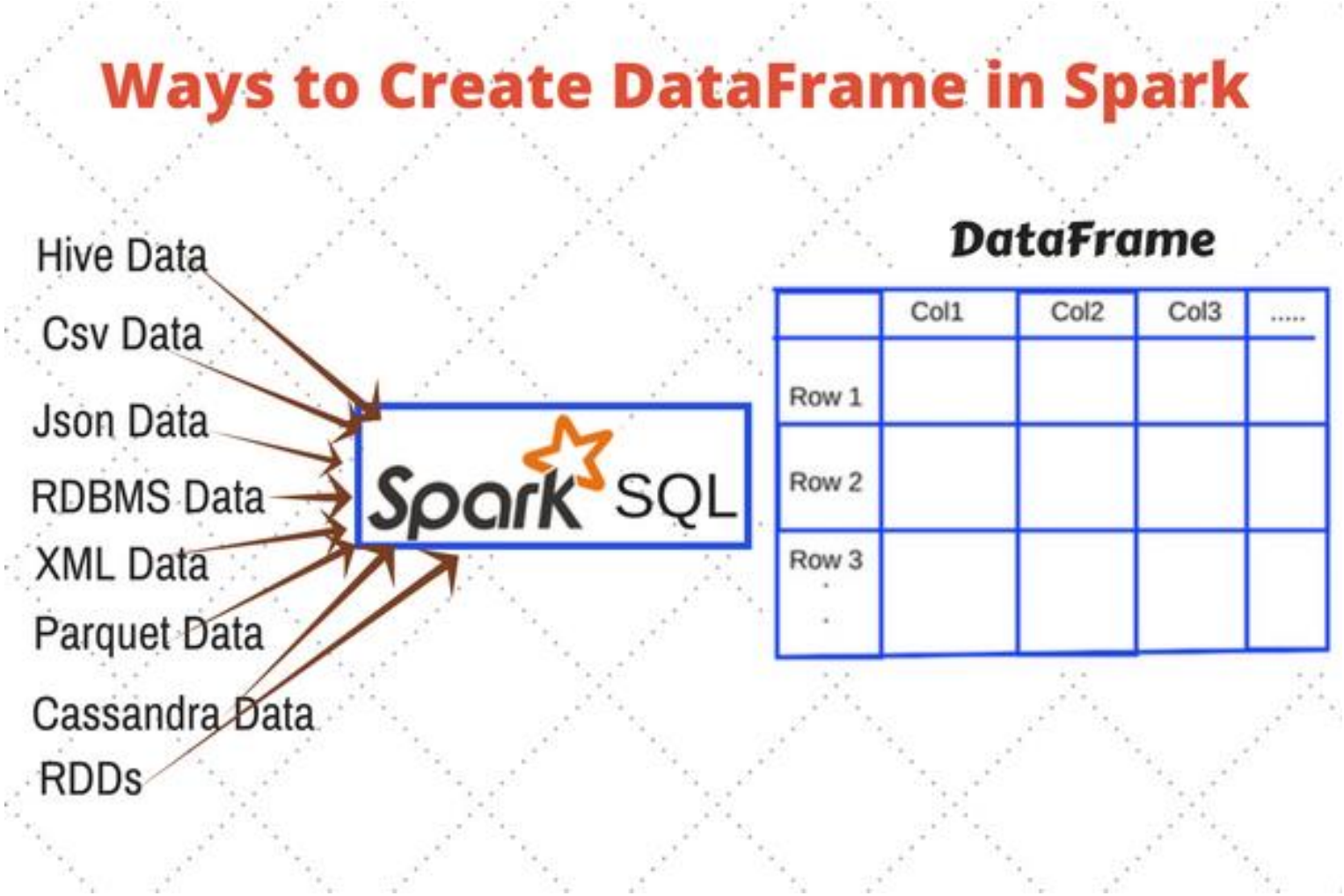
Performance Test

https://luminousmen.com/post/big-data-file-formats

4.
DF Writer/ Reader

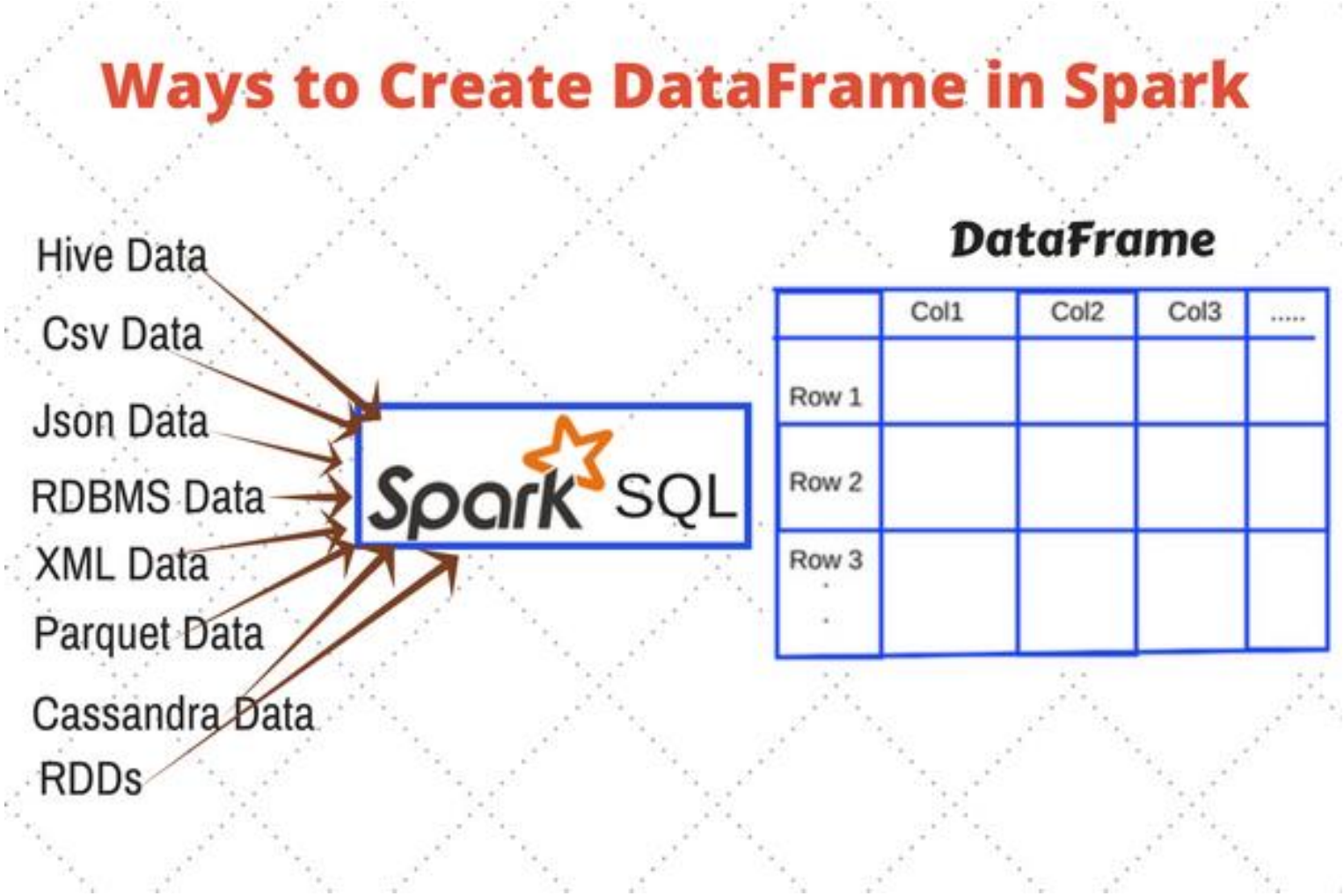
DataFrameReader

- Bắt đầu làm việc với Spark SQL sẽ là tạo DataFrame, DataSet dữ liệu.
- Spark có sẵn đối tượng DataFrameReader hỗ trợ tạo DF từ nhiều nguồn khác nhau.



DataFrameWriter

- Ghi dữ liệu của DataFrame ra các hệ thống lưu trữ bên ngoài
- Hỗ trợ nhiều định dạng.



5.
Thực hành