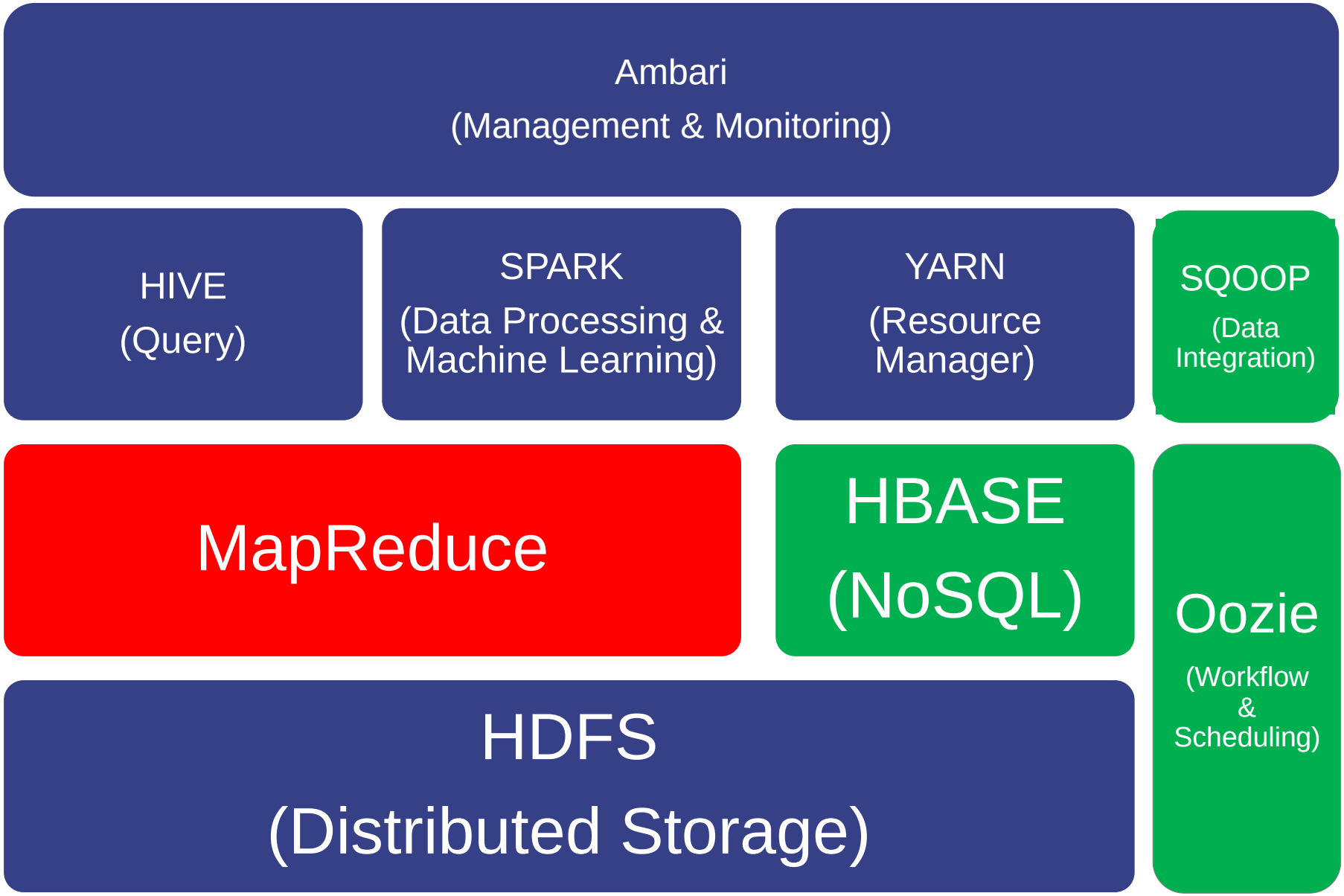

# Programming In Spark Using PySpark

## Data Engineering DLL

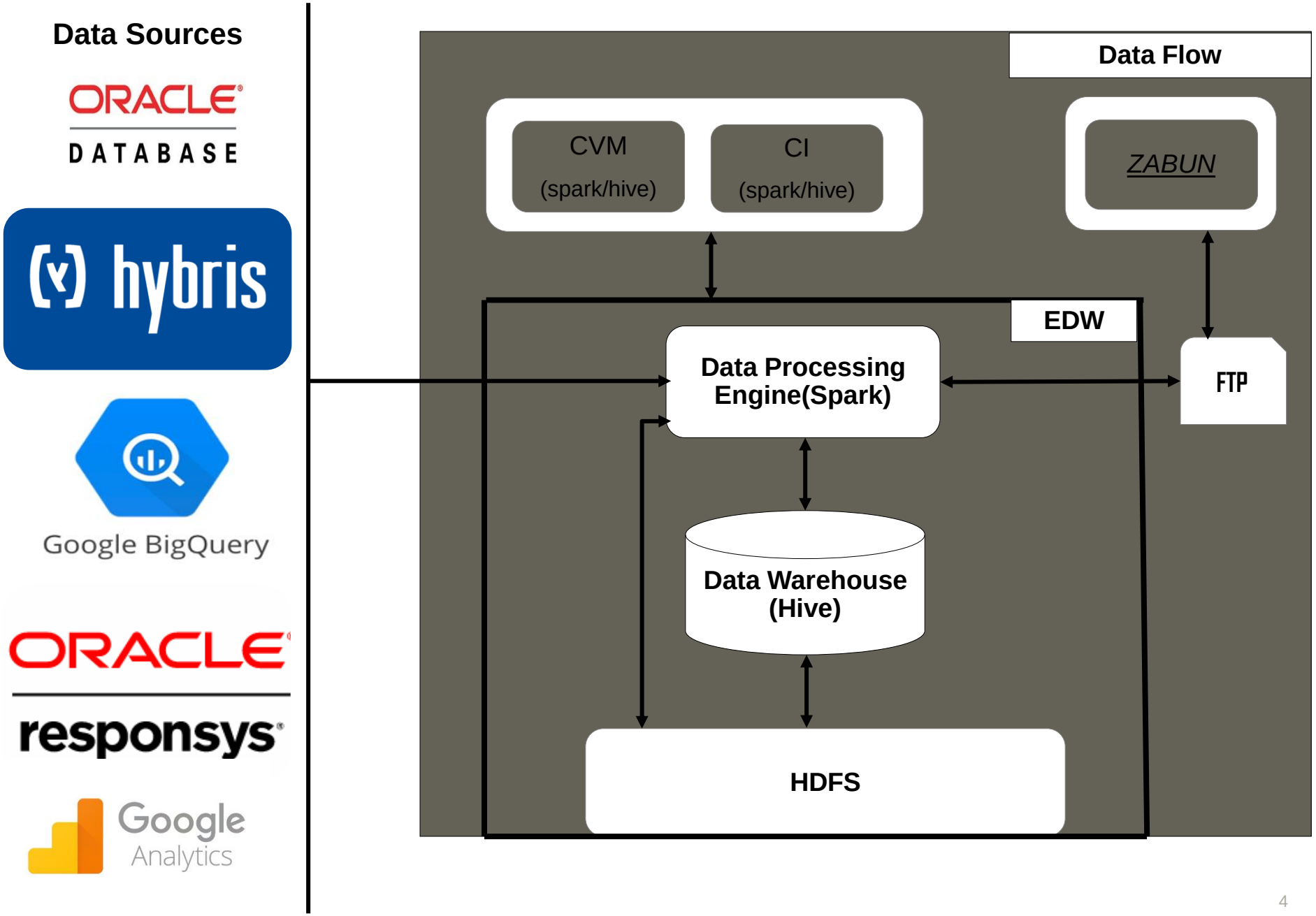
## Content

- DLL Hadoop Architecture
- India Opensource Data Flow
- What is Spark
- Why Spark
- File Formats
- Spark Architecture
- Important Concepts of Spark
- Let's Code…

# DLL Hadoop Architecture



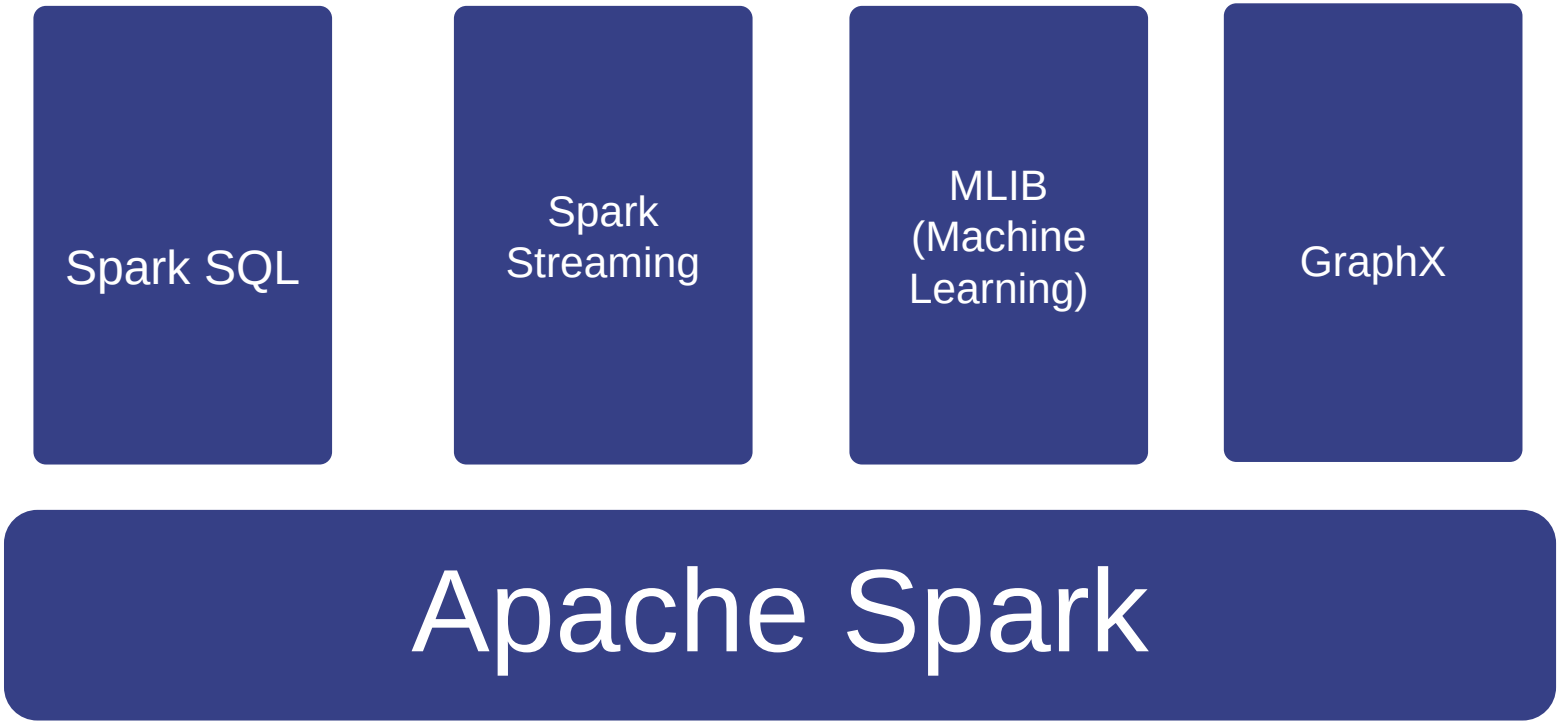
# INDIA OPENSOURCE DATA FLOW



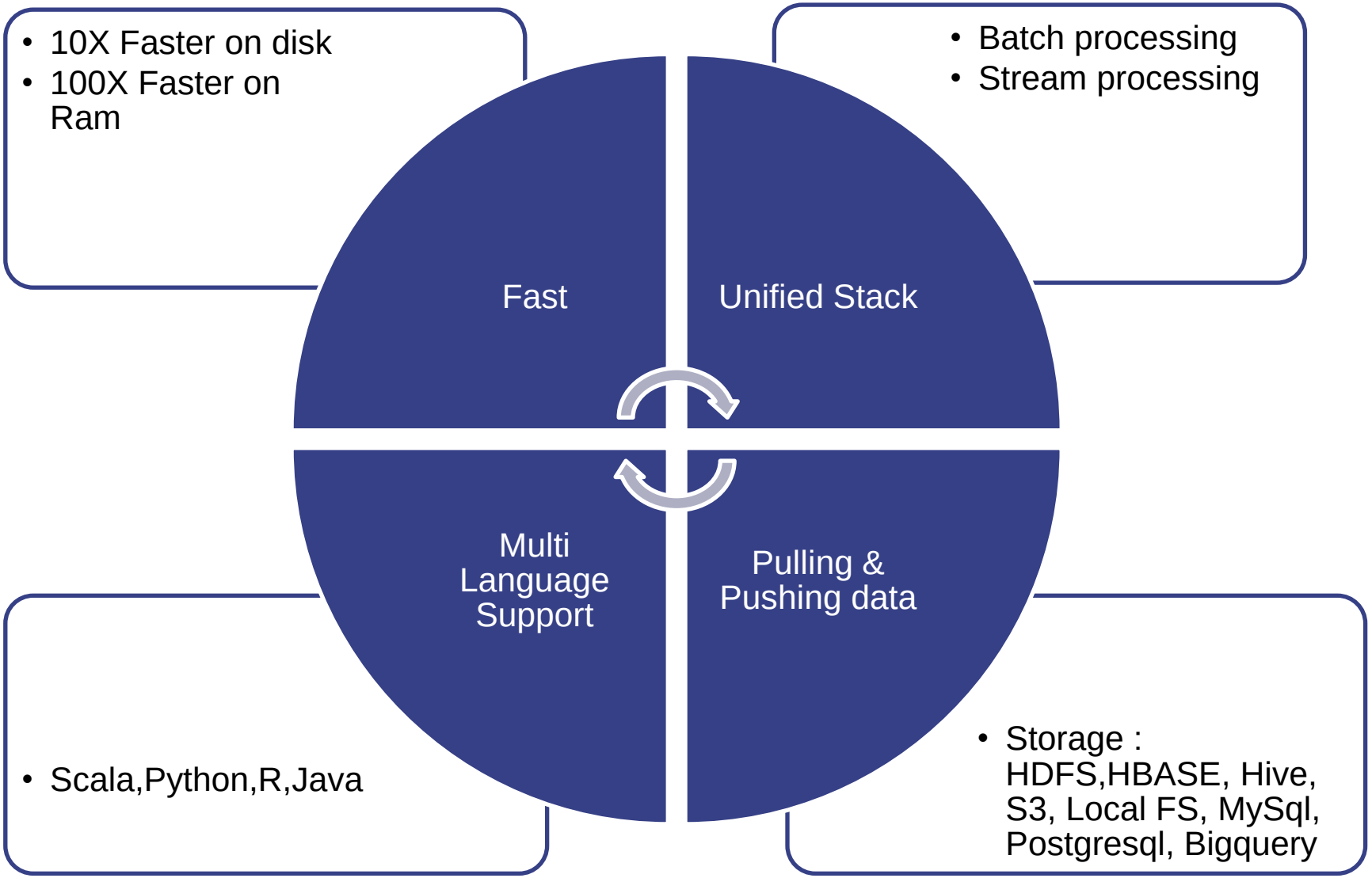
## What is Spark?


Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

## Components of spark



# Why Spark?

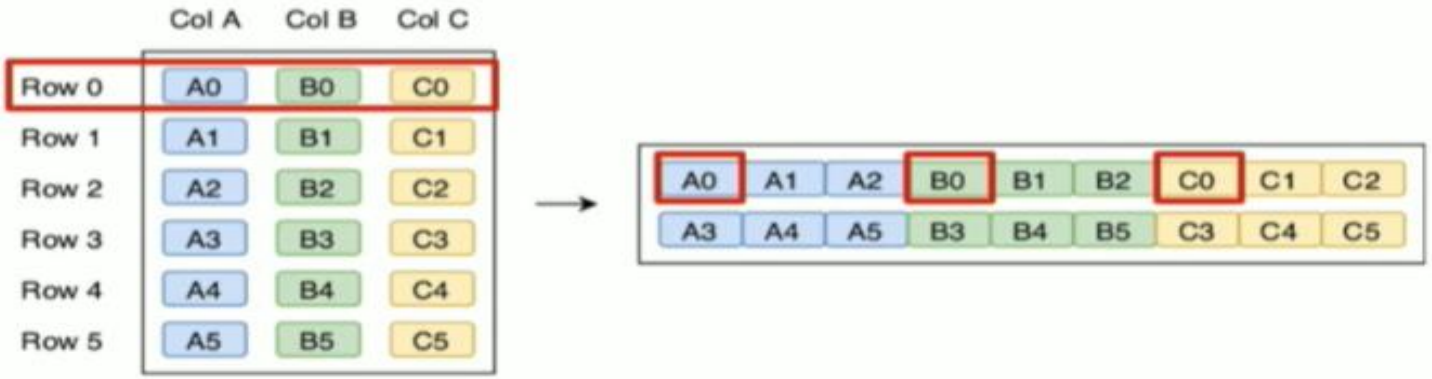

• 10X Faster on disk
• 100X Faster on Ram
• Batch processing
• Stream processing
Fast
Unified Stack
Multi Language Support
Pulling & Pushing data
• Scala,Python,R,Java
• Storage : HDFS,HBASE, Hive, S3, Local FS, MySql, Postgresql, Bigquery

# Types Of File For Spark & Hadoop

| | |
|---|---|
| All Supported Format | CSV, TEXT, RC,PARQUET,ORC,JSON |
| Parquet | • Parquet is an open-source file format available to any project in the Hadoop ecosystem. Apache Parquet is designed for efficient as well as performant flat columnar storage format of data compared to row-based files like CSV. |
| ORC | • The Optimized row columnar (ORC) file format provides a highly efficient way to store data in Hive. It was designed to overcome limitations of the other Hive file formats. |

# Types Of File For Spark & Hadoop

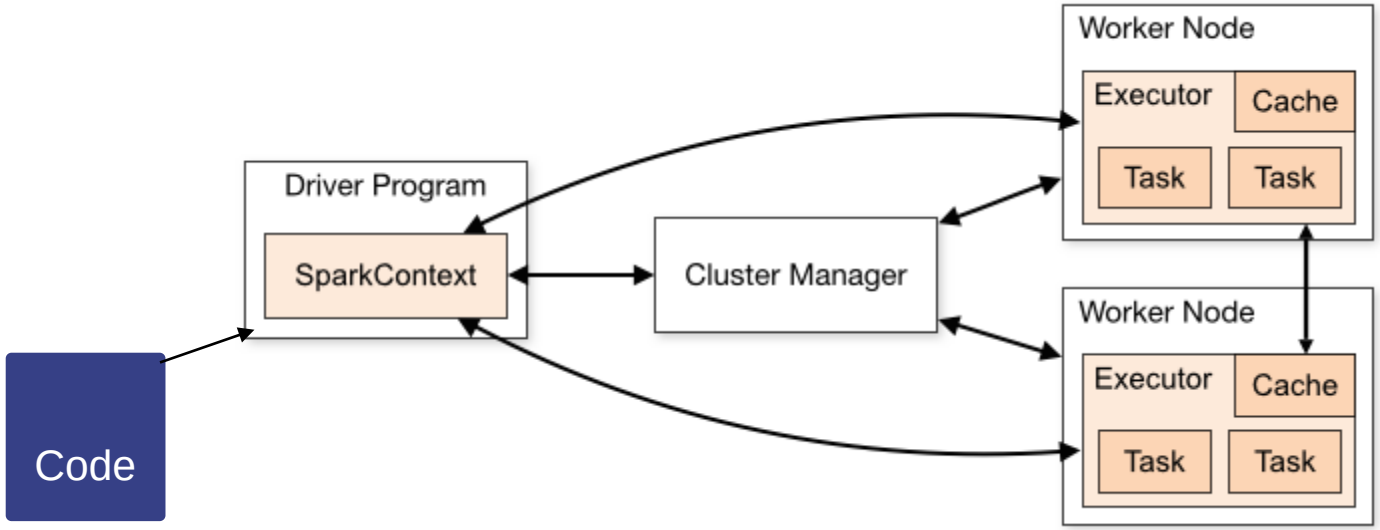
## Hybrid




- Horizontal & vertical partitioning
- Used by Parquet & ORC
- Best of both worlds

databricks

# Spark Architecture




# Working of nodes


Data Distributed Across Clusters

# Important to know

| | |
|---|---|
| RDD/DataFrame | • In Spark, a Data Frame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python. |
| Immutable | • Immutability is defined as unchangeable. When applied to an object, it means that its state can't be modified after it's created. |

## Why Immutable?

- Immutability rules out a big set of potential problems due to updates from multiple threads at once.
- Immutable data is definitely safe to share across processes.
- Immutable data can as easily live-in memory and on disk. This makes it reasonable to easily move operations.

# Important to know

DAG

- DAG (directed acyclic graph) is a collection of all RDD/DataFrame and the transformations on them.
- A Dag is created when the user creates a RDD/DataFrame and apply transformation on it, this results .

```
%spark2.pyspark
ecom=get_csv('file:///shared/sasusers/vinayh/Ecommerce_data.csv','csv','false')
ecom=ecom.dropna()
ecom=ecom.withColumn("InvoiceDate",to_date(col("InvoiceDate"),"MM/dd/yyyy"))
cust_details=get_csv('file:///shared/sasusers/vinayh/cust_details.csv','csv','true')
df  = cust_details.join(ecom, on=[cust_details['CustomerID']==ecom['CustomerID'],cust_details['InvoiceNo']==ecom['InvoiceNo']], how='inner')
```

SPARK JOBS FINISHED

```
== Physical Plan ==
*(2) BroadcastHashJoin [CustomerID#3350, InvoiceNo#3351], [CustomerID#3312, InvoiceNo#3306], Inner, BuildLeft
:- BroadcastExchange HashedRelationBroadcastMode(List(input[0, int, true], input[1, string, true]))
:  +- *(1) Project [CustomerID#3350, InvoiceNo#3351, email#3352, cell#3353L, address#3354]
:     +- *(1) Filter (isnotnull(InvoiceNo#3351) && isnotnull(CustomerID#3350))
:        +- *(1) FileScan csv [CustomerID#3350,InvoiceNo#3351,email#3352,cell#3353L,address#3354] Batched: false, Format: CSV, Location: InMemoryFileIndex[file:/shared/sasusers/vinayh/cust_details.csv], PartitionFilters: [], PushedFilters: [IsNotNull(InvoiceNo), IsNotNull(CustomerID)], ReadSchema: struct<CustomerID:int,InvoiceNo:string,email:string,cell:bigint,address:string>
+- *(2) Project [InvoiceNo#3306, StockCode#3307, Description#3308, Quantity#3309, cast(cast(unix_timestamp(InvoiceDate#3310, MM/dd/yyyy, Some(Asia/Dubai)) as timestamp) as date) AS InvoiceDate#3331, UnitPrice#3311, CustomerID#3312, Country#3313]
   +- *(2) Filter ((AtLeastNNulls(n, InvoiceNo#3306,StockCode#3307,Description#3308,Quantity#3309,InvoiceDate#3310,UnitPrice#3311,CustomerID#3312,Country#3313) && isnotnull(CustomerID#3312)) && isnotnull(InvoiceNo#3306))
      +- *(2) FileScan csv [InvoiceNo#3306,StockCode#3307,Description#3308,Quantity#3309,InvoiceDate#3310,UnitPrice#3311,CustomerID#3312,Country#3313] Batched: false, Format: CSV, Location: InMemoryFileIndex[file:/shared/sasusers/vinayh/Ecommerce_data.csv], PartitionFilters: [], PushedFilters: [IsNotNull(CustomerID), IsNotNull(InvoiceNo)], ReadSchema: struct<InvoiceNo:string,StockCode:string,Description:string,Quantity:int,InvoiceDate:string,UnitP...
```

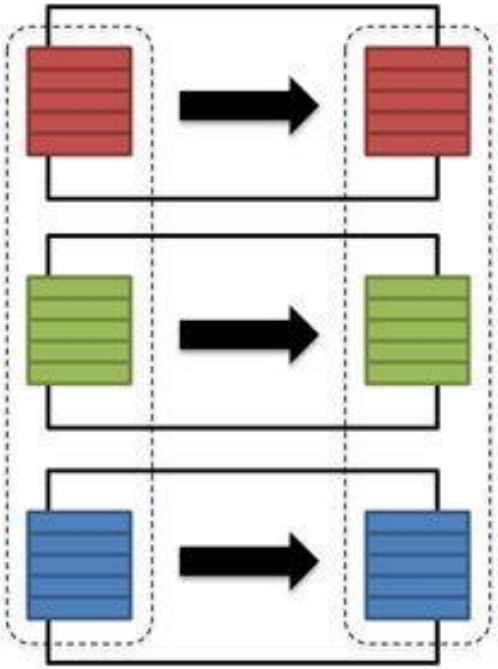
# Important to know

**Transformations**

- Spark transformations is a function that produces new RDD/Dataframe from the existing RDD's/ Dataframe's
- Transformation are lazy in nature when we call some operation on dataframe, it does not execute immediately.

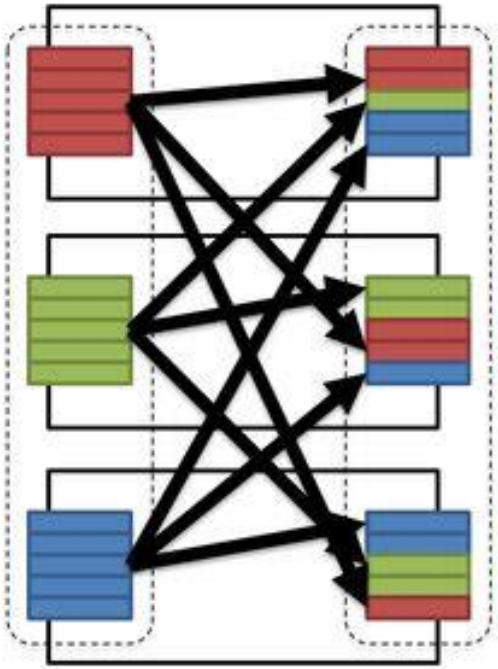
**Narrow transformation**

- Input and output stays in same partition
- No data movement is needed



**Wide transformation**

- Input from other partitions are required
- Data shuffling is needed before processing



# Important to know

| | |
|---|---|
| Actions | • Action is an operation which kicks off a job to execute on the cluster<br>Action example : show(),count(),write.csv() etc. |
| Shuffling | • Shuffling refers to moving data around to various worker nodes to complete the task |
| Broadcasting | • It providers copy of an object to each worker node. When each node has the copy of the data there is less communication between nodes.<br>• Using broadcasting can drastically speed up the join operations specially when dataframe is small compared to other. |
| Cache | • Cache is a optimization technique in dataframe for iterative and interactive spark application to improve the performance of the job. |

**THANK YOU, Happy Learning…**