Definition of Big Data

- Big Data refers to **very large, complex, and rapidly generated datasets**
- Cannot be **stored, processed, or analyzed** using traditional database systems
- Includes:
  - **Structured data**
  - **Semi-structured data**
  - **Unstructured data**
- Requires **distributed storage and parallel processing**
- Characterized by **5 V’s**:
  - Volume
  - Velocity
  - Variety
  - Veracity
  - Value

Evolution of Big Data (Main 4 Eras)

---

1. Traditional Data Era (Before 1990s)

- Data was:
  - Small in size
  - Highly structured
  - Centrally stored
- Technologies used:
  - File systems
  - Relational Database Management Systems (RDBMS)

    - SQL
- Examples:
    - Banking transaction records
    - Payroll systems
    - Student information systems
- Limitations:
    - No support for unstructured data
    - Limited scalability
    - High cost of system expansion

---

2. Data Warehousing & Business Intelligence Era (1990s – Early 2000s)

- Growth of enterprise-level data
- Focus on **historical data analysis**
- Technologies used:
    - Data warehouses
    - ETL (Extract, Transform, Load) tools
    - OLAP systems
- Features:
    - Centralized storage
    - Batch processing
    - Structured reporting
- Examples:
    - Sales analysis
    - Financial reporting
    - Customer profiling

- Limitations:
  - Expensive infrastructure
  - Poor handling of unstructured data
  - No real-time analytics

---

3. Web and Social Media Data Era (Mid-2000s)

- Rapid increase in internet usage
- Data became:
  - Huge in volume
  - Mostly unstructured
  - Continuously generated
- Data sources:
  - Social media platforms
  - Websites
  - Multimedia content
- Examples:
  - Facebook posts and likes
  - YouTube videos
  - Web server logs
- Challenges:
  - Traditional databases failed to scale
  - Storage and processing inefficiency

---

4. Big Data Era (Hadoop & Real-Time Analytics) (Late 2000s – Present)

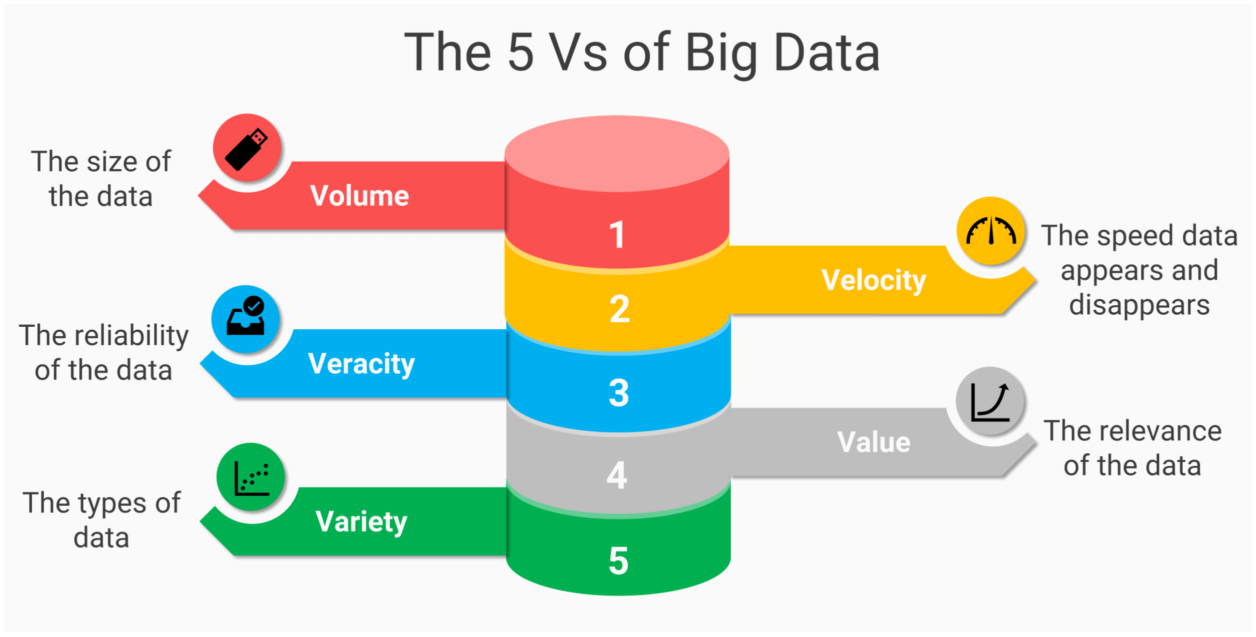
- Introduction of **distributed computing**
- Technologies used:

    - Hadoop (HDFS, MapReduce)
    - Apache Spark
    - NoSQL databases
    - Cloud platforms
- Features:
    - Distributed storage
    - Parallel processing
    - Fault tolerance
    - Real-time analytics
- Examples:
    - Recommendation systems (Netflix, Amazon)
    - Fraud detection in banking
    - IoT sensor data analysis
    - Search engine indexing

Conclusion

- Big Data evolved from:
    - Small, structured data systems
    - To large, distributed, real-time data environments
- Modern Big Data technologies enable:
    - Faster decision-making
    - Advanced analytics
    - AI and machine learning applications

Characteristics of Big Data – 5 V's (VTU Style, Detailed)



1. Volume

- Refers to the **huge amount of data** generated and stored
- Data size ranges from:
    - Terabytes (TB)
    - Petabytes (PB)
    - Exabytes (EB) and beyond
- Generated from:
    - Social media platforms
    - Sensors and IoT devices
    - Transaction systems
    - Multimedia (images, videos)
- Traditional databases cannot handle such massive data volumes

- Requires:
    - Distributed storage systems
    - Scalable architectures

Example:

- Facebook generates terabytes of user data daily
- E-commerce websites store millions of transaction records

---

2. Velocity

- Refers to the **speed at which data is generated, transmitted, and processed**
- Data is produced:
    - Continuously
    - In real time or near real time
- High velocity data needs **fast processing and quick decision-making**
- Batch processing is often insufficient

Sources of high-velocity data:

- Stock market transactions
- Online payments
- Live social media feeds
- Sensor data from IoT devices

Example:

- Real-time fraud detection in banking
- Live traffic updates in navigation apps

---

3. Variety

- Refers to **different types and formats of data**
- Big Data includes:

    - **Structured data** (tables, rows, columns)
    - **Semi-structured data** (XML, JSON, logs)
    - **Unstructured data** (text, images, videos, audio)
- Traditional systems are designed mainly for structured data
- Big Data systems can handle all formats

Example:

- Emails (text)
- Social media posts (text + images)
- CCTV videos
- Server log files

---

4. Veracity

- Refers to the **quality, accuracy, and reliability of data**
- Big Data often contains:
    - Noise
    - Duplicates
    - Incomplete or inconsistent data
- Poor data quality can lead to:
    - Incorrect analysis
    - Wrong business decisions
- Data cleaning and validation are essential

Challenges:

- Fake social media data
- Sensor errors
- Human-generated data inconsistencies

Example:

- Fake reviews on e-commerce platforms
- Incorrect readings from faulty sensors

---

5. Value

- Refers to the **usefulness and meaningful insights** extracted from data
- Large volumes of data are useless without proper analysis
- The main goal of Big Data analytics is to:
  - Discover patterns
  - Predict trends
  - Support decision-making
- Value depends on effective analytics and interpretation

Example:

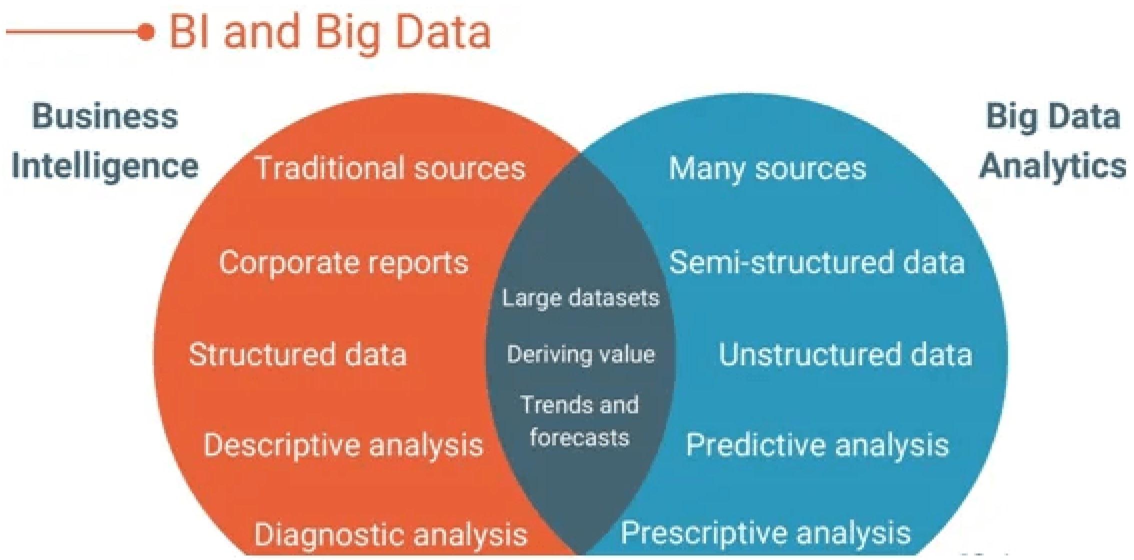
- Customer behavior analysis for targeted marketing
- Predictive maintenance in industries
- Personalized recommendations (Netflix, Amazon)

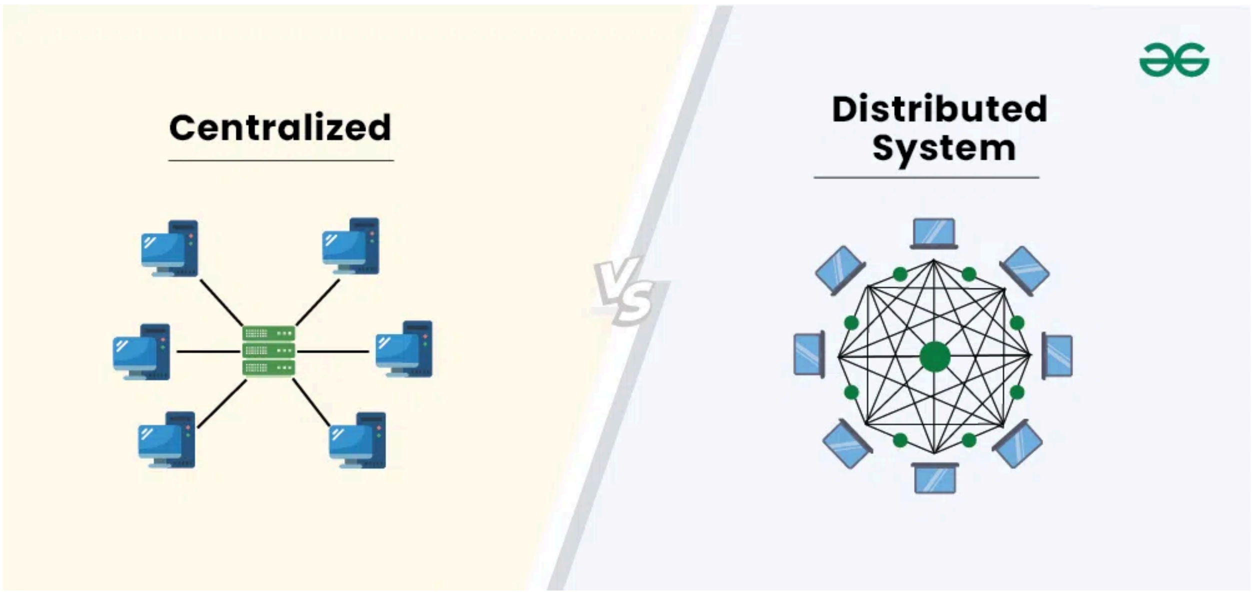
Conclusion

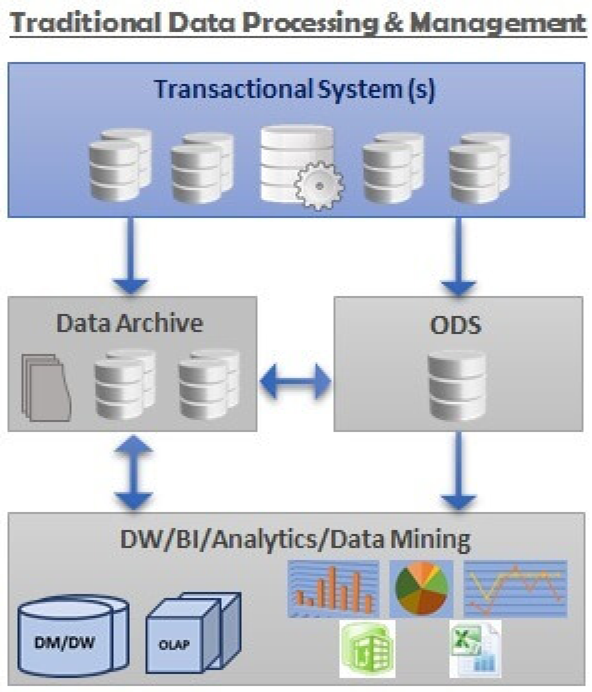
- The **5 V’s** describe the fundamental characteristics of Big Data
- They explain why traditional systems are insufficient
- Understanding these characteristics helps in:
  - Designing Big Data systems
  - Choosing appropriate tools and technologies
  - Extracting maximum business and analytical value

Comparison of Traditional Data Processing Systems and Big Data Systems

Traditional Data Processing Systems and Big Data Systems differ significantly in terms of data size, data types, architecture, and processing methods. Traditional systems were designed to handle structured, limited-volume data, whereas Big Data systems are built to manage massive, fast, and diverse datasets generated in modern digital environments.






Traditional Data Processing & Management
Transactional System (s)
Data Archive
ODS
DW/BI/Analytics/Data Mining
DM/DW
OLAP

1. Traditional Data Processing Systems (Detailed Explanation)

Traditional data processing systems were developed to manage data generated by early business and enterprise applications.

- Designed to handle **small to moderate volumes of structured data**
- Data is stored in **relational databases (RDBMS)** using tables, rows, and columns
- Uses **centralized architecture** (single server or limited servers)
- Processing is mainly **batch-oriented**
- Uses **SQL** for querying and manipulation
- Scaling is done by **vertical scaling** (upgrading hardware)
- Limited support for unstructured or semi-structured data
- High cost when data volume increases

Examples:

- Banking transaction systems
- Payroll and accounting systems
- Student information systems

---

2. Big Data Systems (Detailed Explanation)

Big Data systems were introduced to overcome the limitations of traditional systems and handle modern data challenges.

- Designed to handle **very large volumes of data (TBs–EBs)**
- Supports **structured, semi-structured, and unstructured data**
- Uses **distributed architecture** (cluster of commodity machines)
- Supports **batch processing and real-time/stream processing**
- Uses frameworks like Hadoop and Spark
- Scaling is done by **horizontal scaling** (adding more nodes)
- Built-in **fault tolerance and high availability**
- Cost-effective for large-scale data processing

Examples:

- Social media data analysis
- Recommendation systems (Netflix, Amazon)
- IoT sensor data processing
- Real-time fraud detection

---

3. Column-wise Comparison

Aspect	Traditional Data Processing Systems	Big Data Systems
Data Volume	Small to moderate	Very large (TBs–EBs)
Data Type	Structured only	Structured, semi-structured, unstructured

Aspect	Traditional Data Processing Systems	Big Data Systems
Architecture	Centralized	Distributed
Storage	Relational databases	Distributed file systems and NoSQL
Processing	Batch processing	Batch and real-time processing
Scalability	Vertical scaling	Horizontal scaling
Fault Tolerance	Limited	High (replication, recovery)
Cost	Expensive at large scale	Cost-effective using commodity hardware
Flexibility	Low	High
Examples	Payroll, banking systems	Social media, IoT, analytics platforms

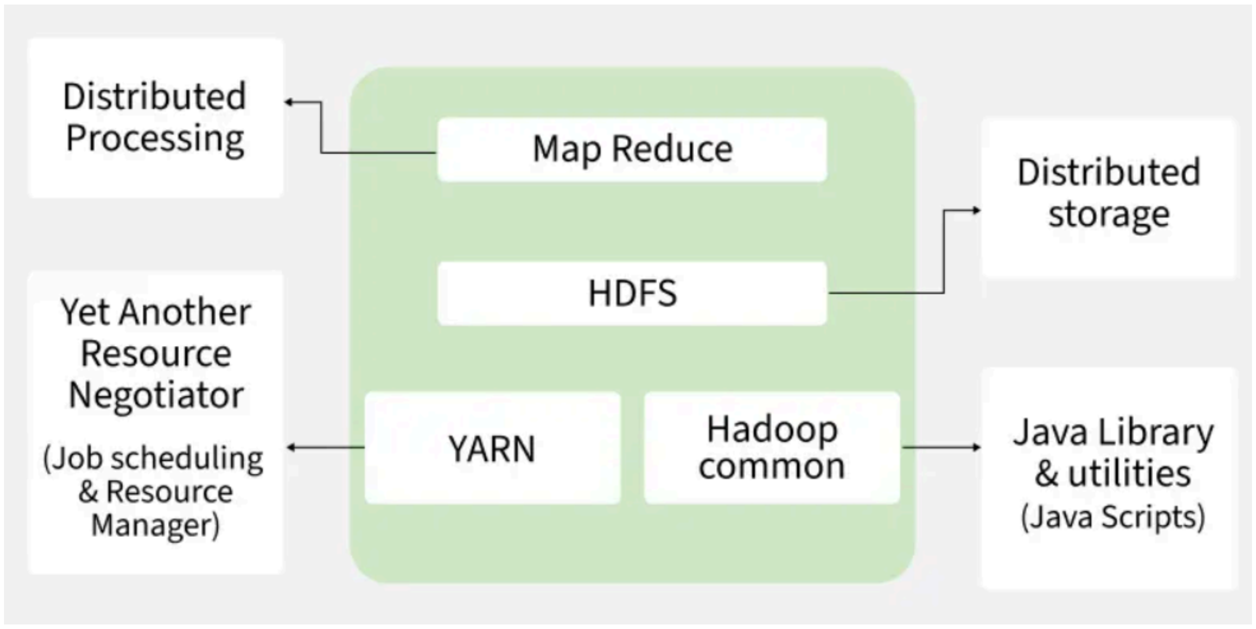
---

Conclusion

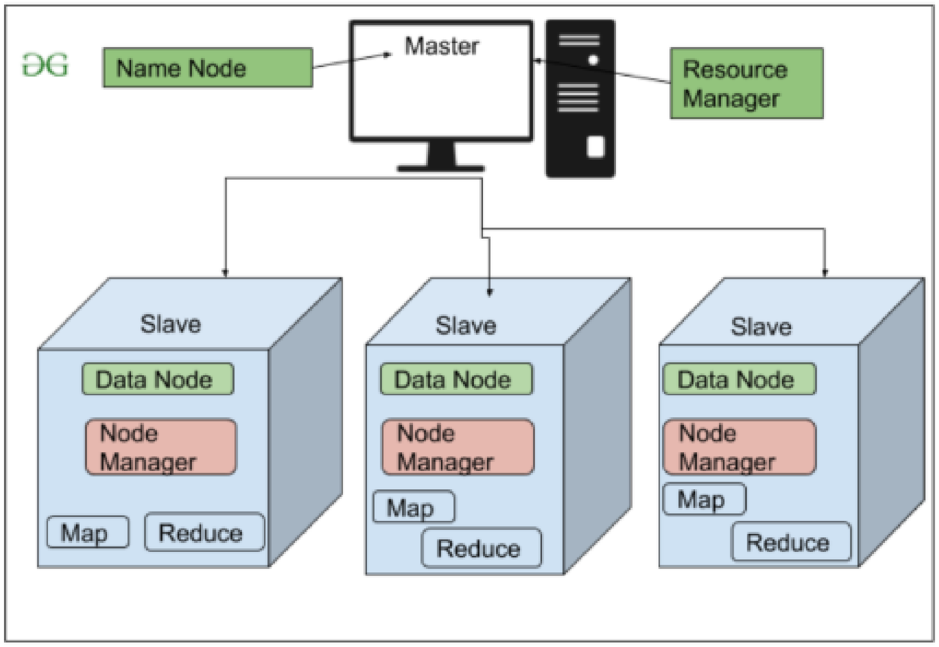
- Traditional systems are suitable for **small, structured, and stable data**
- Big Data systems are designed for **large-scale, fast, and diverse data**
- Modern applications require Big Data systems for **scalability, flexibility, and real-time insights**

Hadoop Ecosystem – Explanation with Architecture Diagram

The **Hadoop ecosystem** is a collection of **open-source tools and frameworks** designed to store, process, manage, and analyze **large-scale data** efficiently using distributed computing. Hadoop provides a scalable, fault-tolerant, and cost-effective solution for handling Big Data across clusters of commodity hardware.


Distributed Processing
Map Reduce
Distributed storage
HDFS
Yet Another Resource Negotiator (Job scheduling & Resource Manager)
YARN
Hadoop common
Java Library & utilities (Java Scripts)

1. Hadoop Distributed File System (HDFS)



HDFS is the storage layer of Hadoop.

- Stores very large files across multiple nodes
- Files are split into fixed-size blocks
- Blocks are replicated for fault tolerance

- Optimized for high-throughput access

HDFS Components:

- **NameNode**
  - Master node
  - Stores metadata (file name, block location)
  - Manages file system namespace
- **DataNode**
  - Slave node
  - Stores actual data blocks
  - Performs read/write operations
- **Secondary NameNode**
  - Periodically checkpoints metadata
  - Assists NameNode (not a backup)

---

2. YARN (Yet Another Resource Negotiator)

YARN is the resource management layer.

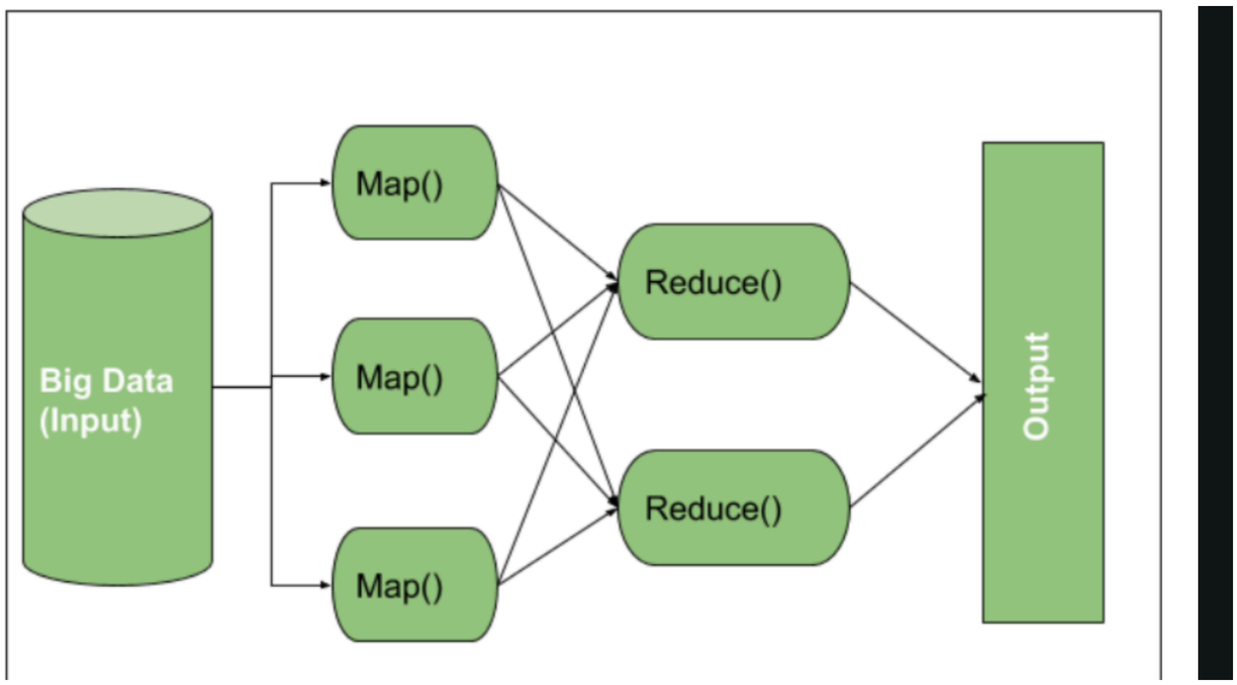
- Manages cluster resources (CPU, memory)
- Schedules and monitors jobs
- Allows multiple applications to run simultaneously

YARN Components:

- **ResourceManager**
  - Global resource scheduler
  - Allocates resources to applications
- **NodeManager**
  - Runs on each node
  - Monitors resource usage

- **ApplicationMaster**
    - Manages lifecycle of an application
    - Requests resources from ResourceManager

3. MapReduce



MapReduce is the data processing framework.

- Processes data in parallel across nodes
- Works in two phases:
    - **Map phase –** processes input data and generates key-value pairs
    - **Reduce phase –** aggregates and processes output from Map phase
- Suitable for batch processing
- Runs on top of YARN

4. Hadoop Common

Hadoop Common provides shared utilities and libraries.

- Contains Java libraries and utilities
- Supports other Hadoop modules
- Includes configuration files and scripts

1. Hive

Apache Hive is a **data warehousing tool** used for querying and analyzing large datasets stored in HDFS.

- Provides **SQL-like language (HiveQL)**
- Converts queries into MapReduce/Spark jobs
- Suitable for **batch processing and analytics**
- Used by users with SQL background
- Example: Sales analysis, log analysis

---

2. Pig

Apache Pig is a **data processing and ETL tool** used to analyze large datasets.

- Uses a scripting language called **Pig Latin**
- Simplifies complex data transformations
- Automatically converts scripts into MapReduce jobs
- Suitable for **data cleaning and preprocessing**
- Example: Filtering, joining, and aggregating large datasets

---

3. HBase

Apache HBase is a **NoSQL, column-oriented database** built on top of HDFS.

- Provides **real-time read and write access**
- Handles large, sparse datasets
- Supports random access to data

- Used when low-latency access is required
- Example: Real-time analytics, IoT data storage

Conclusion

- **Hive** → SQL-based data analysis
- **Pig** → Data transformation and ETL
- **HBase** → Real-time NoSQL data storage

If you want, I can also:

- Write this in **VTU 5-mark format**
- Add **comparison table**
- Reduce it to **short exam notes**