HADOOP MAPREDUCE VS SPARK

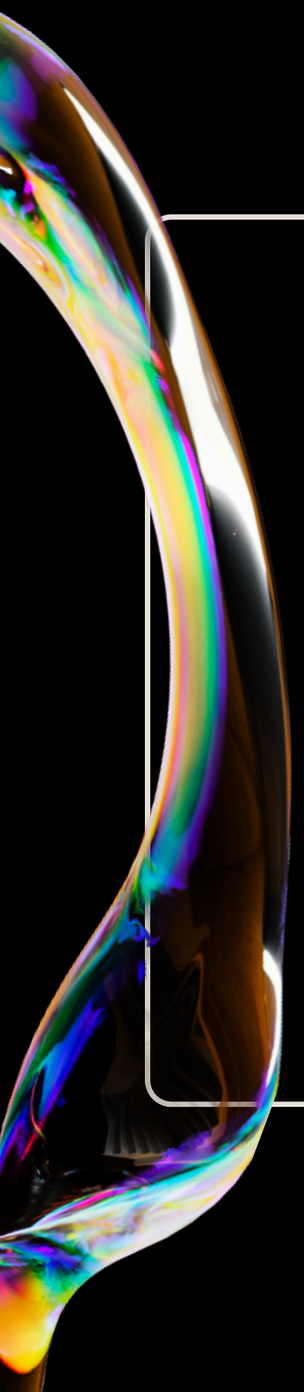
A Comparative Analysis of Big Data Processing Frameworks

Presented by

Rahulji V

Pragadish S

Saravnakrishnan B



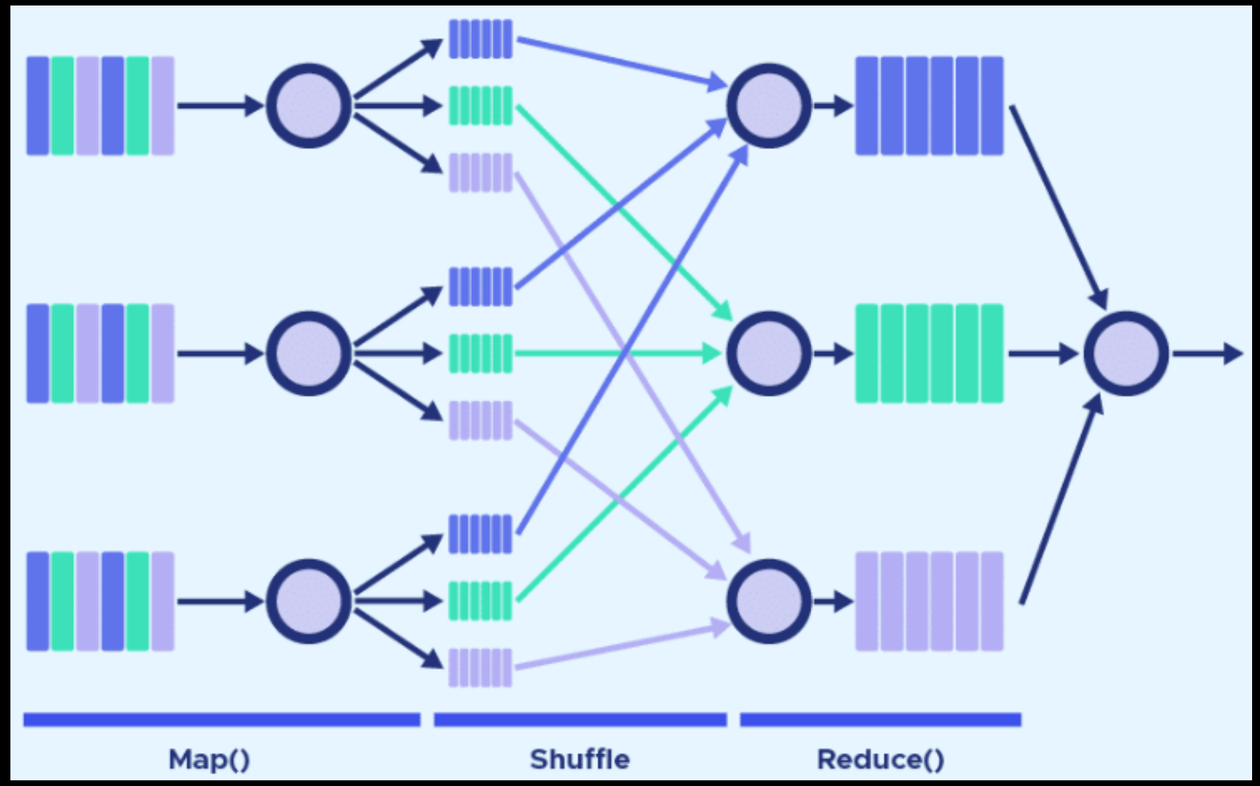
INTRODUCTION

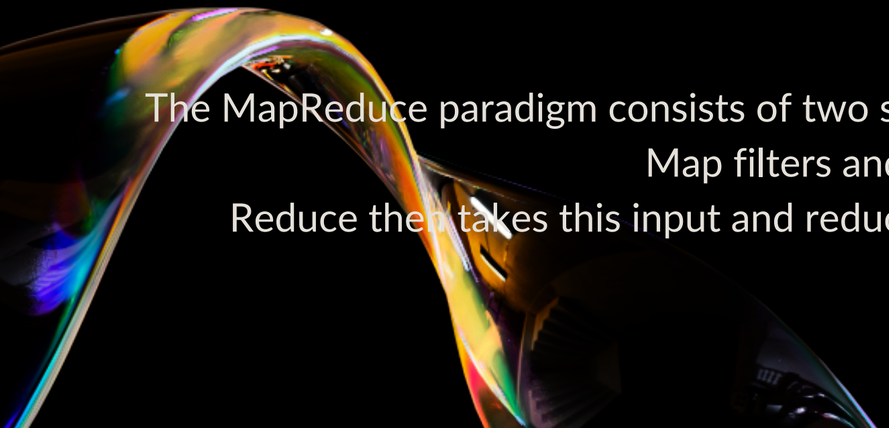
- What is Big Data?
  - Massive volumes of structured and unstructured data
  - Generated from various sources: social media, sensors, transactions
  - Requires specialized tools for processing and storage

- Why do we need distributed computing?
  - Traditional systems struggle with scalability and speed
  - Distributed computing enables parallel data processing
  - Ensures fault tolerance and efficiency

HADOOP MAPREDUCE

Hadoop MapReduce is described as "a software framework for easily writing applications which process vast amounts of data (multi-terabyte data sets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner."





The MapReduce paradigm consists of two sequential tasks: Map and Reduce (hence the name). Here's how each task works:

Map filters and sorts data while converting it into key-value pairs.

Reduce then takes this input and reduces its size by performing some kind of summary operation over the data set

MAPREDUCE

Key Components

- HDFS (Hadoop Distributed File System): Stores data across multiple machines.
- YARN (Yet Another Resource Negotiator): Manages cluster resources.
- MapReduce Processing Model:

1. Map Phase: Processes and filters input data.
2. Shuffle Phase: Groups and sorts data.
3. Reduce Phase: Aggregates results to produce the final output.

Advantages

- Highly fault-tolerant (data replication in HDFS).
- Can process petabytes of data.
- Runs on commodity hardware (cost-effective).

Disadvantages

- Slower due to disk I/O dependence.
- Complex programming model (Java-based).
- Inefficient for iterative and real-time processing.

HADOOP MAPREDUCE USAGE

Creating a MapReduce Job

- MapReduce programs are typically written in Java, Python, or other JVM languages.
- The job is submitted to YARN, which manages resource allocation.

```
hadoop jar myjob.jar MyMapReduceClass input output
```

Map Function (Mapper)

Processes input data in parallel and outputs key-value pairs.

```
def map_function(_, line):
    for word in line.split():
        yield (word, 1)
```

HADOOP MAPREDUCE USAGE

Reduce Function (Reducer)

Aggregates values for the same key to produce the final result.

```
def reduce_function(word, counts):
    yield (word, sum(counts))
```

Execution Workflow

- Step 1: Read data from HDFS.
- Step 2: Map phase: Distributes input data and applies transformation.
- Step 3: Shuffle & Sort: Groups similar keys together.
- Step 4: Reduce phase: Aggregates results and writes back to HDFS.

APACHE SPARK OVERVIEW

What is Apache Spark?

- An open-source, distributed computing system for big data.
- Designed for speed and ease of use.

Key Features of Spark

- In-Memory Processing: Reduces disk I/O, making computations faster.
- Batch & Real-Time Analytics: Supports both types of workloads efficiently.
- Rich Ecosystem: Components include Spark Core, Spark SQL, Streaming, MLlib, GraphX.
- RDDs (Resilient Distributed Datasets): Enable fault-tolerant parallel processing.

SPARK SQL USAGE

Creating a SparkSession

- A SparkSession is the entry point for working with Spark SQL and DataFrames.
- It replaces the older SparkContext, SQLContext, and HiveContext, unifying them into a single interface.
- It allows you to create DataFrames, execute SQL queries, and interact with Spark components.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Example").getOrCreate()
```

Querying Data

Spark SQL allows querying structured data using SQL syntax.

```
df = spark.sql("SELECT * FROM table")
df.show()
```

SPARK SQL USAGE

Transformations

Common transformations include filtering, grouping, and selecting columns.

```
df_filtered = df.filter(df["age"] > 30)
df_grouped = df.groupby("department").count()
df_selected = df.select("name", "salary")
```

User-Defined Functions (UDFs)

Custom functions for complex operations.

```
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

def square(x):
    return x * x

square_udf = udf(square, IntegerType())
df.withColumn("squared_value", square_udf(df["column_name"]))
```

ADVANTAGES

SPEED

Speed In-memory processing makes Spark 100x faster than MapReduce for iterative workloads.

EASE OF USE

Supports Python, Scala, Java, and SQL, making it user-friendly for data engineers and analysts.

DISADVANTAGES

HIGH MEMORY CONSUMPTION

Requires significant RAM for working thus, making it expensive for large-scale deployments.

COMPLEX DEPLOYMENT

Setting up and tuning Spark for optimal performance can be challenging compared to Hadoop

HADOOP MAPREDUCE

- Hadoop's MapReduce model reads and writes from a disk, thus slowing down the processing speed.
- Hadoop is designed to handle batch processing efficiently.
- Hadoop is a high latency computing framework, which does not have an interactive mode.
- With Hadoop MapReduce, a developer can only process data in batch mode only.
- Hadoop is a cheaper option available while comparing it in terms of cost

APACHE SPARK

- Spark reduces the number of read/write cycles to disk and stores intermediate data in memory, hence faster-processing speed.
- Spark is designed to handle real-time data efficiently.
- Spark is a low latency computing and can process data interactively.
- Spark can process real-time data, from real-time events like Twitter, and Facebook.
- Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost.

HADOOP MAPREDUCE

- Hadoop is a highly fault-tolerant system where Fault-tolerance achieved by replicating blocks of data. If a node goes down, the data can be found on another node
- Hadoop supports LDAP, ACLs, SLAs, etc and hence it is extremely secure.
- Data fragments in Hadoop can be too large and can create bottlenecks. Thus, it is slower than Spark.
- Hadoop is easily scalable by adding nodes and disk for storage. It supports tens of thousands of nodes.

APACHE SPARK

- Fault-tolerance achieved by storing chain of transformations.If data is lost, the chain of transformations can be recomputed on the original data
- Spark is not secure, it relies on the integration with Hadoop to achieve the necessary security level.
- Spark is much faster as it uses MLib for computations and has in-memory processing.
- It is quite difficult to scale as it relies on RAM for computations. It supports thousands of nodes in a cluster.

THANK YOU

for your time and attention

A Presentation by
Rahulji V
Pragadish S
Saravnakrishnan B