Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant

Introduction to Big Data

Introduction, Big Data Ecosystem

AGENDA

1. Introduction to Big Data
2. Overview of Big Data ecosystem and tools
3. Hands-on with Big Data Tools

What do you think existed before the universe began? How did the universe begin?

1. Introduction to Big Data

"Where have you heard about Big Data, and how do you think it impacts our lives?"

Here are some examples to inspire your thinking:

- Social media platforms store and analyze user interactions to personalize feeds.
- Online shopping websites use data to recommend products you might like.
- Traffic management systems use IoT devices to monitor and predict congestion in real time.
- Weather forecasting uses massive datasets from sensors and satellites to make accurate predictions.

Big Data refers to datasets that are so **large**, **fast**, or **complex** that traditional data processing methods cannot handle them effectively.

These datasets require specialized tools and techniques to store, process, and analyze.

Why is Big Data important?
In a world where data is constantly generated from devices, applications, and sensors, Big Data enables us to:

- Gain deeper insights into complex problems.
- Make informed decisions in real time.
- Build innovative solutions like recommendation systems, fraud detection, and more.

The Five V's of Big Data

To better understand Big Data, we use the **Five V's** framework. Each "V" represents a key characteristic that defines Big Data.

1. Volume

- The amount of data generated is enormous, often measured in terabytes or even petabytes.
- **Example:** Social media platforms like Twitter and Instagram generate terabytes of data daily through posts, likes, and comments.

2. Velocity

- The speed at which data is generated and needs to be processed.
- **Example:** Stock market systems generate data in milliseconds, requiring real-time analysis to make predictions and decisions.

3. Variety

- Data comes in multiple formats, including:
  - **Structured:** Tabular data in databases.
  - **Unstructured:** Text, images, videos, etc.
  - **Semi-structured:** JSON, XML, etc.
- **Example:** A company might analyze customer emails (text), product images (pictures), and transaction records (structured data).

4. Veracity

- Refers to the uncertainty or accuracy of the data.
- Cleaning and validating noisy or incomplete data are critical for meaningful insights.
- **Example:** Customer reviews often have typos or inconsistencies that need to be addressed before analysis.

5. Value

- The ultimate goal of Big Data is to derive **value**, insights that drive better decisions or create innovative solutions.
- **Example:** E-commerce websites use data to create personalized product recommendations, leading to improved user experiences and increased sales.

Real-World Applications of Big Data

Big Data is transforming industries across the globe. Here are some examples:

1. Healthcare

- Predict patient outcomes by analyzing data from electronic medical records (EMRs), wearable devices, and diagnostic tools.
- Example: Use Big Data to detect early signs of diseases like cancer through pattern recognition.

2. E-commerce

- Personalized recommendations are powered by analyzing user behavior, search history, and purchase patterns.
- Example: Amazon suggests products based on your browsing and purchase history.

3. Smart Cities

- Manage traffic, energy, and public services using IoT sensors and Big Data analytics.
- Example: Smart traffic lights adjust timings based on real-time traffic data to reduce congestion.

Show a brief video clip on Big Data applications (3–5 minutes).

2. Overview of Big Data ecosystem and tools

Now that we understand what Big Data is and its characteristics, let's dive into the tools and technologies that make working with Big Data possible.

Overview of Big Data Tools and Technologies

The Big Data ecosystem consists of a variety of tools designed for storing, processing, and analyzing massive datasets. Let's explore some of the most widely used tools:

- **Hadoop**: A framework for batch processing of large datasets.
  - Works by breaking down massive data into smaller chunks and processing them in parallel.
  - Ideal for scenarios where data can be processed in batches over time.
  - Example: Analyzing logs from a web server for monthly traffic trends.
- **Spark**: A fast and distributed computing framework.
  - Processes data in-memory, making it significantly faster than Hadoop for many tasks.
  - Can handle both batch and stream processing.
  - Example: Real-time fraud detection in financial transactions.
- **NoSQL Databases**: Designed for non-relational data storage and processing.
  - Examples include **MongoDB** and **Cassandra**.
  - Suitable for flexible data models like documents, key-value pairs, or wide-column stores.
  - Example: Storing user-generated content like reviews, images, or logs in an e-commerce system.

Batch Processing vs. Stream Processing

Big Data processing can be broadly categorized into two types:

- **Batch Processing**:
  - Data is collected over a period, stored, and processed in chunks.
  - Tools like **Hadoop** are optimized for batch jobs.
  - **Example**: Generating daily sales reports for a company.
  - **Pros**: Suitable for large datasets that don't require immediate analysis.
- **Stream Processing**:
  - Data is processed in real time as it is generated.
  - Tools like **Spark Streaming** are used for this purpose.
  - **Example**: Monitoring stock prices and triggering alerts for anomalies.
  - **Pros**: Enables real-time decision-making for time-sensitive applications.

---

1. **Apache Spark**
   - A fast and general-purpose cluster-computing system for large-scale data processing.
2. **Hadoop (HDFS & MapReduce)**
   - A distributed storage and batch processing framework for big data.
3. **Apache Kafka**
   - A distributed event streaming platform for high-throughput real-time data pipelines.
4. **Tableau**
   - A powerful data visualization tool for creating interactive dashboards.
5. **Amazon S3**
   - A cloud-based object storage service for scalable and durable data storage.

6. **Google BigQuery**
    - A fully-managed, serverless data warehouse for fast SQL analytics on large datasets.
7. **Power BI**
    - A business intelligence tool for data visualization and interactive reporting.
8. **Elasticsearch (ELK Stack)**
    - A distributed search engine for real-time log analysis and monitoring.
9. **Pandas (Python)**
    - A library for data manipulation and analysis with Python.
10. **Presto**
    - A distributed SQL query engine for running analytics on large data sets.
11. **Apache Cassandra**
    - A highly scalable NoSQL database for handling large volumes of structured data.
12. **Apache Flink**
    - A stream processing framework for real-time analytics.
13. **Apache Hive**
    - A data warehouse system for querying and analyzing large datasets using SQL.
14. **Apache Airflow**
    - A workflow orchestration tool for scheduling and automating data pipelines.
15. **Google Cloud Dataflow**
    - A unified stream and batch processing system for real-time data pipelines.
16. **Apache Nifi**
    - A tool for automating and managing the flow of data between systems.
17. **R**

- A programming language and environment for statistical computing and graphics.

18. **TensorFlow on Spark**
    - A framework for running deep learning workloads on Spark clusters.
19. **Apache Mahout**
    - A scalable machine learning library for clustering, classification, and collaborative filtering.
20. **H2O.ai**
    - An open-source machine learning platform for scalable model training.
21. **Apache Oozie**
    - A workflow scheduler for managing Hadoop jobs.
22. **Talend**
    - A tool for ETL (Extract, Transform, Load) and data integration workflows.
23. **D3.js**
    - A JavaScript library for creating dynamic and interactive data visualizations.
24. **Splunk**
    - A platform for analyzing machine-generated data to gain operational intelligence.
25. **Apache Ranger**
    - A security tool for managing access control in big data ecosystems.

Apache Hadoop

Hadoop

What is Hadoop?

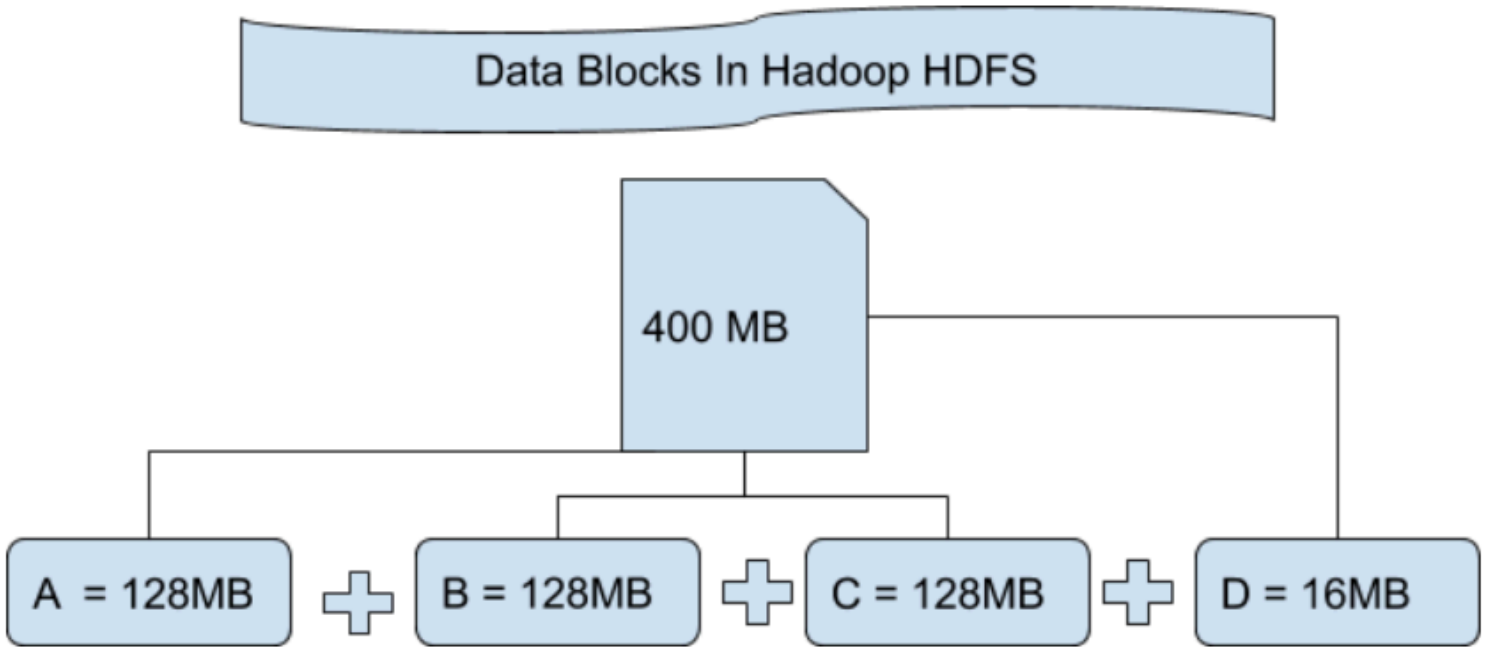
Apache Hadoop is an open-source framework designed for distributed storage and processing of large datasets across clusters of computers.

It allows organizations to store and analyze massive amounts of structured and unstructured data efficiently.

Key Components of Hadoop

HDFS Architecture

1. **HDFS (Hadoop Distributed File System)**
    - A distributed file system that stores data across multiple machines, ensuring redundancy and fault tolerance.
    - **Example:** A 1TB file can be split into smaller chunks (blocks) and stored across multiple machines. If one machine fails, the data remains accessible due to replication.
    - **How it works:**



- In our local PC, by default the block size in Hard Disk is 4KB. When we install Hadoop, the HDFS by default changes the block size to 64 MB.
- Since it is used to store huge data. We can also change the block size to 128 MB.
- Now HDFS works with Data Node and Name Node.
- While Name Node is a master service and it keeps the metadata as for on which commodity hardware, the data is residing, the Data Node stores the actual data.
- Now, since the block size is of 64 MB thus the storage required to store metadata is reduced thus making HDFS better.
- Also, Hadoop stores three copies of every dataset at three different locations. This ensures that the Hadoop is not prone to single point of failure.

2. **MapReduce**
    - A programming model for processing large datasets in a parallel and distributed manner. It consists of two main tasks:
        - **Map:** Processes data and transforms it into key-value pairs.
        - **Reduce:** Aggregates or processes the mapped data to produce meaningful results.
    - **Example:** Counting the frequency of words in a large document using MapReduce.
3. **YARN (Yet Another Resource Negotiator)**
    - Manages cluster resources and schedules jobs for execution. It ensures optimal resource utilization.
    - **Example:** Allocating memory and CPU to different MapReduce jobs running on the cluster.
4. **Hadoop Common**
    - A collection of libraries and utilities required by other Hadoop modules.

Features of Hadoop

- **Scalability:** Can handle petabytes of data by adding more nodes to the cluster.
- **Fault Tolerance:** Data is replicated across nodes, ensuring no data is lost even if a node fails.

-
- **Cost-Effective:** Runs on commodity hardware, making it more affordable for organizations.
- **Support for Multiple Data Formats:** Handles structured, semi-structured, and unstructured data.

How Hadoop Works

1. **Data Storage:**
   Data is stored in HDFS, which splits large files into blocks (e.g., 128 MB) and distributes them across the cluster.
2. **Data Processing:**
   - The MapReduce framework processes data stored in HDFS by splitting tasks across nodes.
   - Each node processes its assigned block independently, reducing the time taken for computation.

Hadoop Ecosystem

Hadoop integrates with a variety of tools for added functionality:

- **Hive:** SQL-like querying for data in HDFS.
- **Pig:** A scripting platform for large-scale data analysis.
- **HBase:** A NoSQL database for real-time data processing.
- **Sqoop:** Imports and exports data between Hadoop and relational databases.
- **Flume:** Collects and ingests log data into HDFS.

Apache Ambari
Management & Monitoring
Ambari
Apache Sqoop
Batch Data
Apache Flume
g Collector (Stream Data)
pache ZooKeeper
ordination
Apache Oozie
Workflow
Apache Pig
Scripting
Apache Hive
SQL Query
Apache HBase
Columnar Storage
YARN (MapReduce V2)
Distributed Processing Framework
HDFS
Hadoop Distributed File System

When to Use Hadoop

- When dealing with large datasets that exceed the storage and processing capabilities of a single machine.
- Batch processing scenarios, such as log analysis or data archiving.

Limitations of Hadoop

1. **Batch Processing Only:** Hadoop MapReduce is not suitable for real-time data processing.
2. **Latency:** Processing can be slower compared to modern tools like Apache Spark.
3. **Complexity:** Requires advanced knowledge to implement and maintain.

Apache Spark

Spark

What is Apache Spark? Apache Spark is an open-source, distributed computing system designed for fast and versatile big data processing.

It extends the capabilities of Hadoop by providing real-time processing, in-memory computation, and a flexible programming interface.

Key Components of Apache Spark

1. **Spark Core**
    - The foundational engine responsible for memory management, fault tolerance, and distributed task execution.
    - Provides APIs in Python (PySpark), Java, Scala, and R.

2. **Spark SQL**
    - Enables querying structured and semi-structured data using SQL-like syntax.
    - **Example:** Querying JSON or Parquet files directly using SQL commands.
3. **Spark Streaming**
    - Processes real-time data streams from sources like Kafka, Flume, or socket streams.
    - **Example:** Real-time fraud detection in credit card transactions.
4. **MLlib (Machine Learning Library)**
    - A scalable library for machine learning tasks such as classification, regression, clustering, and collaborative filtering.
    - **Example:** Building a recommendation system for e-commerce websites.
5. **GraphX**
    - A library for graph processing and analysis, enabling tasks like PageRank and community detection.
    - **Example:** Social network analysis or influence detection in networks.

Features of Apache Spark

- **In-Memory Processing:** Data is processed in memory, significantly speeding up computation.
- **Real-Time Data Processing:** Unlike Hadoop, Spark can process streaming data in near real-time.
- **Ease of Use:** Offers APIs for multiple languages (Python, Scala, Java, R).
- **Unified Framework:** Combines batch processing, streaming, and machine learning in one system.
- **Compatibility with Hadoop:** Works seamlessly with Hadoop's HDFS and other ecosystem tools.

How Spark Works

1. **Driver Program:**
    - The entry point of a Spark application that coordinates all operations.
2. **Cluster Manager:**
    - Allocates resources for tasks (e.g., Spark Standalone, YARN, Mesos).

3. **Executors:**
    - Run tasks and manage data storage in memory or disk.

Spark Ecosystem

- **Data Sources:** Spark integrates with HDFS, Cassandra, MongoDB, and more.
- **Cluster Managers:** Supports Standalone, YARN, and Mesos.
- **Storage Formats:** Handles Parquet, ORC, Avro, JSON, CSV, and more.

When to Use Spark

1. **Real-Time Data Processing:** Use Spark Streaming for scenarios like stock price monitoring or IoT applications.
2. **Batch and Stream Processing:** Unified framework eliminates the need for separate tools.
3. **Machine Learning Pipelines:** Build and deploy scalable ML models with MLlib.

Limitations of Spark

1. **Memory-Intensive:** Requires significant memory for in-memory computation.
2. **Cost:** More resource-heavy compared to Hadoop.
3. **Steeper Learning Curve:** Advanced optimizations may require deeper expertise.

Spark vs. Hadoop

Feature	Apache Spark	Apache Hadoop
Processing Mode	Batch + Real-Time	Batch Only
Speed	Faster (In-Memory)	Slower (Disk-Based)
Ease of Use	APIs for multiple languages	Complex MapReduce setup
Fault Tolerance	High	High

3. Hands-on with Big Data Tools

Setting up Hadoop/Spark environment (use cloud-based sandbox tools) Loading and processing a sample dataset

Live Exercise

Now it's your turn!

Task 1: description of task

```
- instructions
```

END

THANK YOU!

Live Exercise Solutions

In [2]:
```
solutions
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 solutions

NameError: name 'solutions' is not defined
```

Programming Interveiw Questions

1. topic:
    - question

In [ ]:
```

```

Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant