

Buổi 2:

Spark SQL và Spark 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

Nguyễn Chí Thanh – Ban Quản trị Dữ liệu



1. Spark SQL Overview

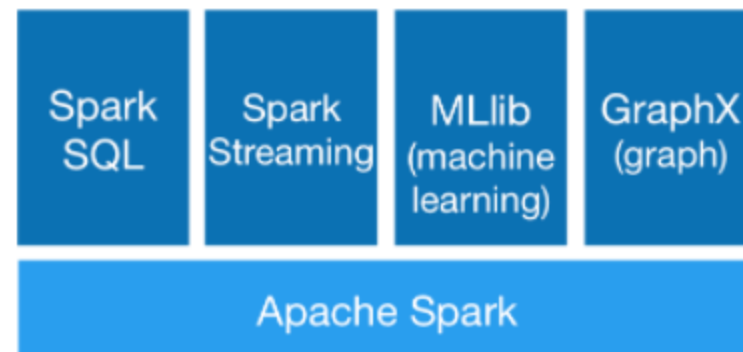


Spark SQL (1)

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*

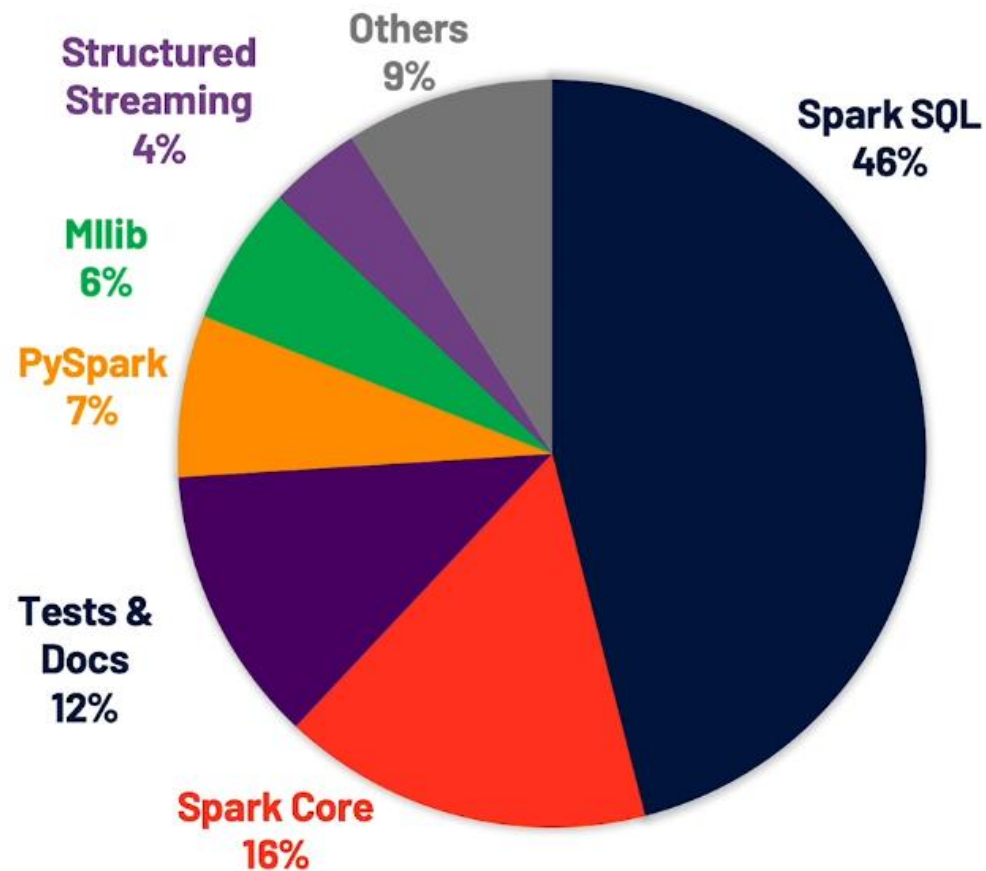
Spark SQL (2)

- Là module "sôi động" nhất trong toàn bộ thư viện

Apache Spark 3.0

3400+ patches from community

Easy to switch to from Spark 2.x

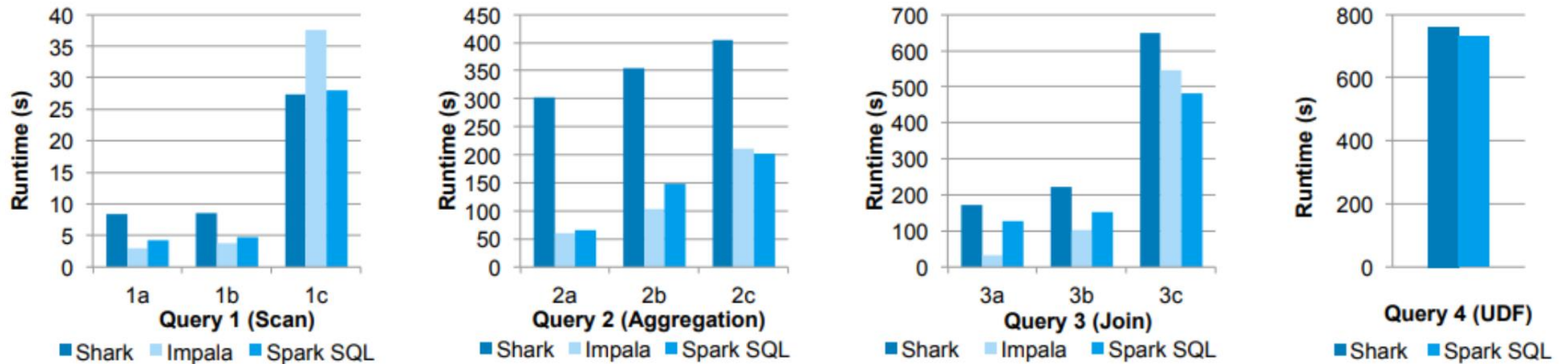


Spark SQL (3)

- 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.



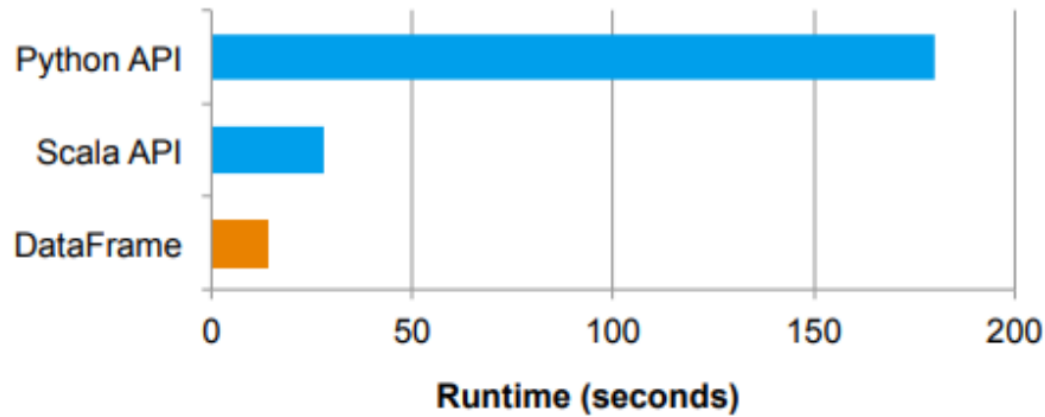
Spark SQL Performance (1)



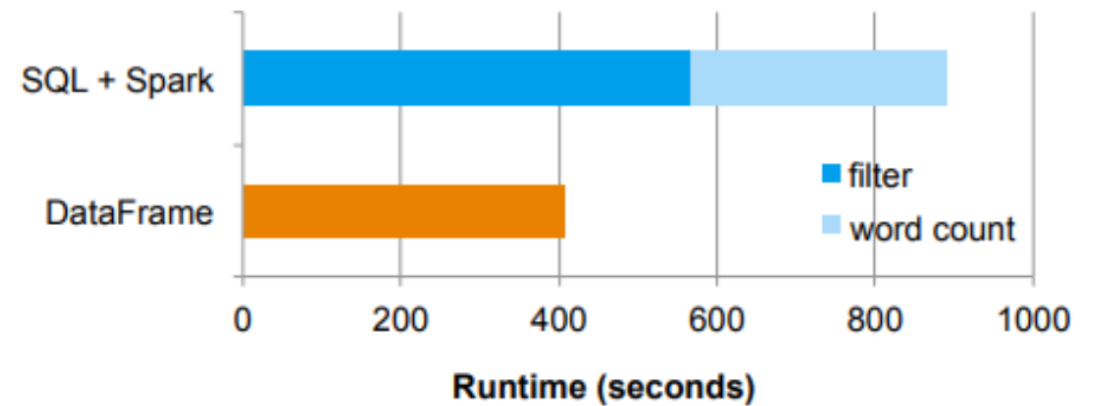
Performance of Shark, Impala and Spark SQL on Big Data benchmark queries

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.

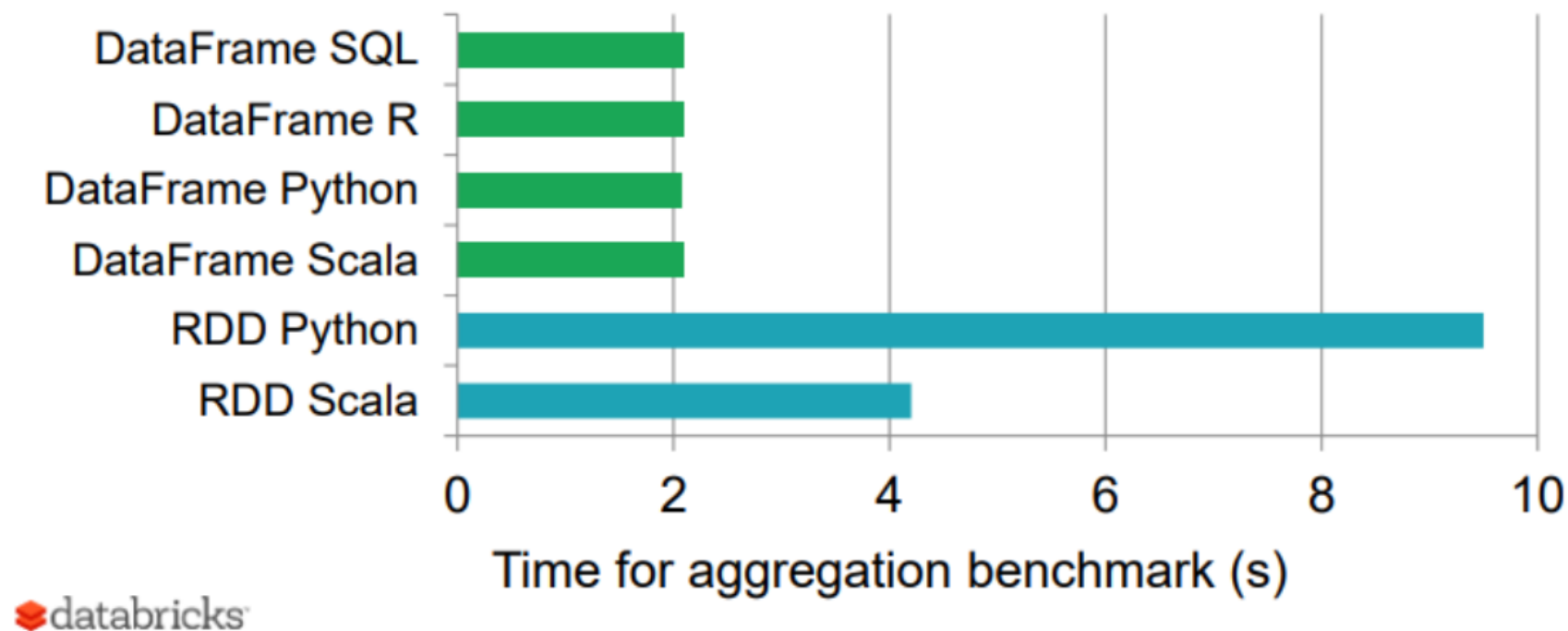


Performance of a two-stage pipeline written as a separate Spark SQL query and Spark job (above) and an integrated DataFrame job (below).

Spark DataFrames vs RDDs and SQL

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

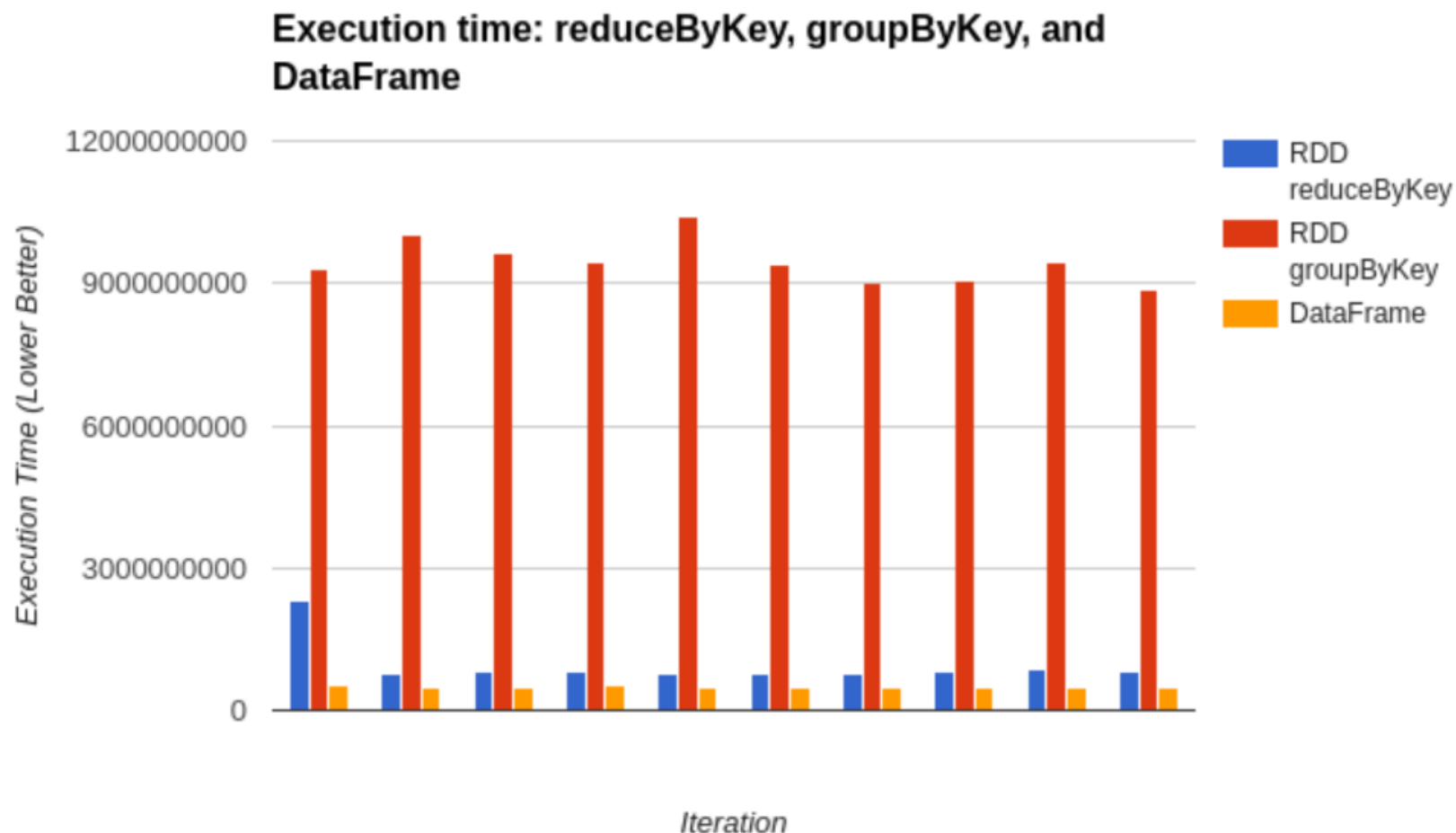
Spark SQL Performance (3)



Comparing Spark DataFrames and RDDs

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

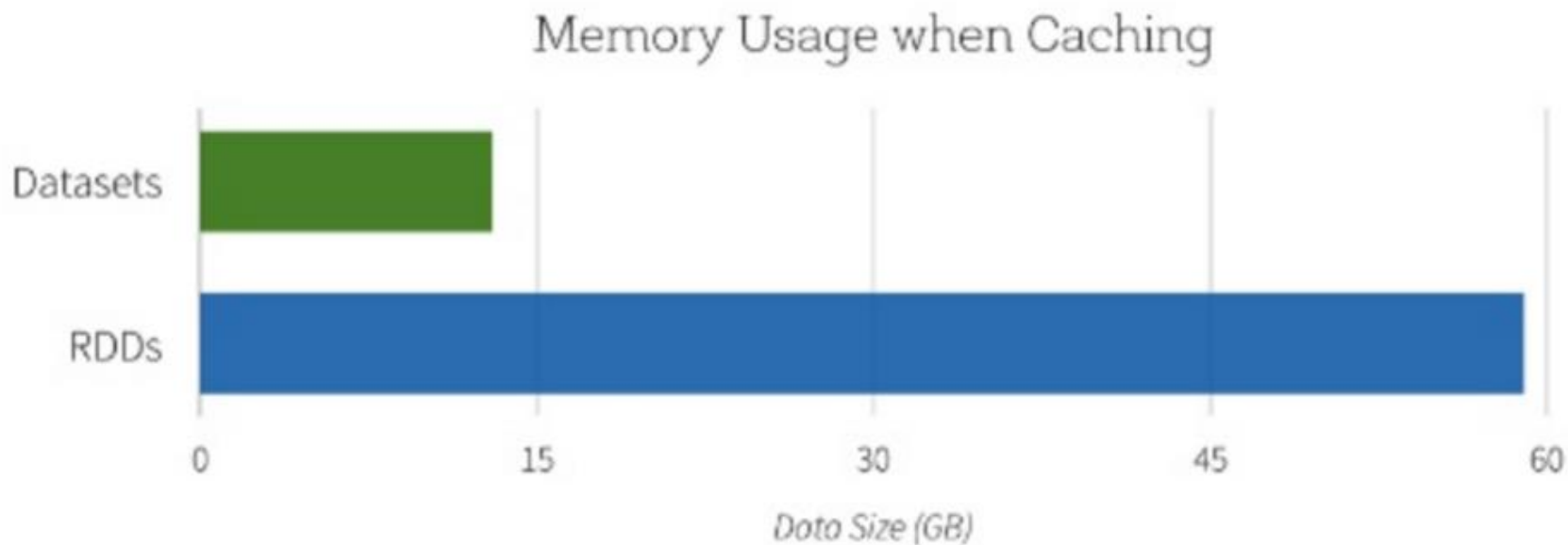
Spark SQL Performance (4)



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

Spark SQL Performance (5)

Space Efficiency



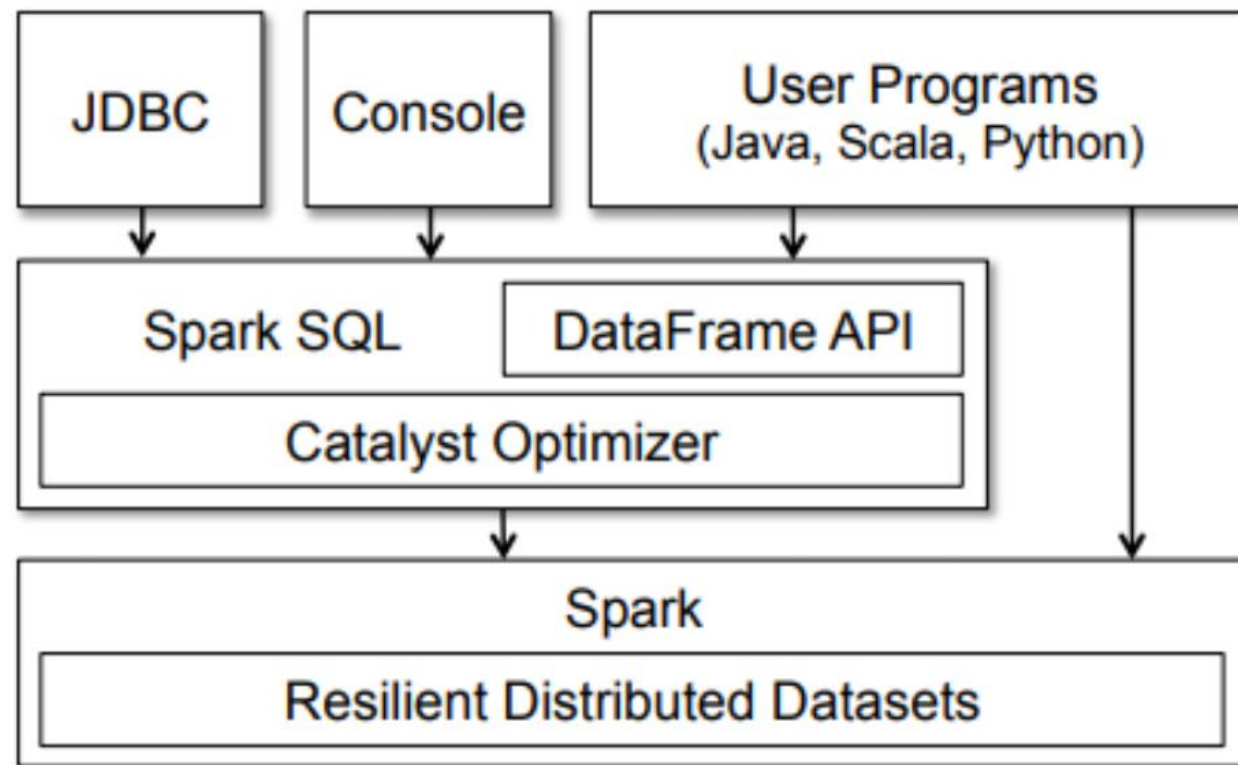
Khả năng sử dụng memory của Spark SQL (Dataset) tốt hơn so với Spark Core RDD

2. Spark SQL Components



Spark SQL Components

- 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

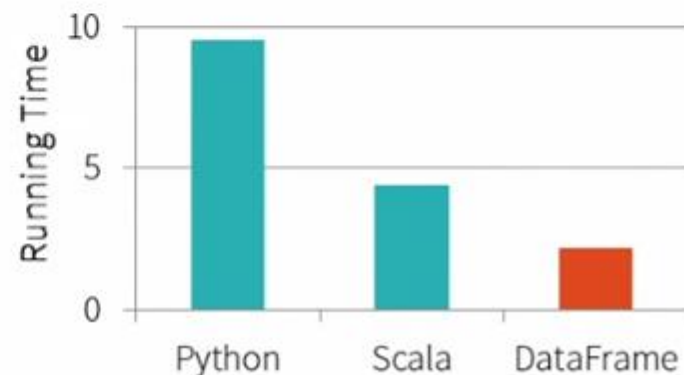
Similar API to data frames in R and Pandas

Automatically optimized via Spark SQL

Coming in Spark 1.3

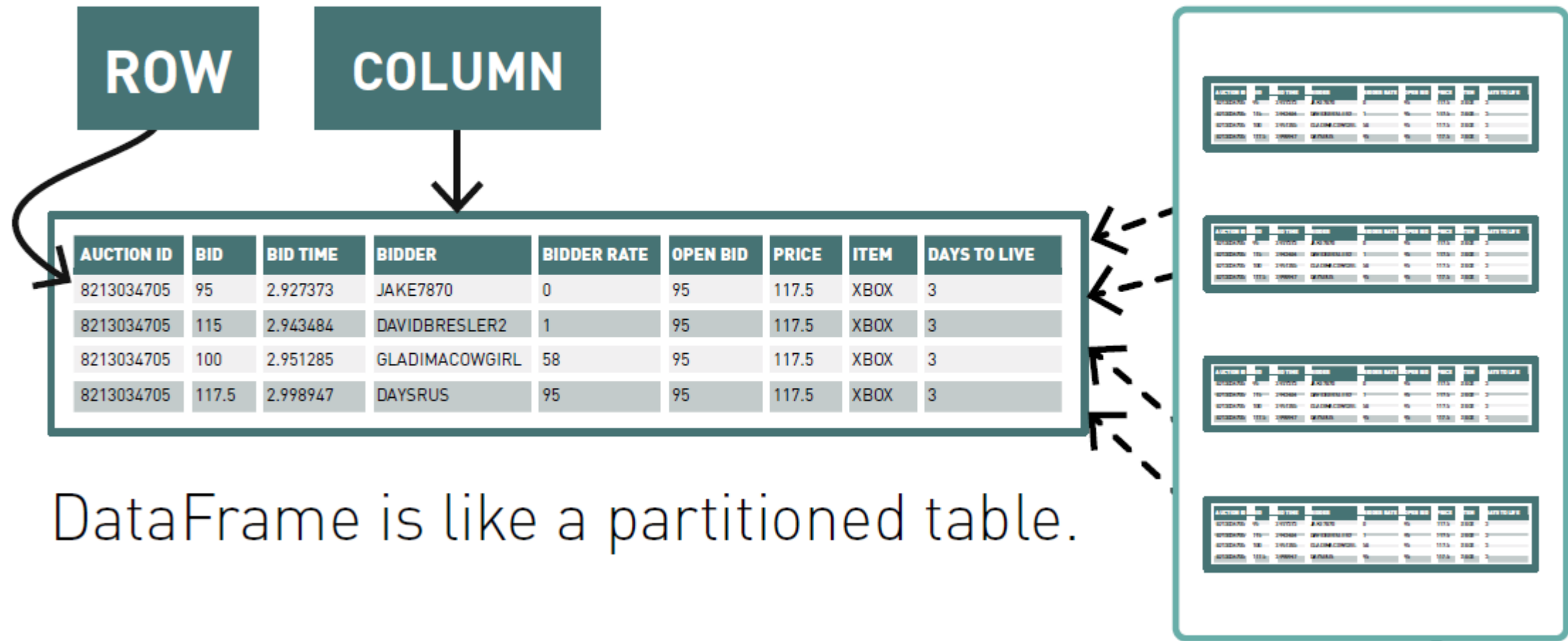
```
df = jsonFile("tweets.json")
```

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



DataFrame (2)

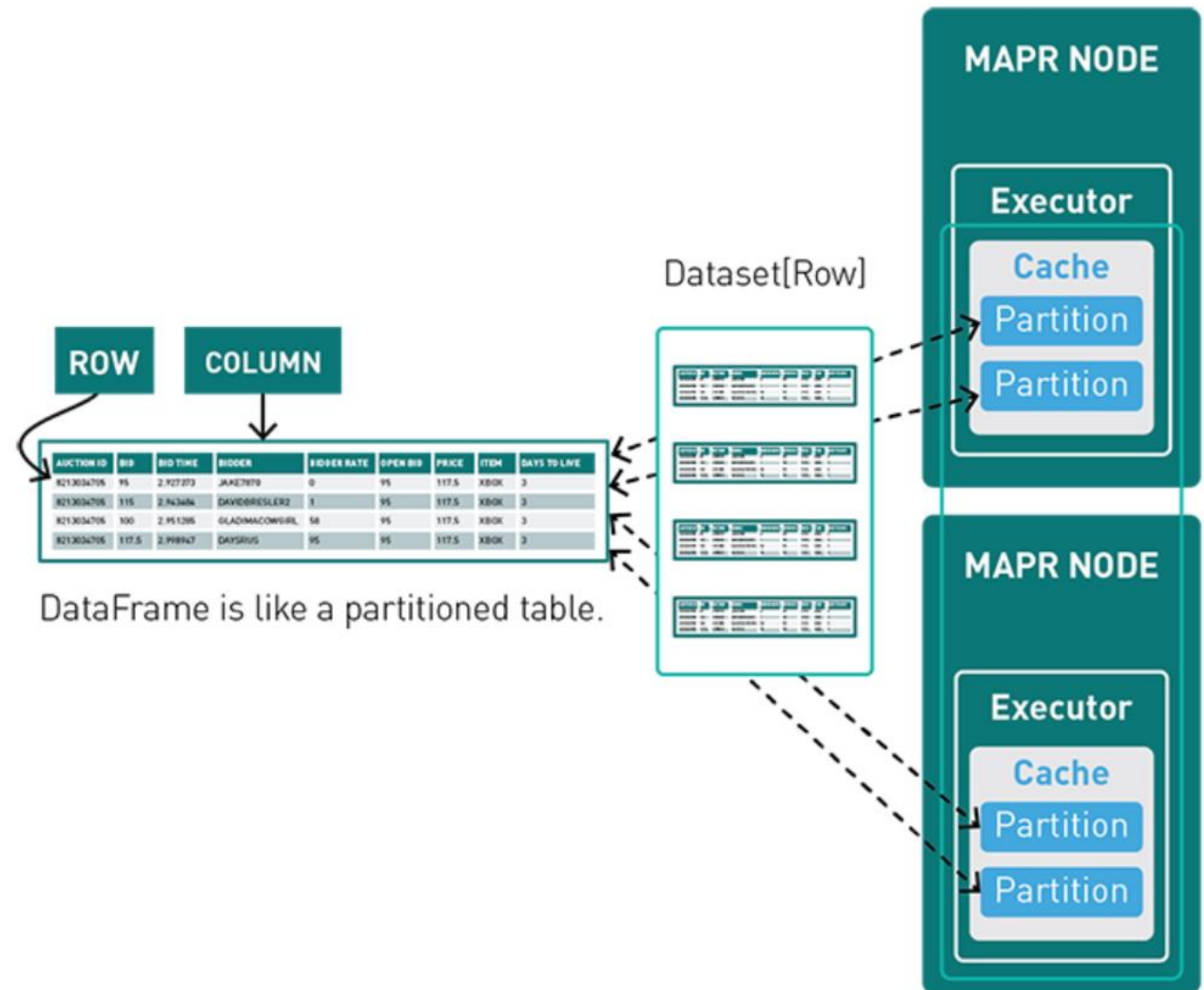
Dataset[Row]



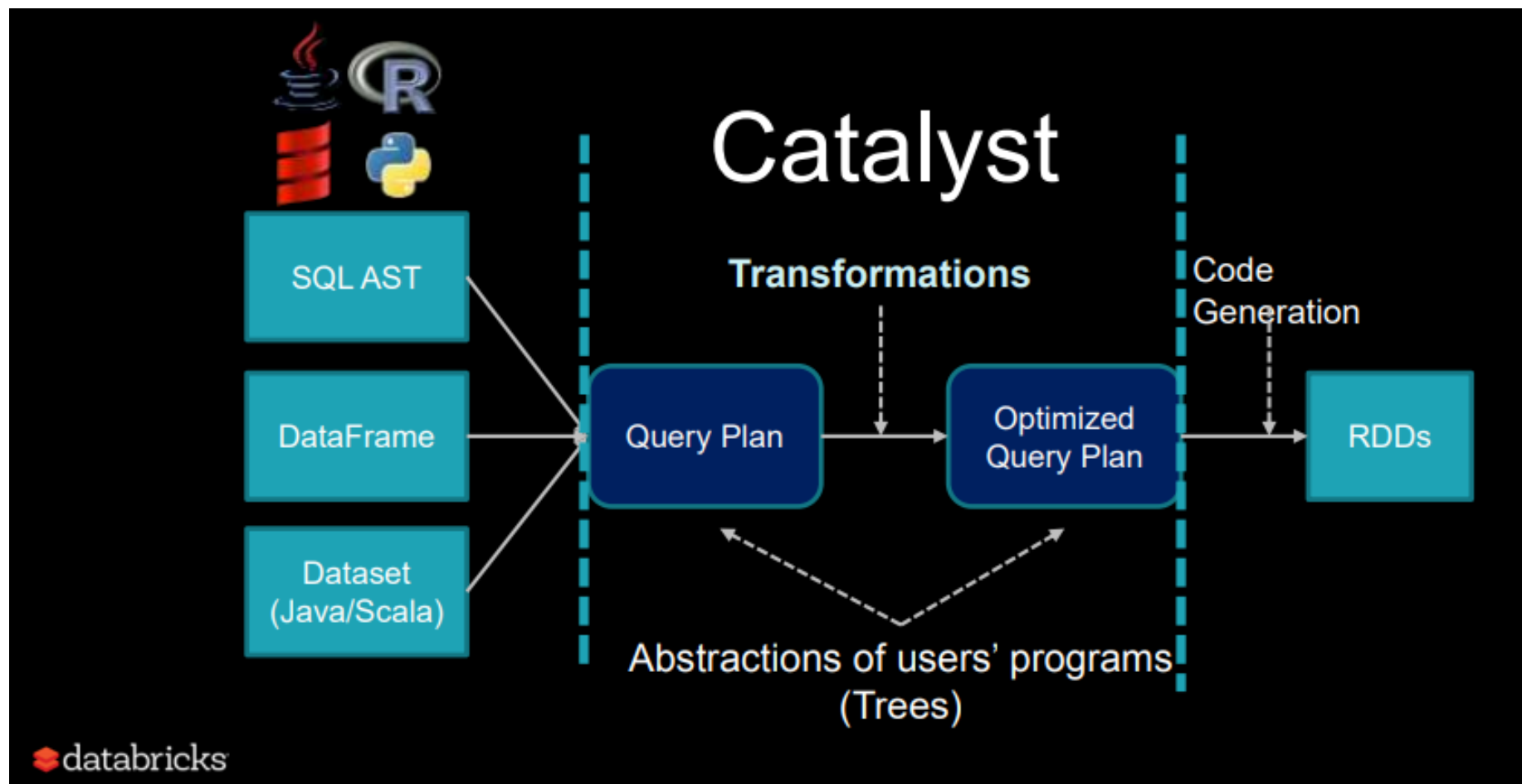
DataFrame is like a partitioned table.

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)



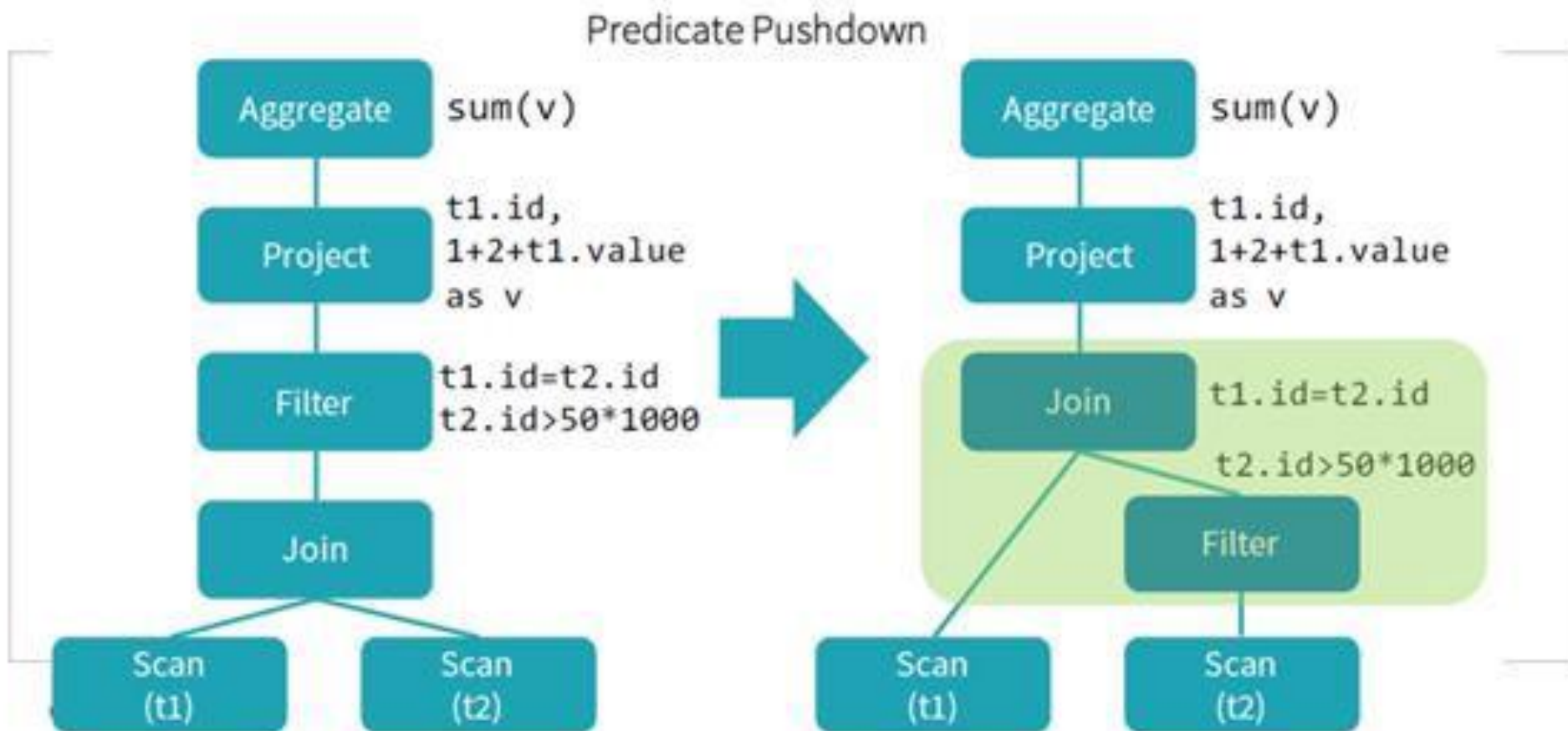
Catalyst Optimizer (2)

- Giả sử bảng t2 cũng có **1 tỷ** bản ghi nhưng chỉ có 100 bản ghi thỏa mãn điều kiện $t2.id > 50 * 1000$. Lẽ nào vẫn phải join full sau đó lại lọc 900k kết quả không thỏa mãn?

```
SELECT sum(v)
FROM (
  SELECT
    t1.id,
    1 + 2 + t1.value AS v
  FROM t1 JOIN t2
  WHERE
    t1.id = t2.id AND
    t2.id > 50 * 1000) tmp
```

Catalyst Optimizer (3)

Combining Multiple Rules



Catalyst Optimizer (4)

```
events =  
  sc.read.json("/logs")  
  
stats =  
  events.join(users)  
    .groupBy("loc", "status")  
    .avg("duration")  
  
errors = stats.where(  
  stats.status == "ERR")
```

DataFrame API



Optimized Plan

```
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  
  }  
}  
...
```

Specialized Code

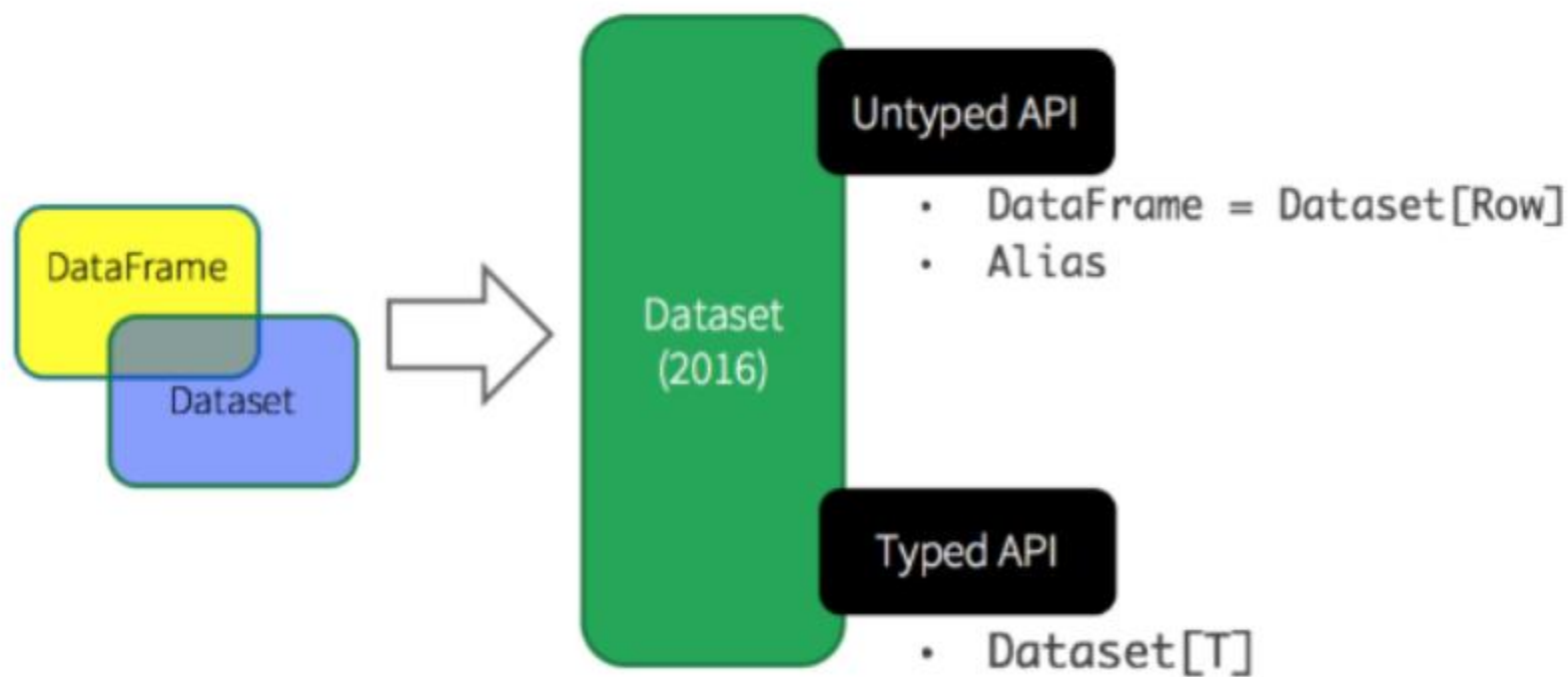
3.

DataFrame >< DataSet




DataSet (1)

Unified Apache Spark 2.0 API



DataSet (1)



	SQL	DataFrames	DataSets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time



THẢO LUẬN

- Các hệ CSDL tại đơn vị.
- Mục đích sử dụng của từng loại, tại sao lại dùng CSDL đó.
- Khả năng đáp ứng nhu cầu.
- Các bất cập trong sử dụng.
- Hướng giải quyết.

4. Spark File Format



File format

Unstructured

- Text
- **CSV** *
- TSV *

Semi-Structured

- **JSON**
- XML

Structured

- **Avro**
- **ORC**
- **Parquet**

Example Data

	Column A	Column B	Column C
Row 0	A0	B0	C0
Row 1	A1	B1	C1
Row 2	A2	B2	C2
Row 3	A3	B3	C3

Row wise

Visually

A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3

On Disk



Column wise

Visually

A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3

On Disk



Hybrid

Visually

A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3

Logical Row Groups



Row Group 1



Row Group 2

Hybrid

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

Logical Row Groups



On Disk



CSV

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|
+-----+-----+-----+
```

Parquet

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>

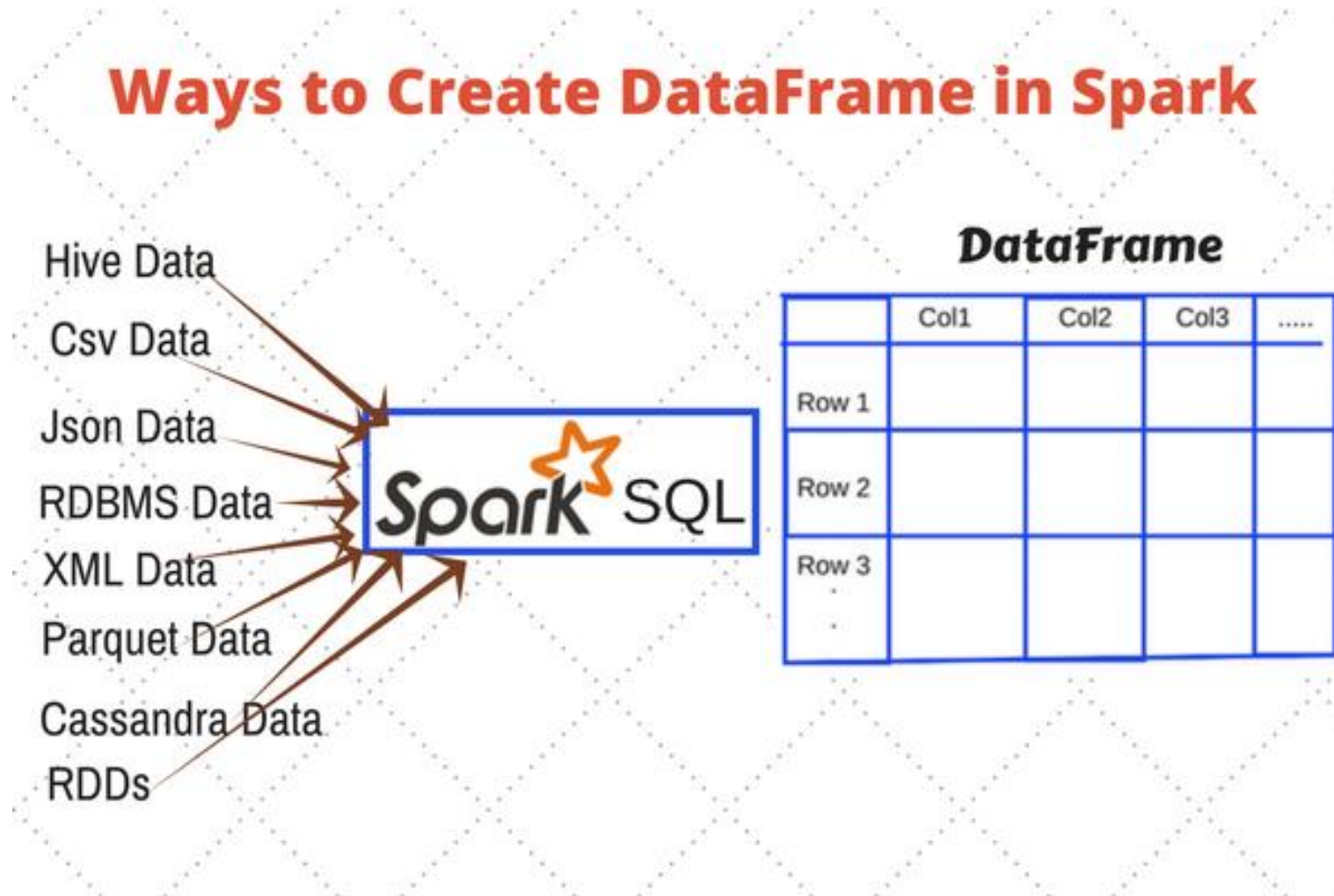
5.

Đọc ghi dữ liệu với DataFrameReader/ Writer



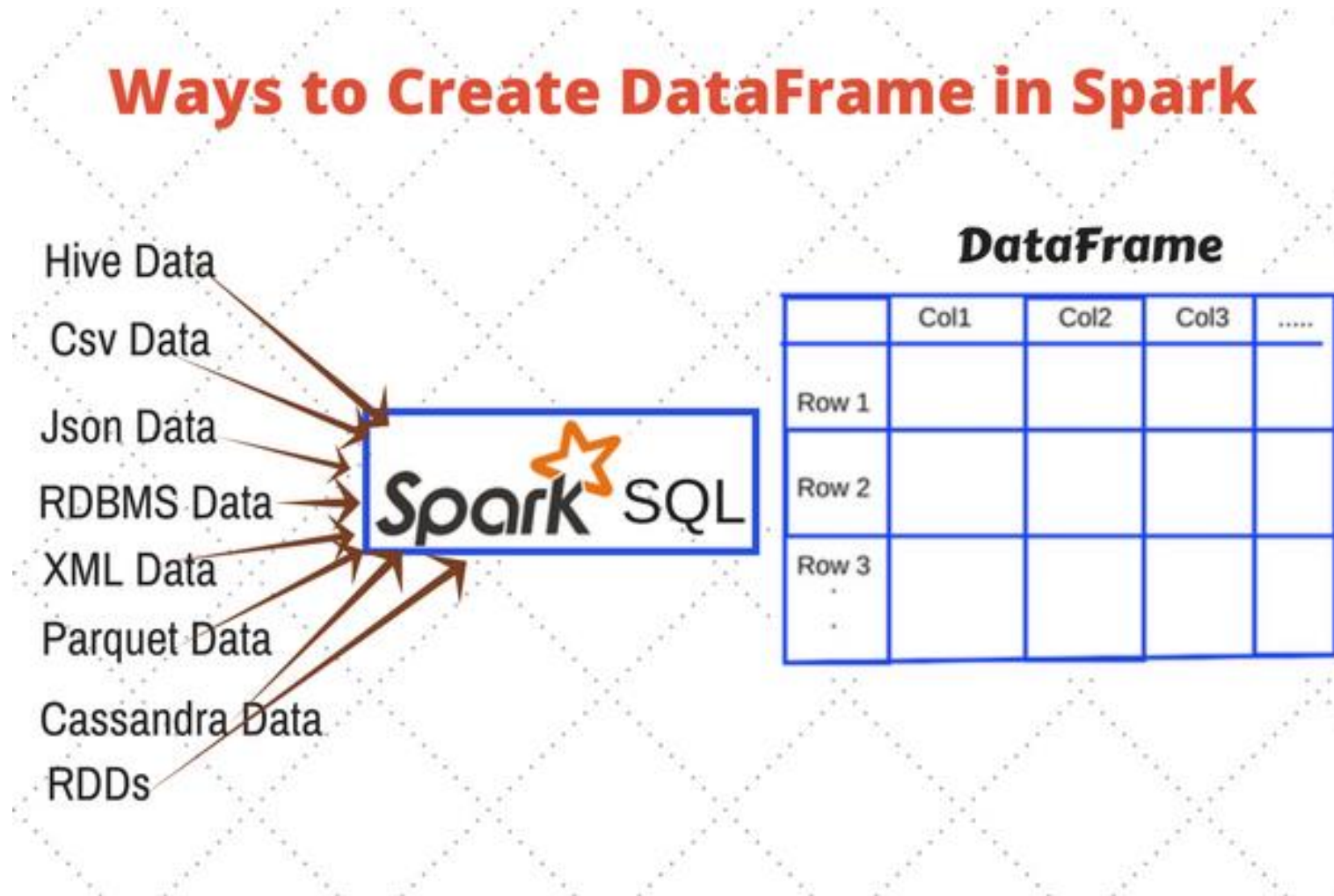
DataFrameReader

- 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.



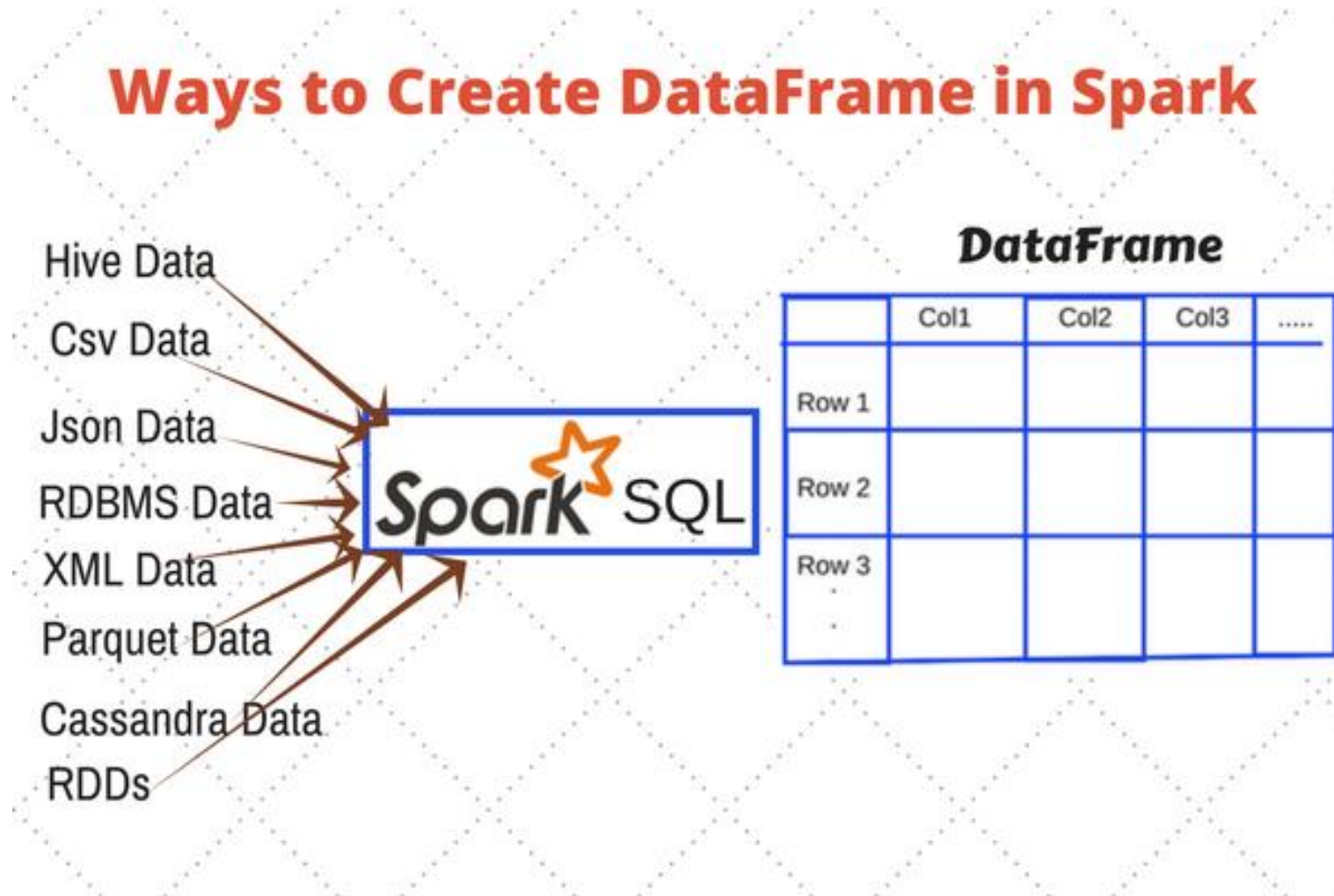
DataFrameWriter

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



DataFrameWriter

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



5. Joins in Spark



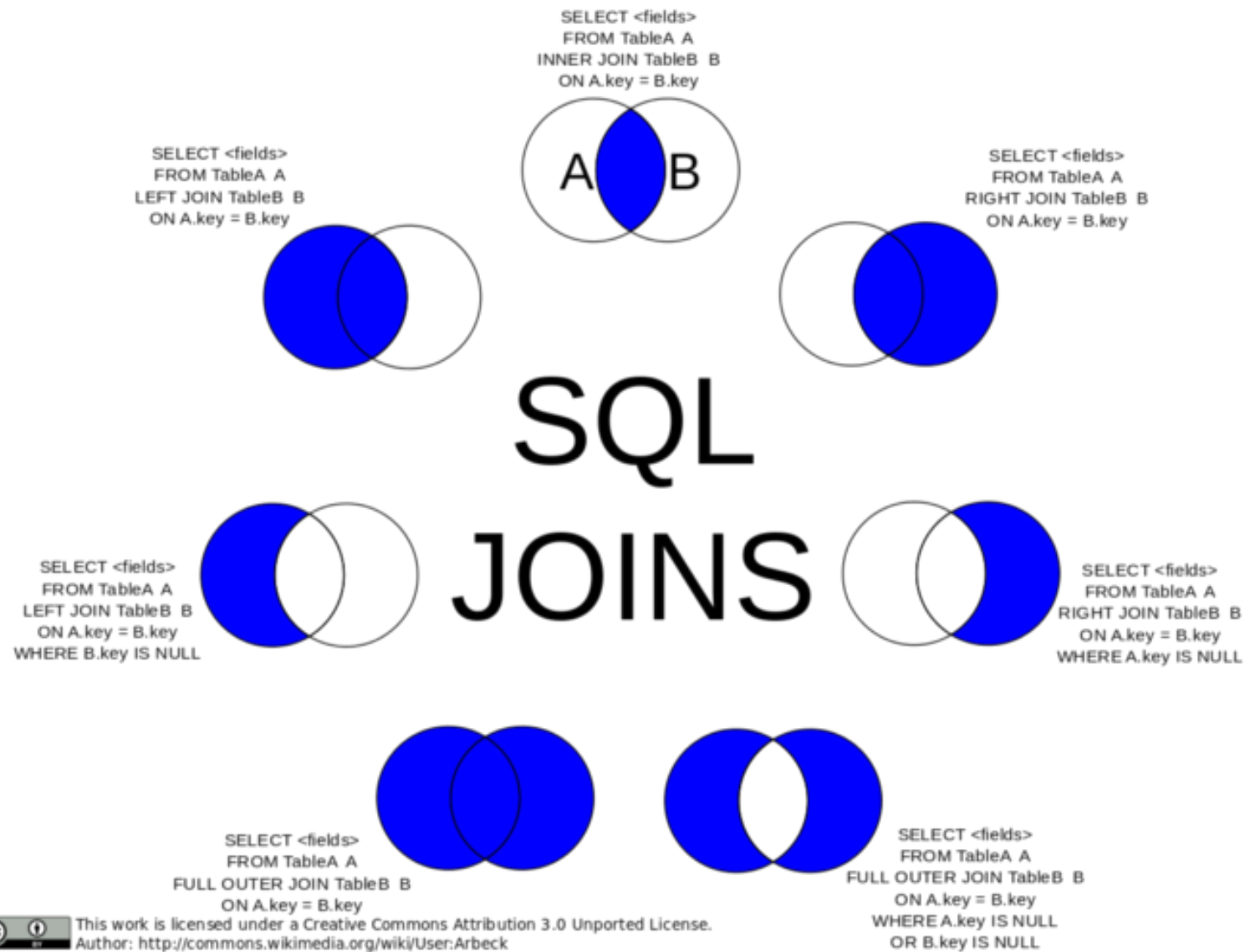
Joins

- SQL Join là việc kết hợp giữa 1 hay nhiều bảng (hoặc tập dữ liệu) để đưa ra 1 tập kết quả khác dựa vào 1 tiêu chí nào đó
- Tiêu chí để join thông thường là 1 biểu thức logic để match dữ liệu giữa các bảng hoặc tập dữ liệu (join expression).
- Là tính toán rất nặng và phức tạp

Các loại Joins (1)

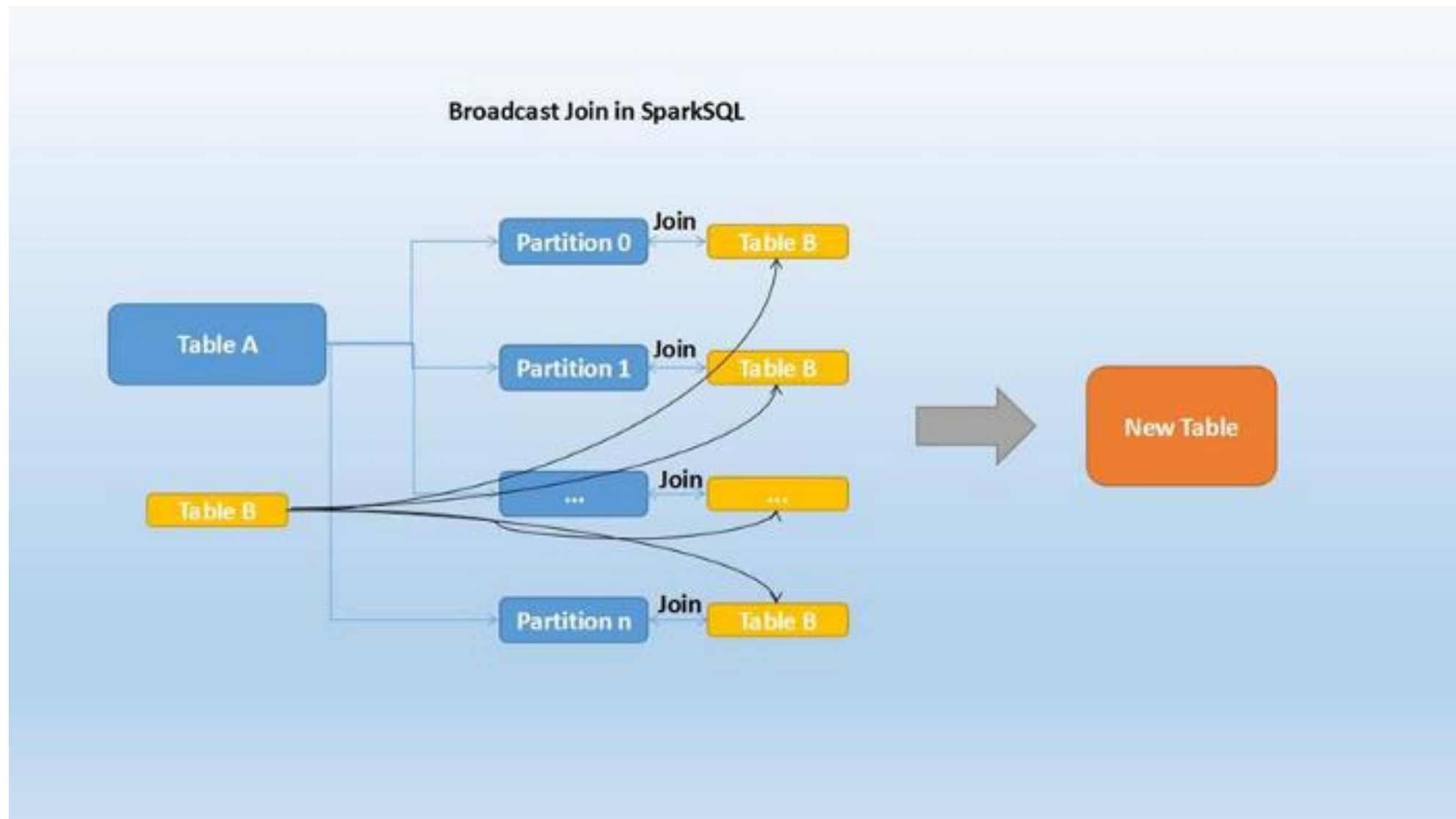
- Inner join (default): Trả về kết quả 2 cột nếu biểu thức join expression true
- Left outer join: Trả về kết quả bên trái kể cả biểu thức join expression false
- Right outer join: Ngược với Left
- Outer join: Trả về kết quả 2 bên
- Left anti join: Trả về kết quả bên trái khi join expression false
- Left semi join: Trả về kết quả bên trái khi join expression true
- Cross join (aka Cartesian): Join kiểu tích đề-các

Các loại Joins (2)



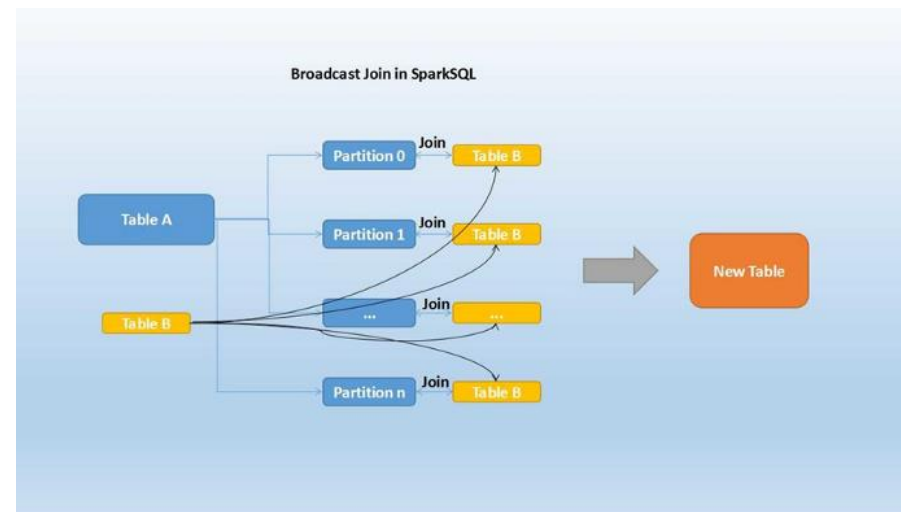
This work is licensed under a Creative Commons Attribution 3.0 Unported License.
 Author: <http://commons.wikimedia.org/wiki/User:Arbeck>

Broadcast Join (1)



Broadcast Join (1)

- Thường có tốc độ nhanh.
- Phù hợp với việc join 1 bảng to với 1 bảng nhỏ hơn
- Cấu hình ngưỡng "nhỏ":
`spark.sql.autoBroadcastJoinThreshold`

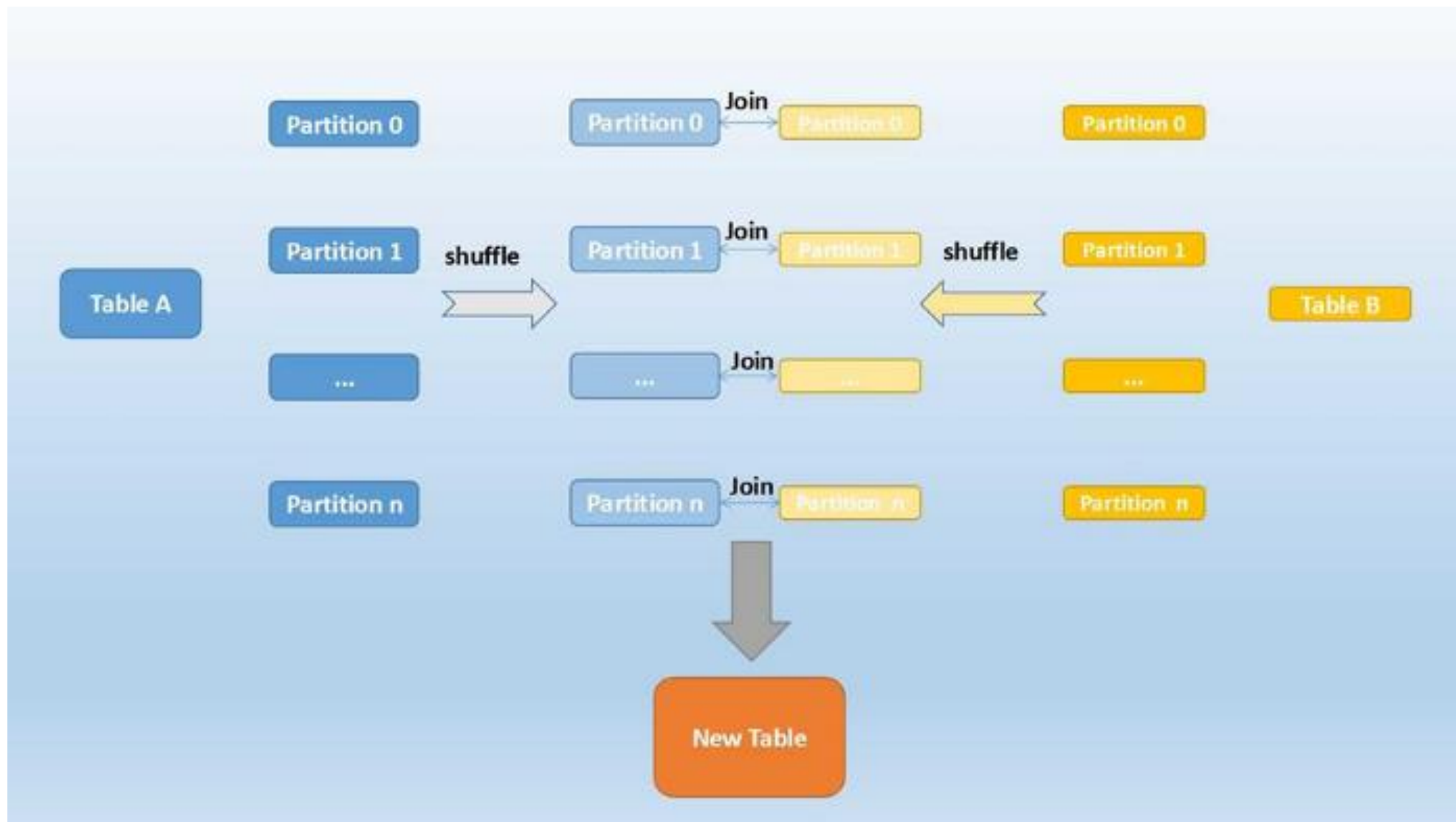


Shuffle

- Các executor khi muốn trao đổi dữ liệu, chúng phải gửi cho nhau qua mạng, quá trình đó gọi là Shuffle

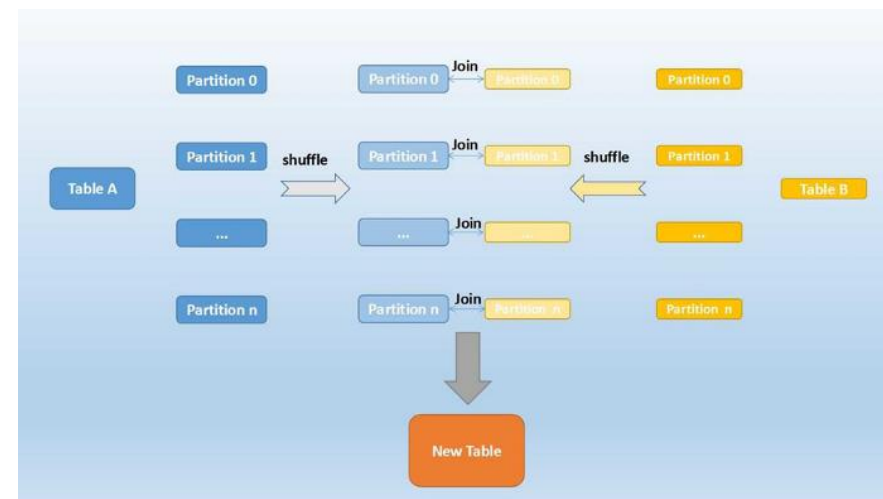


Shuffle Hash Join



Shuffle Hash Join

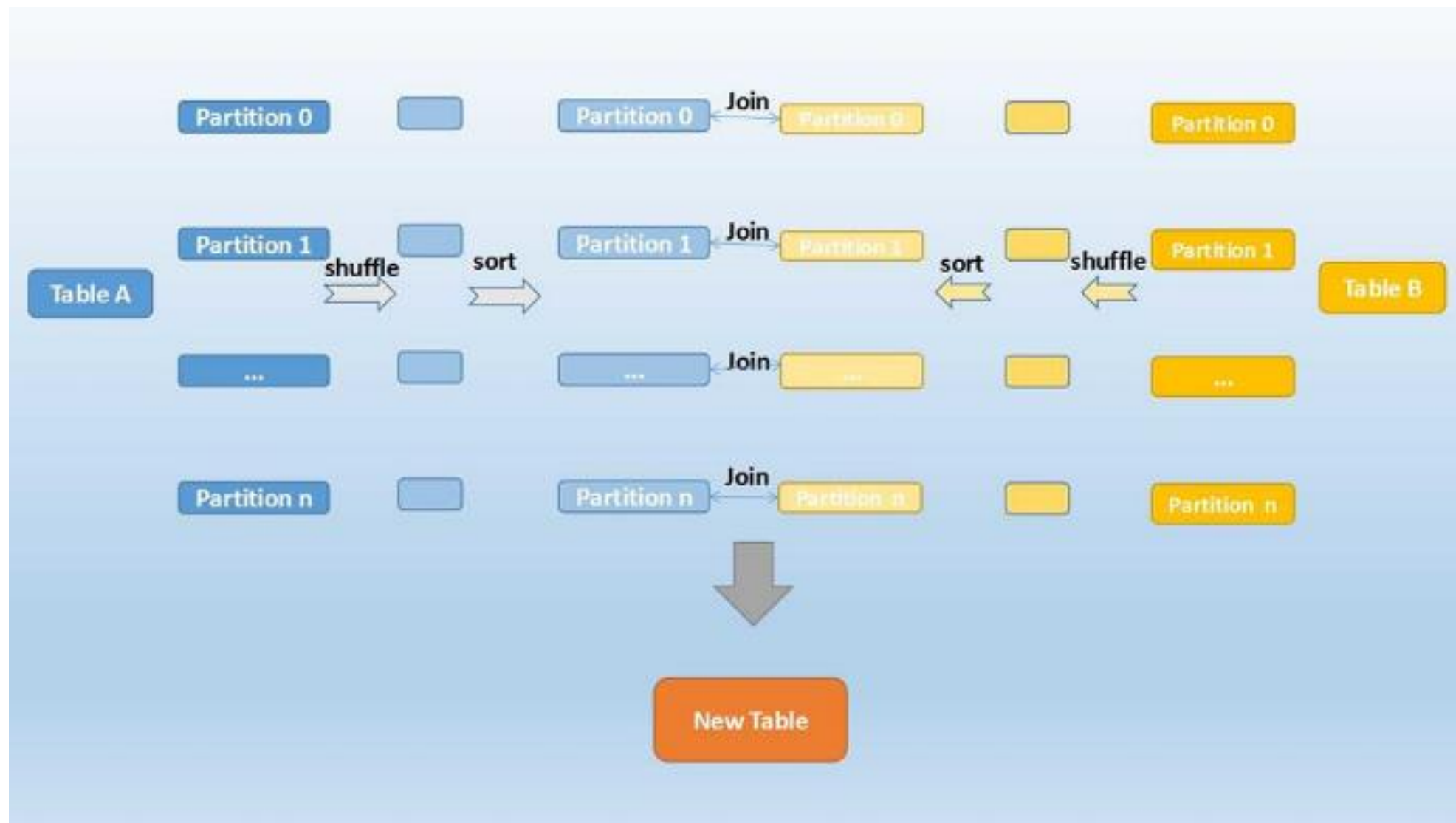
- VD ban đầu *partition0*, 1 chứa cả *key A* và *key B*.
- Khi join, Spark sẽ tạo thành 1 DF trung gian mà ở đó, *partition0* mới chỉ chứa *key A*, *partition1* chỉ chứa *key B*
- Quá trình di chuyển toàn bộ *key A* từ *partition0* cũ sang *partition0* mới chính là *shuffle*
- Việc move key này sẽ là key của cả 2 DataFrame cần join.



Shuffle in Shuffle Hash Join

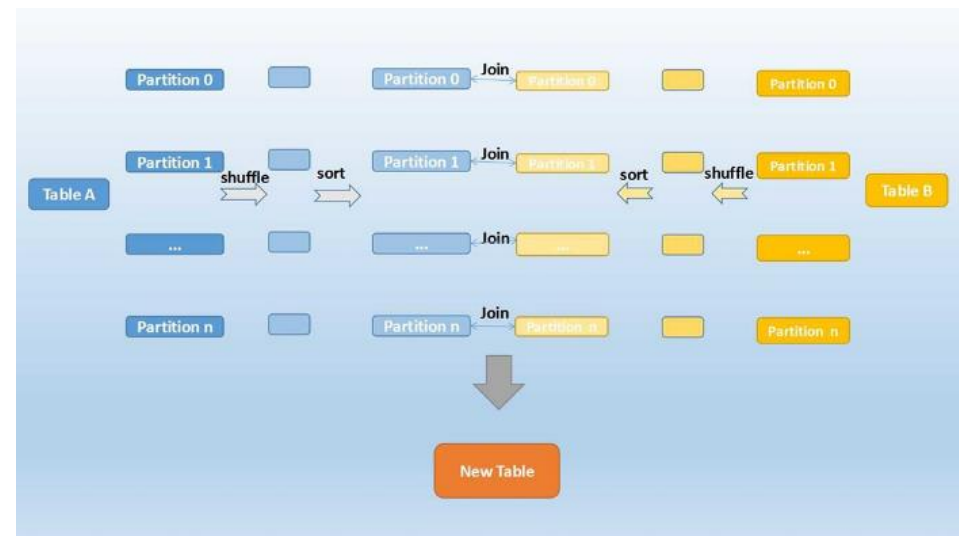
- Để xác định key = nhau, 1 phép hash được thực thi, trong máy tính, để so sánh A với B, người ta sẽ so sánh mã hash của A và B thay vì xem xét giá trị của nó.
- Để bản ghi được chuyển đi, dữ liệu sẽ cần serialize ở đầu gửi và deserialize ở đầu nhận.
- Hàng chục, hàng trăm executor phải mở các connection đến nhau để truyền dữ liệu.
- Băng thông mạng ảnh hưởng trực tiếp đến tốc độ shuffle.

Sort Merge Join



Sort Merge Join

- DF được sort theo key cần join
- Sau khi sort, thực thi như Shuffle Hash Join
- Tránh được trường hợp dữ liệu 1 key lại ở nhiều partitions.



6. Thực hành

