PySpark

Introduction to Big Data and Apache Spark

Agenda

01 What is Big Data?

02 The 5 Vs of Big Data

03 Problems with Big Data

04 Introduction to Hadoop

05 What is Hadoop?

06 MapReduce and HDFS

07 Introduction to Spark

08 Apache Spark Architecture

09 Spark vs Hadoop

10 Applications Using Apache Spark

Introduction to **Big Data**

What is Big Data?

What is Big Data?

Definition: Big Data are extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions

Big Data Analytics

- The process of examining large and varied data sets to uncover information including hidden patterns, market trends, and customer preferences
- This will help organizations make informed business decisions

Big Data and Facebook

- **Huge Amount of Data:** We're aware of the expenses in storing and handling huge amounts of data

UNIQUE

- **Heterogeneous Data:** They are unstructured, semi-structured, and structured data. Lots of variety from lots of sources

- **Accessing and processing speed:** If you have a 100 Mbps I/O channel and you need to process 2TBs of data – it will take you nearly 6 hours to process the data

Big Data and Facebook

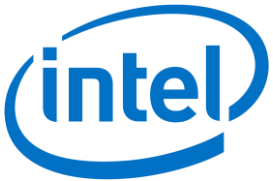
- There are ***2.3 billion*** monthly active users and counting
- There are ***250 billion photos*** uploaded to Facebook
- This equates to ***350 million*** photos per day!
- Facebook generated ***4 petabytes*** of data per day!

1 petabyte = 1,000,000 Gb

Imagine handling all this **data**!

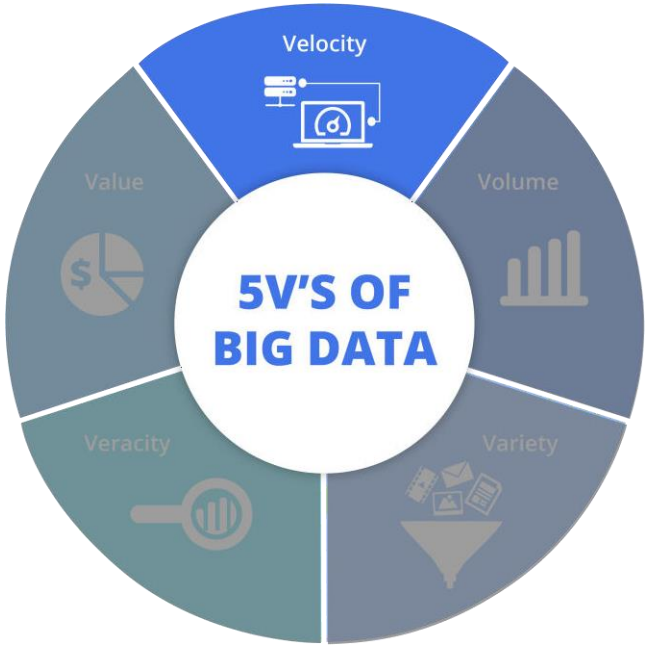
Big Data and Rest of the World

Almost every firm which has access to a huge repository had their fair share of problems with data!



5 V’s of **Big Data**

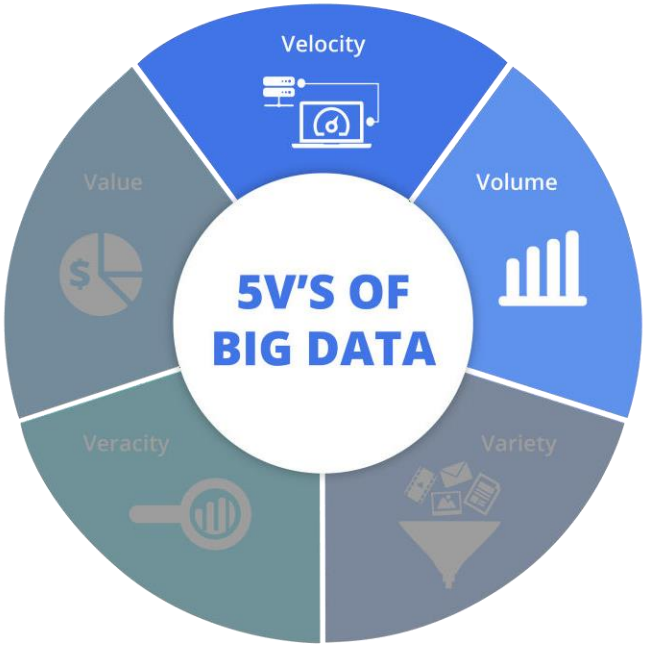
The 5 V's of Big Data



Velocity

- Velocity refers to the high speed of accumulation of data
- Big Data velocity indicates speed at which data flows in from sources such as machines, networks, social media, mobile phones, etc.
- There is a massive and continuous flow of data
- **Example:** More than 3.5 billion searches per day are made on Google

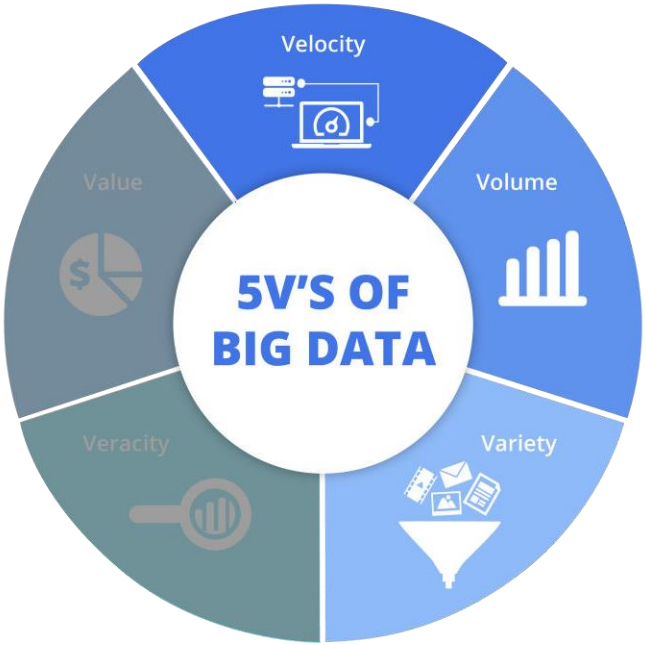
The 5 V's of Big Data



Volume

- Volume means huge amount of data!
- To determine the value of data, size of data plays a very crucial role
- When dealing with Big Data, it is necessary to consider a characteristic 'Volume'
- **Example:** In the year 2016, the estimated global mobile traffic was **6.2 Exabytes (6.2 billion GB) per month**

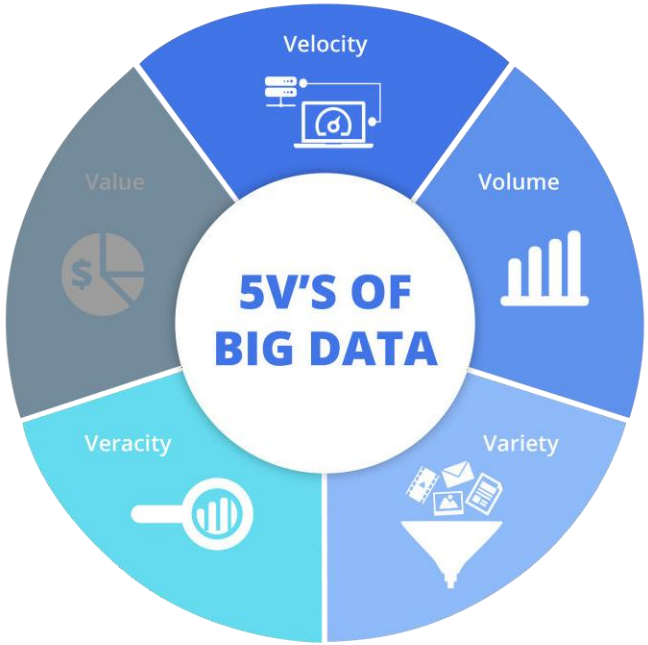
The 5 V's of Big Data



Variety

- It refers to the nature of data that is structured, semi-structured, and unstructured
- It also refers to heterogeneous sources
- Variety is basically the arrival of data from new sources that are both inside and outside of an enterprise

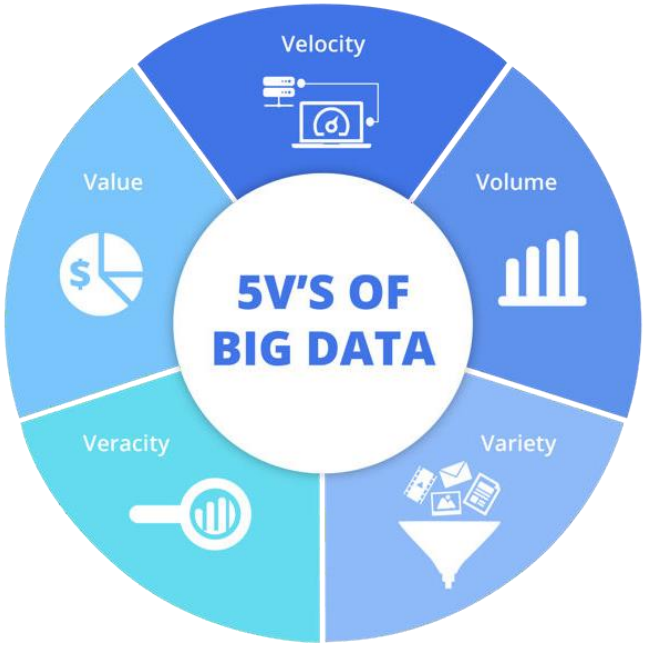
The 5 V's of Big Data



Veracity

- It refers to inconsistencies and uncertainties in data
- Data which is available can sometimes get messy; hence, quality and accuracy of the data are difficult to control
- **Example:** Data in bulk could create confusion, whereas less amount of data could convey half or incomplete information

The 5 V's of Big Data



Value

- The bulk of Data having no value is of no good to the company, unless you turn it into something useful
- Data (just by itself) is of no use or importance, but it needs to be converted into something valuable to extract information
- Hence, you can state that 'Value' is the most important 'V' of all the 5Vs

What is **Hadoop**?

What is Hadoop?

- Hadoop is a **framework** which is used to store Big Data across a spectrum of devices
- This is done to help you process **Big Data** in **parallel**

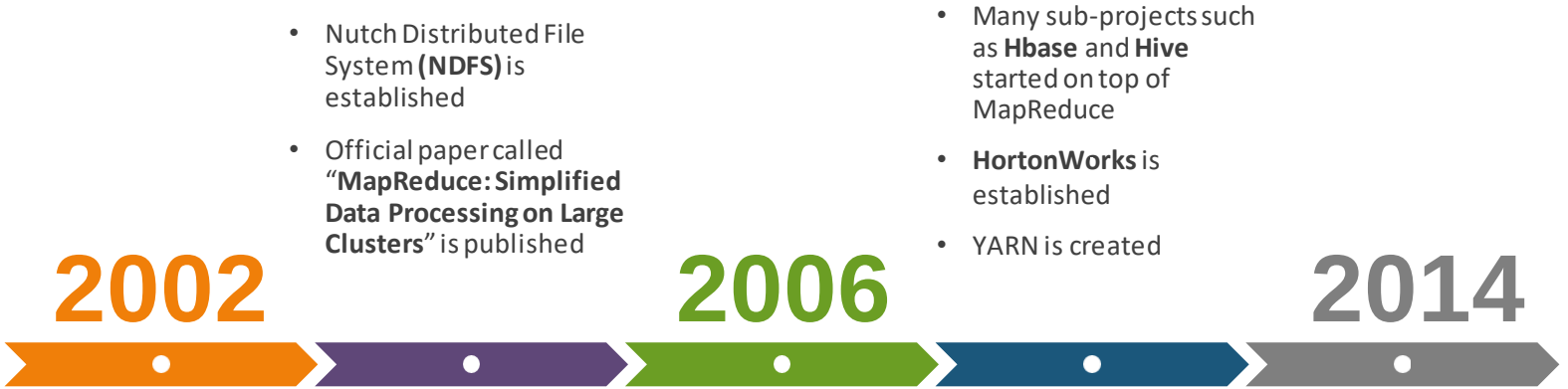
Three important components in the Hadoop Ecosystem that you should know:

1. HDFS

2. MapReduce

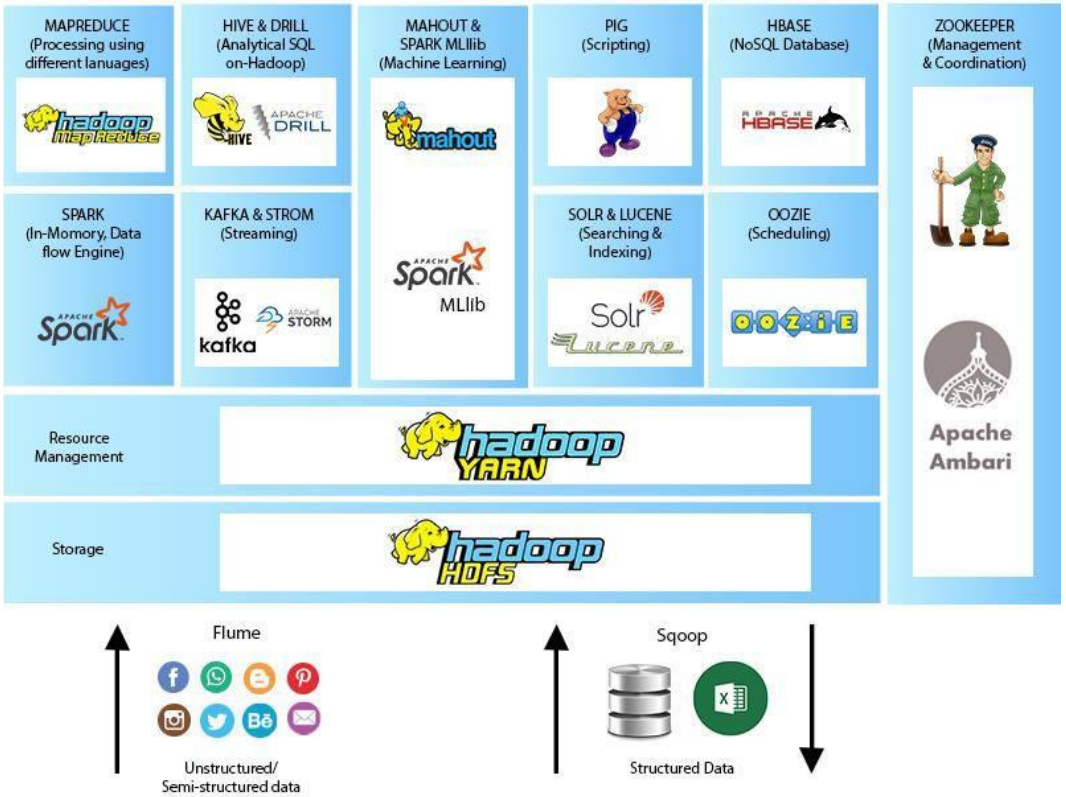
3. YARN

History of Hadoop



Hadoop was named after the toy elephant that the founder's son had!

Hadoop Ecosystem



Components of **Hadoop**

Hadoop Components: HDFS

- The Hadoop Distributed File System (HDFS) is a storage system which runs on **Java** programming language and used as a primary storage device in Hadoop applications
- HDFS consists of two components, which are **Namenode** and **Datanode**
- These applications are used to store large data across **multiple nodes** on the Hadoop cluster

Hadoop Components: HDFS

NameNode:

- NameNode is a daemon which **maintains** and **operates** all DATA nodes (slave nodes)
- It acts as the **recorder** of metadata for all blocks in it, and it contains information like size, location, source, hierarchy, etc.
- It records all **changes** that happen to metadata
- If any file gets **deleted** in the HDFS, the NameNode will automatically record it in **EditLog**
- NameNode frequently receives **heartbeat** and **block report** from the data nodes in the cluster to ensure they are working and live

Hadoop Components: HDFS

DataNode:

- It acts as a **slave node** daemon which runs on each slave machine
- The data nodes act as a **storage** device
- It takes responsibility to serve **read** and **write request** from the user
- It takes the responsibility to act according to the instructions of **NameNode**, which includes deleting blocks, adding blocks, and replacing blocks
- It sends **heartbeat reports** to the NameNode regularly, and the actual time is once in every 3 seconds

Hadoop Components: MapReduce

- **Map function:** It converts one set of data into another, where individual elements are broken down into tuples (key/value pairs)
- **Reduce function:** It takes data from the Map function as an input. Reduce function aggregates and summarizes the results produced by Map function

Hadoop Components: MapReduce

- **MapReduce** acts as a **core component** in Hadoop Ecosystem, as it facilitates the logic of processing
- It is a **software framework** which enables us in writing applications that process large data sets using **distributed** and **parallel** algorithms in a Hadoop environment
- Parallel processing feature of MapReduce plays a **crucial role** in Hadoop ecosystem. It helps in performing Big Data analysis using **multiple machines** in the same cluster

Hadoop Components: YARN

- Supports a **variety** of processing engines and applications
- Separates its duties across **multiple components**
- **Dynamically** allocates pools of resources to other applications on the go

- YARN supports **multiple scheduling methods** based on the queue format for submitting processing jobs
- The default scheduler works on **FIFO**

Key characteristics of **Hadoop**

Why do we need Hadoop?

Flexibility

- Major challenges for any company or organization would be on how to handle the structured and unstructured data that is available with the Big Data
- Hadoop is very flexible, and it would be able to easily handle structured and unstructured data
- Hadoop can also handle encoded data and process them according to the company needs

Why do we need Hadoop?

Scalability

- If the maximum size of the storage nodes is reached, then it is easy to add additional nodes to the cluster
- This makes the Hadoop framework easily scalable across the spectrum
- The nodes are independent of each other, so adding a new node to the cluster will not be a trivial task

Why do we need Hadoop?

Spare Tire

Fault Tolerance

- In Hadoop, the data is stored in HDFS where data gets replicated at multiple locations
- Even if one or two of the systems collapse, the file is still available on the backup systems so that it would be easily retrieved
- Hadoop is bulletproof in terms of fault tolerance

Why do we need Hadoop?

Rapid Data Processing and Low Costs

- Hadoop generally works on the concept of parallel processing
- Hadoop can perform batch processes 10 times faster than on a single thread server or on the mainframe
- Hadoop is a comparatively cheap and cost-effective way of handling the Big Data when compared with the other frameworks

When to use **Hadoop**?

When to use Hadoop?

YAHOO!

amazon

Searching

When searching for large amount of data

Aol.

twitter

Data Warehouse

When handling different type of data

facebook

Google

Log Processing

When processing lot of logs for data

ZEE5

The New York Times

Video Processing

When processing heterogenous graphics

When **not**
to use **Hadoop**?

When not to use Hadoop?

Low Latency Data Access

When you need **quick access** to your data!

Multiple Data Modification

Hadoop **cannot** handle data modification efficiently!

Non-efficient with small files

Hadoop works well only with large files but not with small ones!

Introduction to **Apache Spark**

Introduction to Apache Spark

Spark was introduced by **Apache Software Foundation** for speeding up the computational procedures of the software processes.

Spark uses Hadoop in two ways – one is **storage,** and another is **processing**. Since Spark has its own **cluster management** computation, it uses Hadoop for **storage purposes** only

What is Apache Spark?

- Apache Spark is a **lightning-fast cluster computing** technology, designed for fast computation
- It is based on Hadoop **MapReduce,** and it extends the **MapReduce** model to **efficiently** use it for more types of **computations**, which includes interactive queries and **stream processing**
- Spark is one of Hadoop's sub-project developed in **2009** in UC Berkeley's **AMPLab** by Matei Zaharia

History of Spark

- Apache Spark started as a research project at the **UC Berkeley AMPLab** in 2009
- It was open sourced in early **2010**
- Many of the ideas behind the system were presented in various **research papers** over the years
- After being released, **Spark** grew into a broad developer **community** and moved to the Apache Software Foundation in **2013**
- Today, the project is developed **collaboratively** by a community of hundreds of developers from **hundreds** of **organizations**

Features of Apache Spark

Speed: Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk

- This is possible by reducing the number of read/write operations to disk. It stores the intermediate processing of data in memory

Supports multiple languages: Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages

- Spark has 80 high-level operators for interactive querying

Advanced Analytics: Spark not only supports 'Map' and 'reduce' but also supports SQL queries, Streaming data, Machine Learning (ML), and Graph algorithms

Features of Apache Spark

In-Memory Computation: No need to fetch data from the disk every single time

- Saves a LOT of time

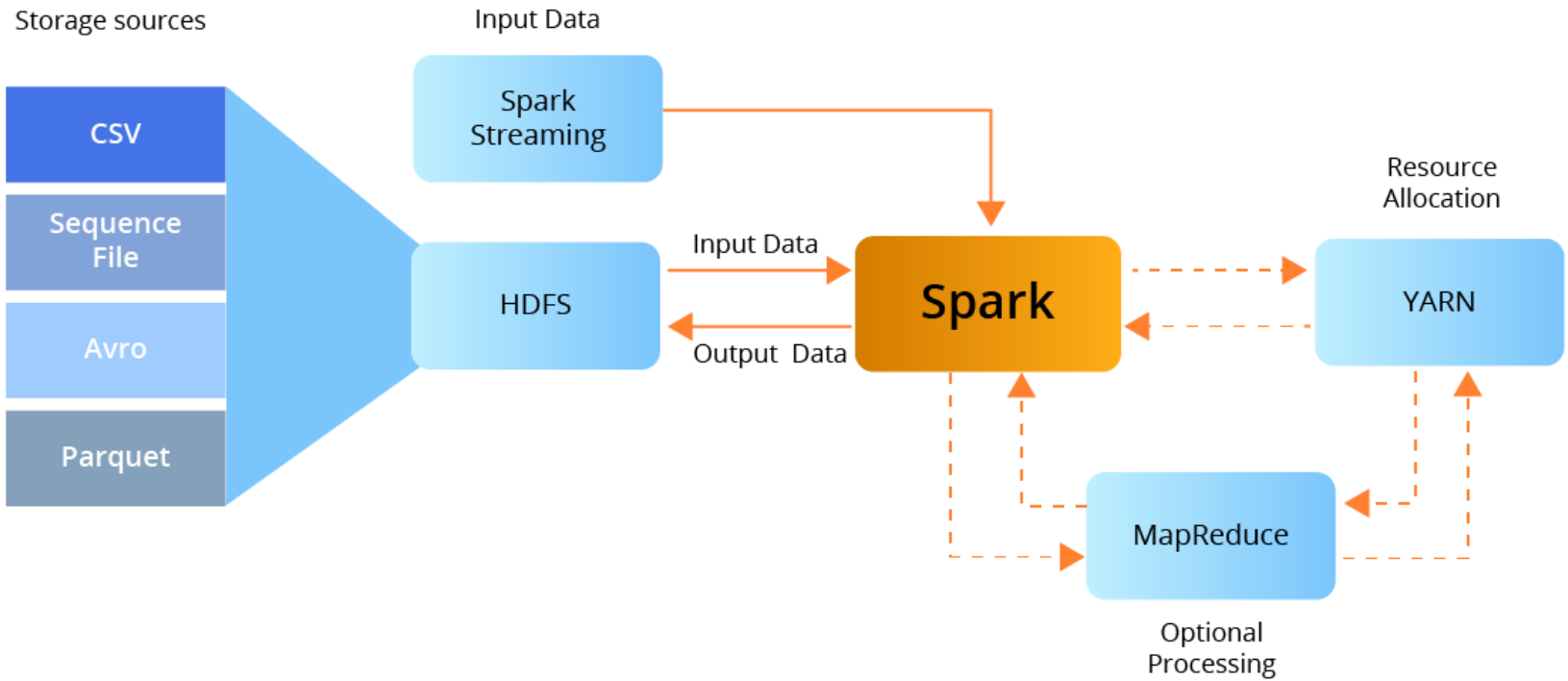
Fault Tolerance: Fault tolerance is provided through Spark abstraction-RDD

- This implies very less data loss or nil

Lazy Evaluation: All transformations made in Spark RDD involves creation of a new RDD

- This enhances efficiency of the system

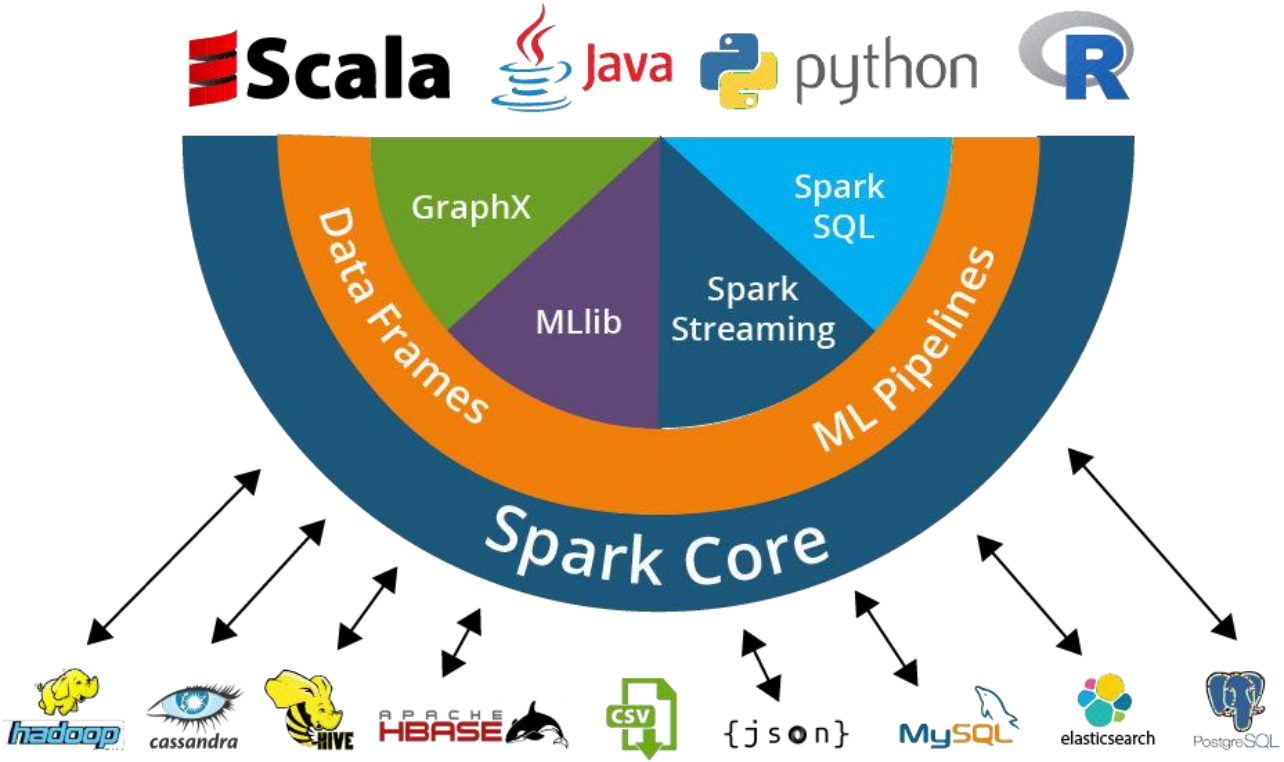
An Overview of Apache Spark



Who uses Spark?

Apache Spark
Ecosystem

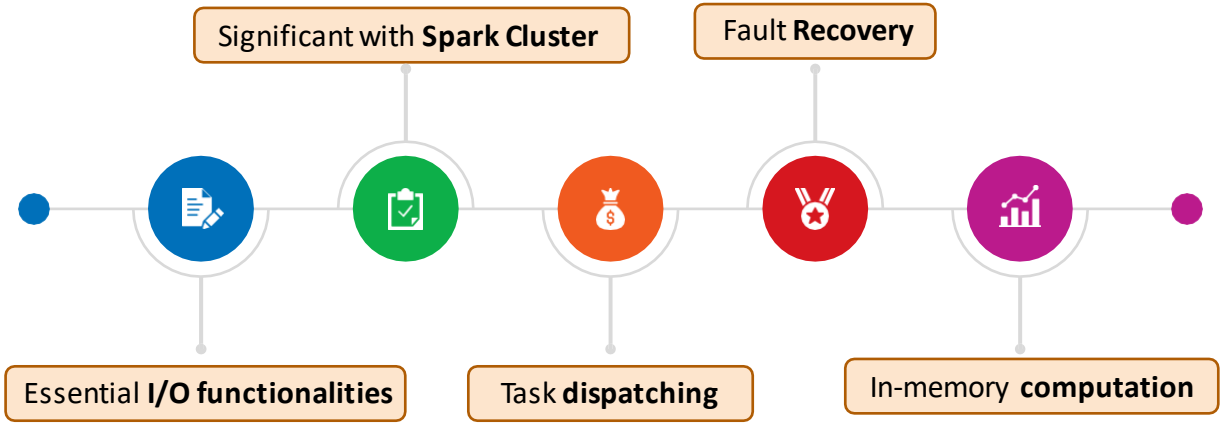
Apache Spark Ecosystem



Apache Spark Ecosystem

Apache Spark Core

- Spark Core is the underlying general execution engine for Spark platform that all other functionality is built upon
- It provides In-Memory computing and referencing datasets in external storage systems



Apache Spark Ecosystem

Spark SQL

- The Spark SQL component is a **distributed framework** for structured data processing
- Using Spark SQL, Spark gets **more information** about the structure of data and the computation
- With this information, Spark can perform **extra optimization**. It uses same **execution engine** while computing an output. It does not depend on API/language to express the computation

Apache Spark Ecosystem

Spark Streaming

- It is an **add-on** to **core Spark API** which allows scalable, high-throughput, fault-tolerant stream processing of live data streams
- Spark can **access data** from sources such as Kafka, Flume, Kinesis, or TCP socket
- Finally, the data received is given to the **file system**, databases, and live dashboards. Spark uses **Micro-batching** for real-time streaming

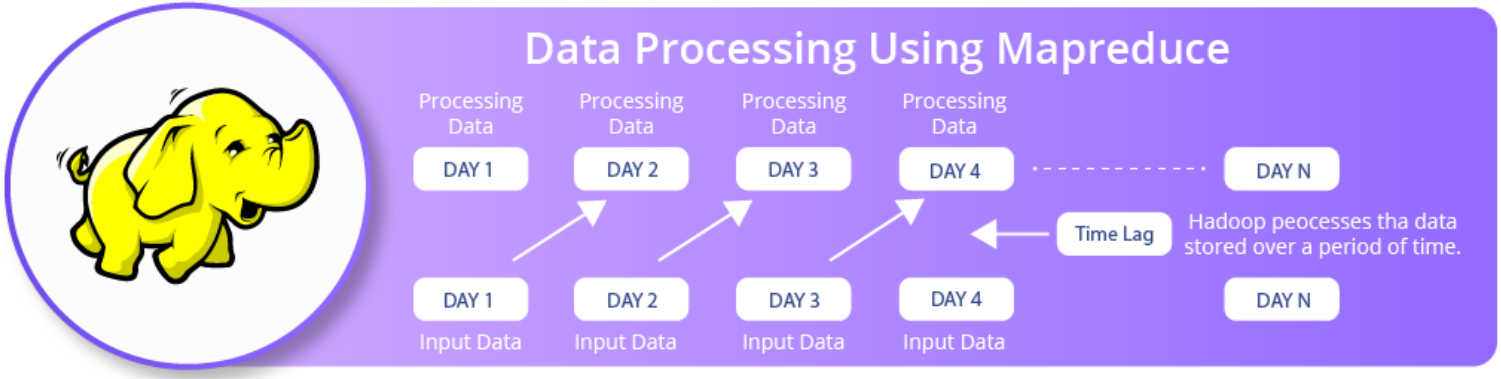
Apache Spark Ecosystem

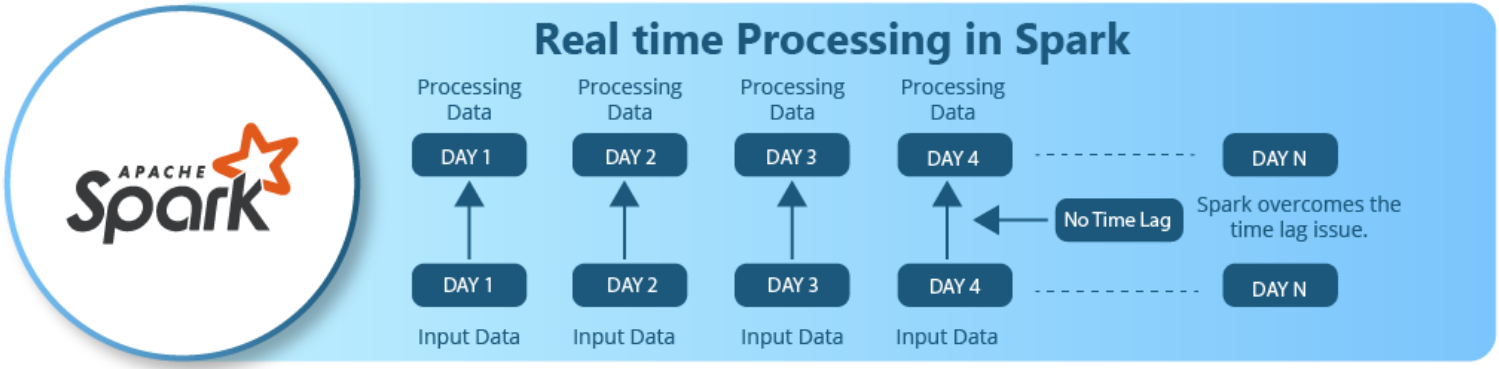
MLlib (Machine Learning Library)

- MLlib in Spark is a scalable Machine Learning library that discusses both high-quality algorithm and high speed
- The motive behind MLlib creation is to make Machine Learning scalable and easy. It contains Machine Learning libraries that have an implementation of various Machine Learning algorithms
- For example: clustering, regression, classification, and collaborative filtering

Hadoop vs **Spark**

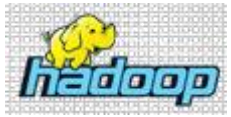
Spark vs Hadoop





Hadoop vs Spark

#1. Category

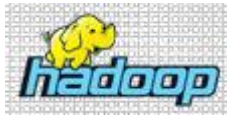


Basic Data Processing Engine

Data Analytics Engine

Hadoop vs Spark

#2. Usage

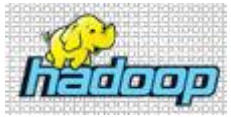


Batch processing with huge volume of data

Processes real-time data from events like Twitter and Facebook handles

Hadoop vs Spark

#3. Latency

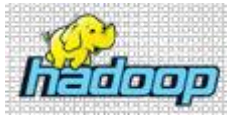


High Latency Computing

Low Latency Computing

Hadoop vs Spark

#4. Data

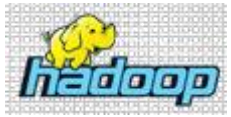


Data processing done only in Batch Mode

Data can be processed interactively as well

Hadoop vs Spark

#5. Ease of use

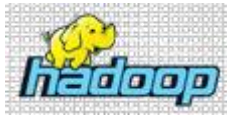


MapReduce is complex and requires handling of low-level APIs

Easier to use, data can be processed using high-level operators

Hadoop vs Spark

#6. Scheduler



No in-memory computation; external Job Scheduler is required

Support for in-memory computation; no need for external scheduler

Using **Apache Spark** With **Hadoop**

Apache Spark and Hadoop

Hadoop Adding Value to Spark

1. Hadoop includes HDFS, which allows you to distribute data across nodes within a cluster of servers; thus, it eliminates the need for lots of custom hardware
2. Hadoop provides enhanced data security, which is critical for production workloads

Spark Adding Value to Hadoop

1. Spark's Machine Learning algorithms can execute faster since they're executed in-memory, as opposed to MapReduce programs
2. Spark is used for data streaming and in-memory distributed processing for faster real-time analysis

Using Apache Spark with Hadoop

- Apache Spark fits into the **Hadoop open-source community**, building on top of the Hadoop Distributed File System (HDFS)
- Spark is **not tied** to the two-stage MapReduce paradigm and promises performance up to **100 times faster** than Hadoop MapReduce for certain applications

Using Apache Spark with Hadoop

- **Apache Spark Compatibility with Hadoop [HDFS, HBASE, and YARN]** – Apache Spark is fully compatible with Hadoop's Distributed File System (HDFS)
- Same goes with other Hadoop components such as **YARN** (Yet Another Resource Negotiator) and the **HBase** distributed database

Using Apache Spark with Hadoop

- **Runs 100 times faster** – Spark can speed up jobs that run on the Hadoop data-processing platform
- Apache Spark provides the **ability** to create **data-analysis jobs** that can run 100 times faster than those running on the standard Apache Hadoop MapReduce
- **MapReduce** is **slow** because it executes jobs in batch mode, which means that **real-time** analysis of data is not an option

Using Apache Spark with Hadoop

- **Alternative to MapReduce** – it executes jobs in short bursts of micro-batches that are five seconds or less apart
- It provides more **stability** than real-time, stream-oriented Hadoop frameworks such as Twitter Storm
- The software can be used for a **variety** of jobs, such as an **ongoing analysis** of live data, and with a software library, more computationally **in-depth jobs** involving Machine Learning and graph processing

Using Apache Spark with Hadoop

- **Support for Multiple Languages** – developers can write data-analysis programs and code in Java, Scala, or Python
- Access to more than **80 high-level operators** in Spark

Using Apache Spark with Hadoop

- **Library Support** – designed to add to the types of processing jobs being explored more aggressively with the latest commercially supported deployments of Hadoop
- MLlib implements a slew of common Machine Learning algorithms, such as **naïve Bayesian** classification or **clustering**
- Spark Streaming enables **high-speed processing** of data ingested from multiple sources
- GraphX allows for **computations** on **graph data**

Apache Spark & hadoop

Spark MLlib

Apache Spark Streaming

Spark GraphX

Using Apache Spark with Hadoop

- **Stable API** – with the version 1.0, Apache Spark offers a stable API (Application Programming Interface)
- Developers use it with Spark through their **own applications**
- This helps in using **Storm** more **easily** in Hadoop-based deployment

Using Apache Spark with Hadoop

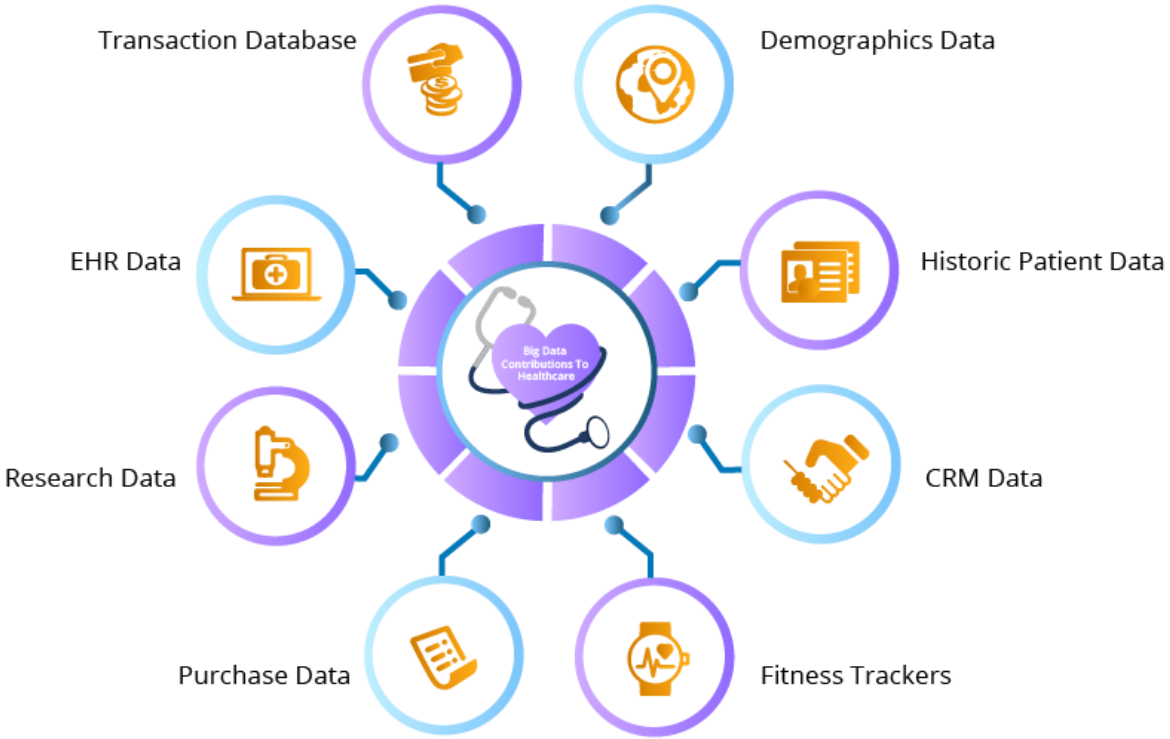
- **SPARK SQL Component** – used to access structured data; it allows the data to be interrogated alongside unstructured data in analysis work
- Spark SQL (**Alpha stage**) allows SQL-like queries to be run against **data stored** in Apache Hive
- Extracting data from Hadoop via SQL queries is an **advantage** when considering **real-time querying** functionality which goes around Hadoop

Using Apache Spark with Hadoop

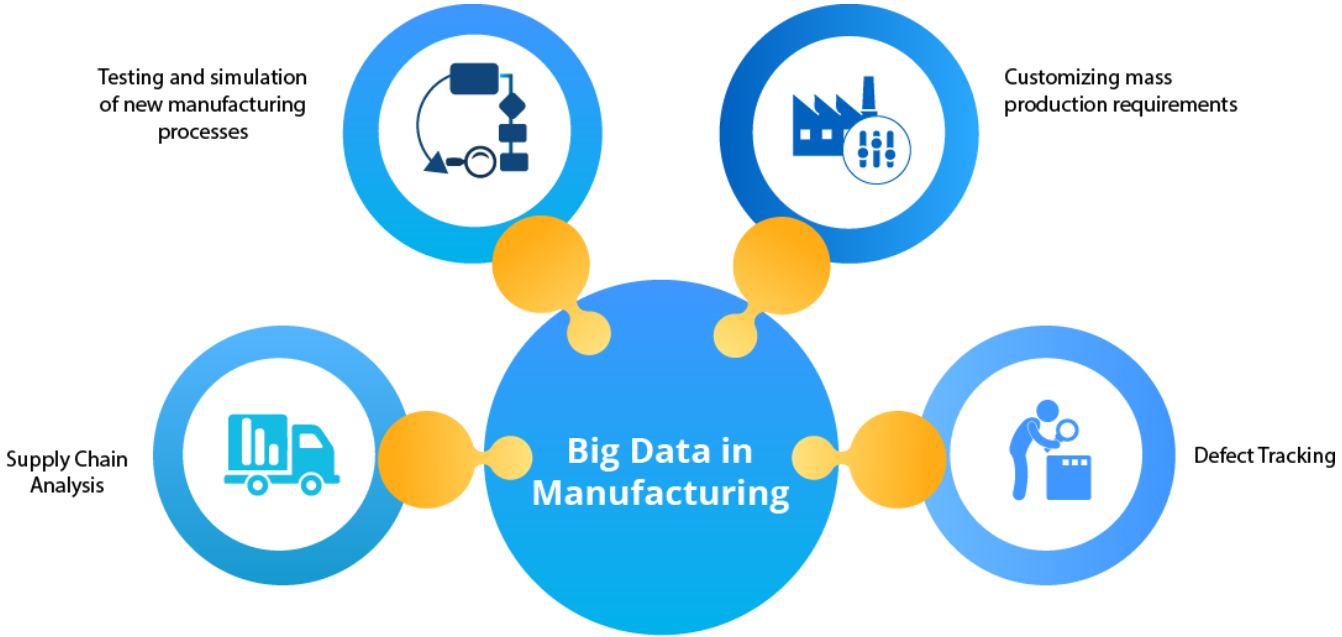
- **SPARK GraphX** – GraphX is a library for manipulating graphs
- It provides **analysis** and graph **computation** for Big Data
- GraphX comes with a **variety** of graph algorithms and graph computations
- GraphX extends the Spark RDD with a Resilient Distributed Property Graph

Applications using **Apache Spark**

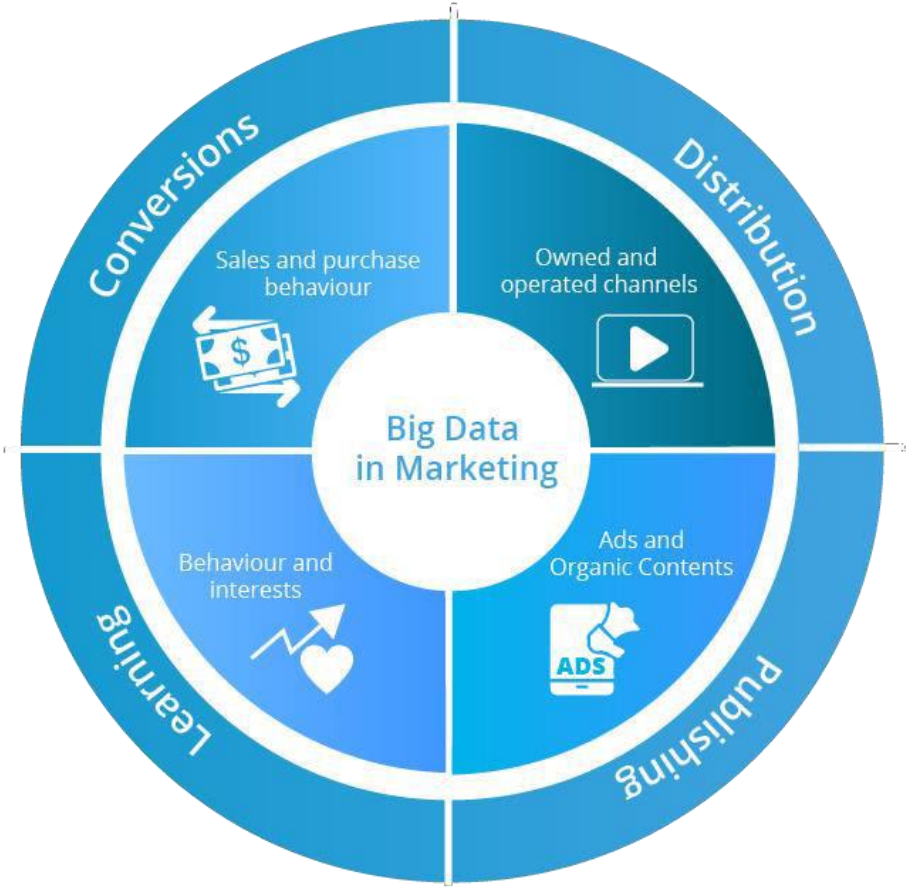
Apache Spark Application #1: Healthcare



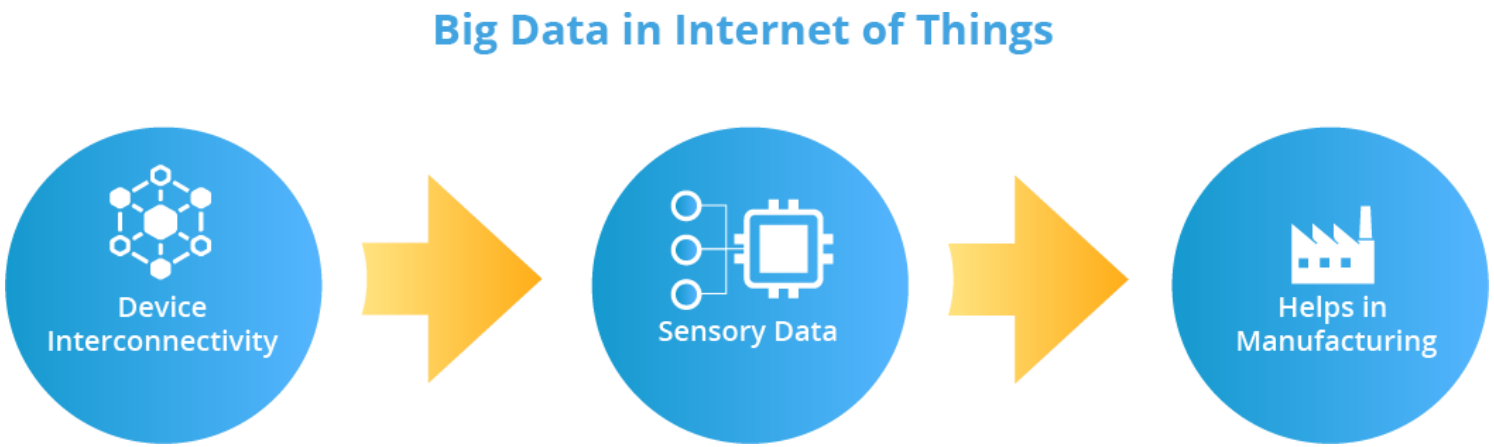
Apache Spark Application #2: Manufacturing



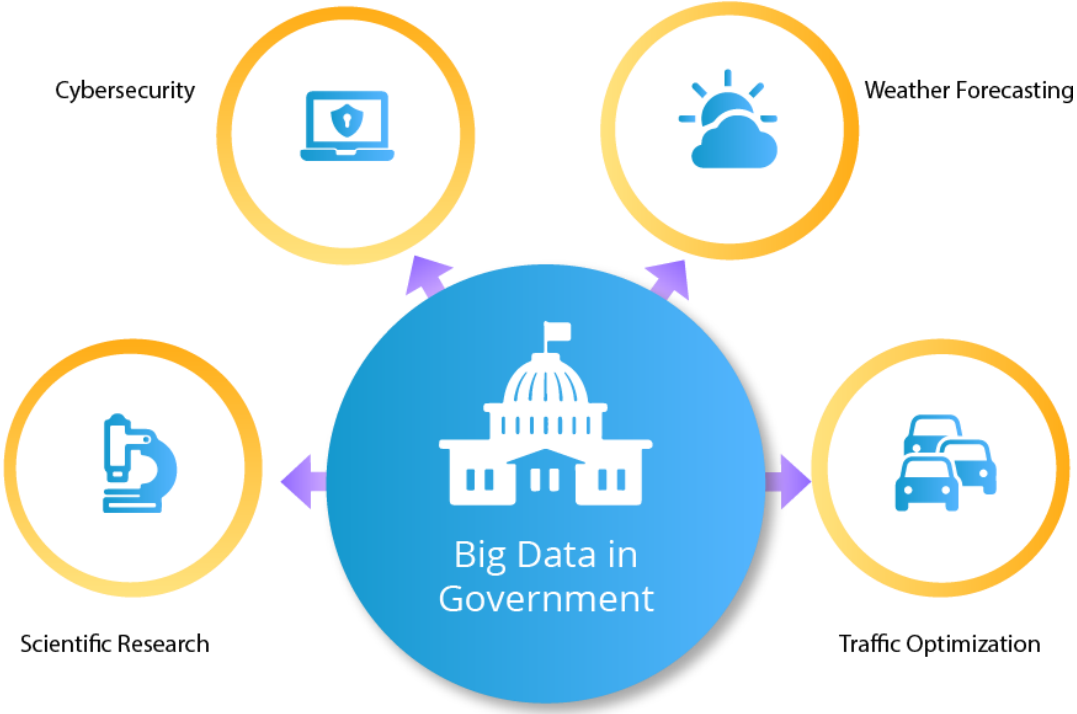
Apache Spark Application #3: Media



Apache Spark Application #4: Internet of Things



Apache Spark Application #5: Government



Quiz!

Question #1

Which of the following is a V from the Big Data chain?

A	Value
B	Veracity
C	Vacancy
D	Vertex

Answer #1

Which of the following is a V from the Big Data chain?

A	Value
B	**Veracity**
C	Vacancy
D	Vertex

Question #2

Which of the following is **NOT** a part of Hadoop?

A	Reduce
B	Map
C	Javascript
D	YARN

Answer #2

Which of the following is **NOT** a part of Hadoop?

A	Reduce
B	Map
C	**Javascript** ✓
D	YARN

Question #3

"Apache Spark's components are more efficient than MapReduce" – True or False?

A	True
B	False

Answer #3

"Apache Spark's components are more efficient than MapReduce" – True or False?

A	**True**
B	False

+91-9030485102

rganesh0203@gmail.com

https://topmate.io/rganesh_0203