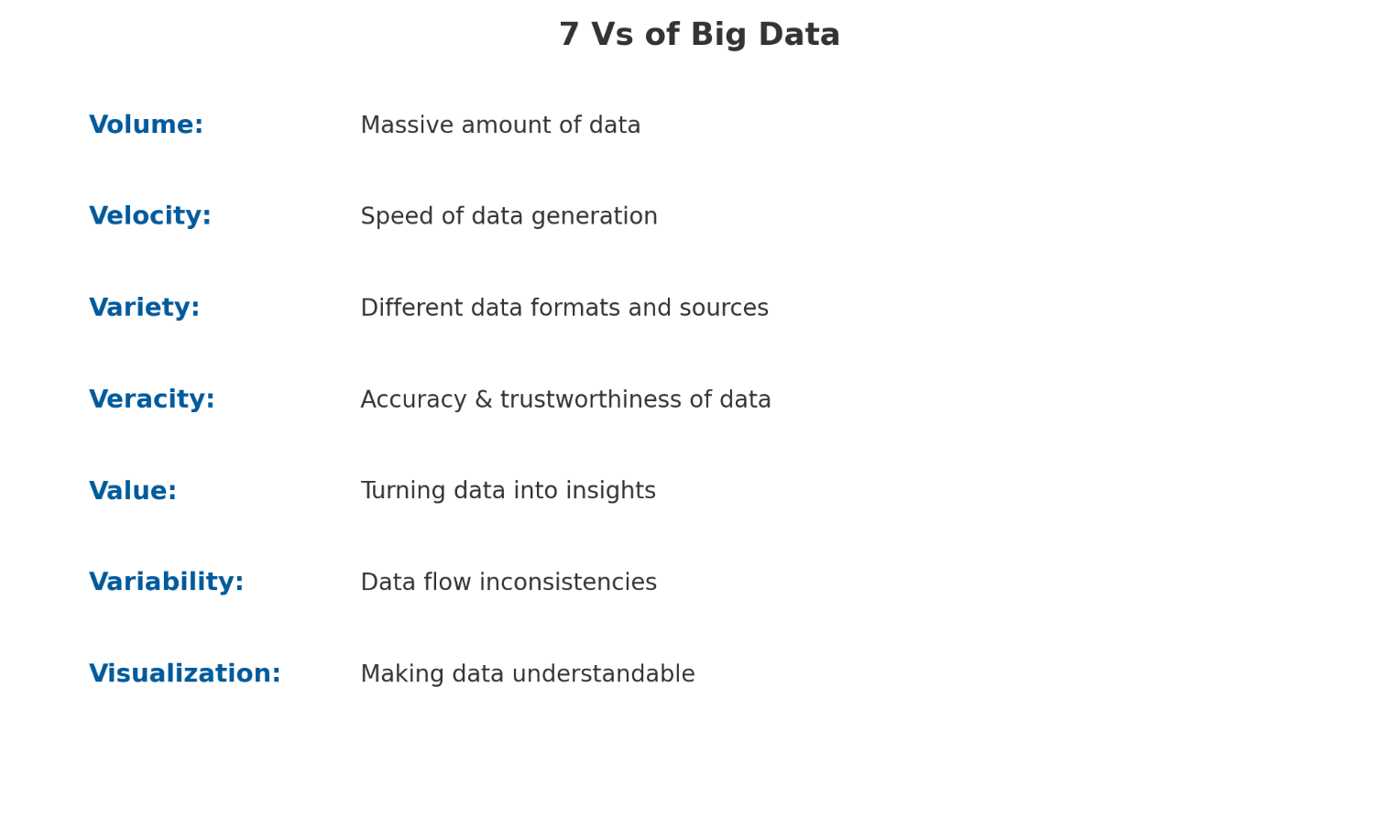
**Day 7 - Big Data, GFS, Hadoop**

**📅** – 13/06/2025

**🔹What is Big Data?**

| **Definition** | **Big Data refers to extremely large datasets that traditional data processing tools cannot handle efficiently.** |
| --- | --- |
| **Interesting Take** | Imagine the entire internet data every second — that's Big Data! It's like a tsunami of information, non-stop. |

**🔹 7 Vs of Big Data**

**🔹 Advantages of Big Data**

| **Advantage** | **Explanation** |
| --- | --- |
| Better Decision Making | Data-driven insights enable smarter decisions. |
| Cost Reduction | Efficient storage & processing save infrastructure cost. |
| Personalized Services | Tailored marketing, recommendations, and care. |
| Fraud Detection | Pattern analysis detects anomalies in real time. |

**🔹 Challenges of Big Data**

| **Challenge** | **Explanation** |
| --- | --- |
| Data Quality | Noise, duplication, missing data. |
| Security | Need for strict access & usage controls. |
| Processing Speed | Real-time needs demand high performance. |
| Talent Shortage | Skilled professionals are limited. |

**🔹 Big Data Pipeline**

| **Stage** | **Description** |
| --- | --- |
| Data Ingestion | Collect data from sources (IoT, logs, APIs). |
| Storage | Store using HDFS, S3, NoSQL, or Data Lake. |
| Processing | Use Spark, Hadoop, or Flink to clean and transform. |
| Analytics | BI Tools, ML Models, or SQL Queries |
| Visualization | Dashboards, Reports (Tableau, Power BI, Grafana) |

**🔹 Traditional System vs Big Data System**

| **Aspect** | **Traditional System** | **Big Data System** |
| --- | --- | --- |
| Scale | GB to TB | TB to PB |
| Processing | Centralized | Distributed |
| Data Types | Structured only | Structured, Semi, Unstructured |
| Speed | Batch | Batch + Real-Time |
| Storage | RDBMS | HDFS, NoSQL, Data Lakes |

**🔹 Limitations of Traditional RDBMS Architecture**

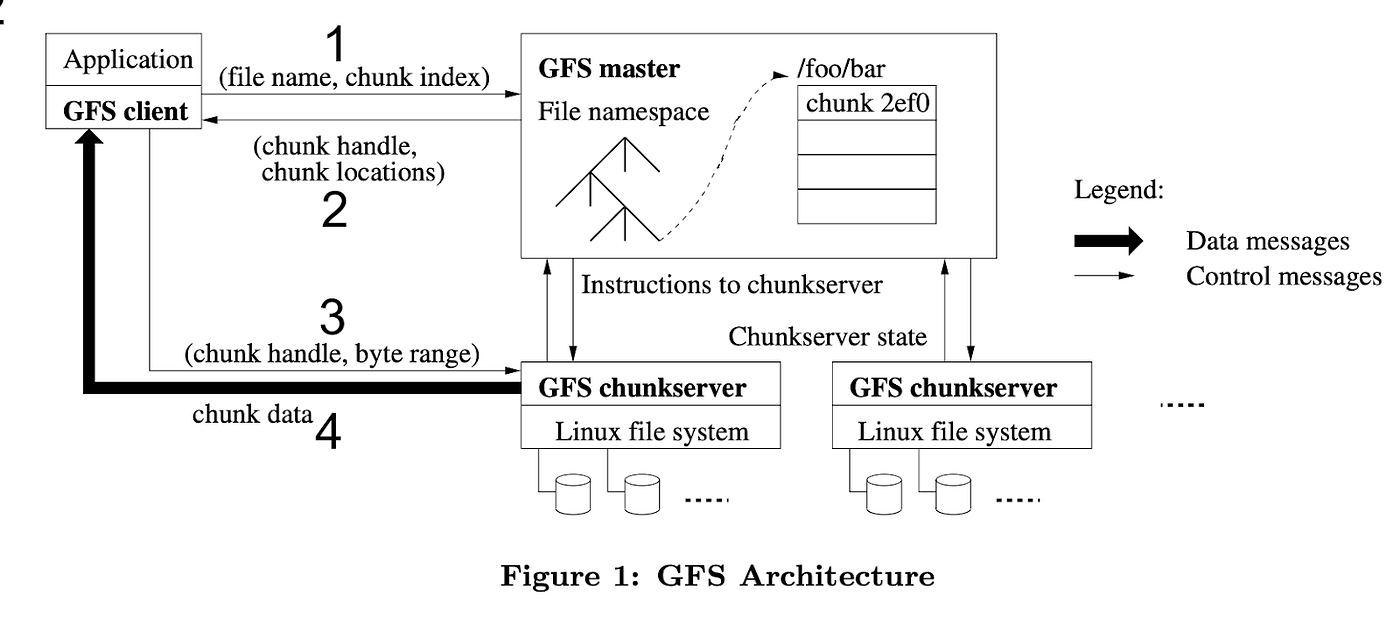
| **Limitation** | **Details** |
| --- | --- |
| Scalability Issues | Can't handle large-scale horizontally scalable workloads. |
| Schema-Rigidity | Difficult to adapt to dynamic or semi-structured data. |
| Expensive Scaling | Needs costly vertical hardware upgrades. |

**🔹 Distributed Architecture**

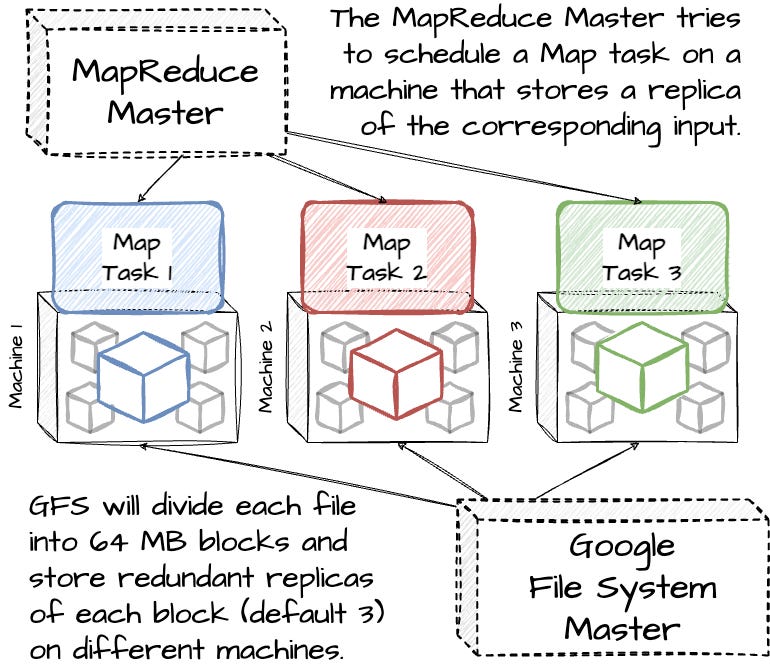
| **What is it?** | **Distributes processing across multiple machines for fault-tolerance, speed, and scalability.** |
| --- | --- |

**🔹 Google’s Big Data Challenge & Solution**

| **Challenge** | **Billions of web pages, files, and user data growing every second.** |
| --- | --- |
| **Solution** | **Google File System (GFS)** + **MapReduce** to handle massive-scale processing and storage. |

**🔹 Components of GFS**

| **Component** | **Function** |
| --- | --- |
| Master Server | Manages metadata (file namespace, chunk locations). |
| Chunk Servers | Stores fixed-size chunks of data (usually 64MB). |
| Clients | Request read/write operations from master and chunk servers. |



**🔹 How MapReduce Works with GFS**

* **Map Phase**: Break data into key-value pairs.
* **Shuffle Phase**: Organize by keys.
* **Reduce Phase**: Aggregate results.
* **GFS Role**: Stores the data chunks and outputs between Map and Reduce phases.

**🔹 What is Hadoop?**

| **Definition** | Hadoop is a framework that allows you to first store Big Data in a distributed environment, so that, you can process it parallely. There are basically two components in Hadoop

**🔹 Features of Hadoop**

| **Feature** | **Details** |
| --- | --- |
| Fault Tolerant | Uses data replication. |
| Scalable | Easily adds nodes. |
| Cost-Effective | Uses commodity hardware. |
| Flexible | Works with structured, semi-structured, unstructured data. |

**🔹 History of Hadoop**

| **Invented By** | **Doug Cutting and Mike Cafarella (inspired by Google Papers)** |
| --- | --- |

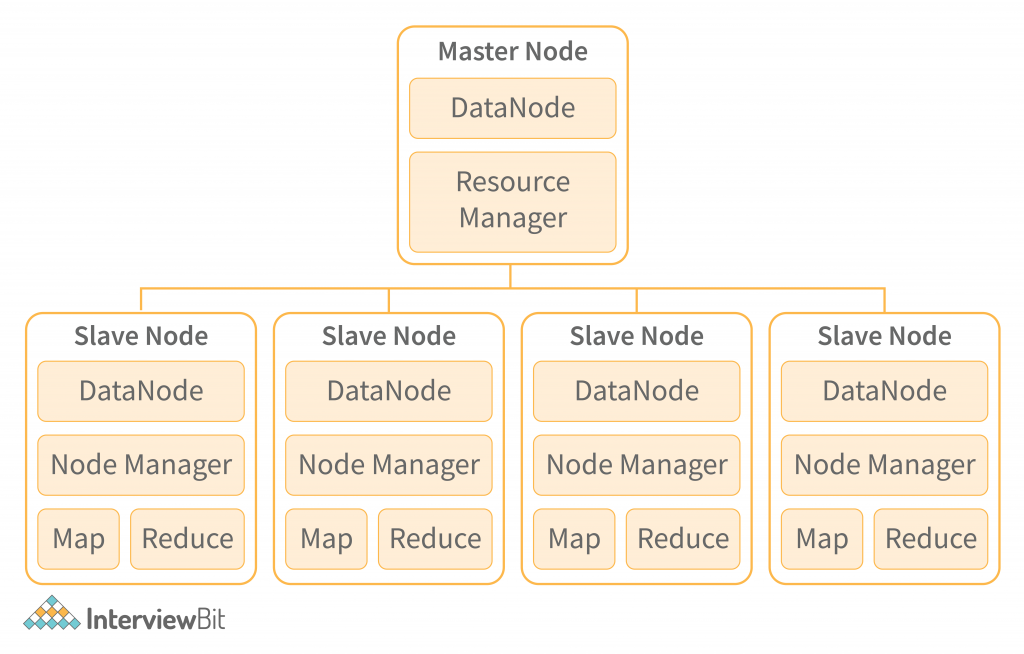
**🔹 Hadoop Components**

**🔹 1. Hadoop Common (Core Libraries)**

* The essential utilities and libraries that support other Hadoop modules.
* Provides the necessary Java files and scripts required to start Hadoop.

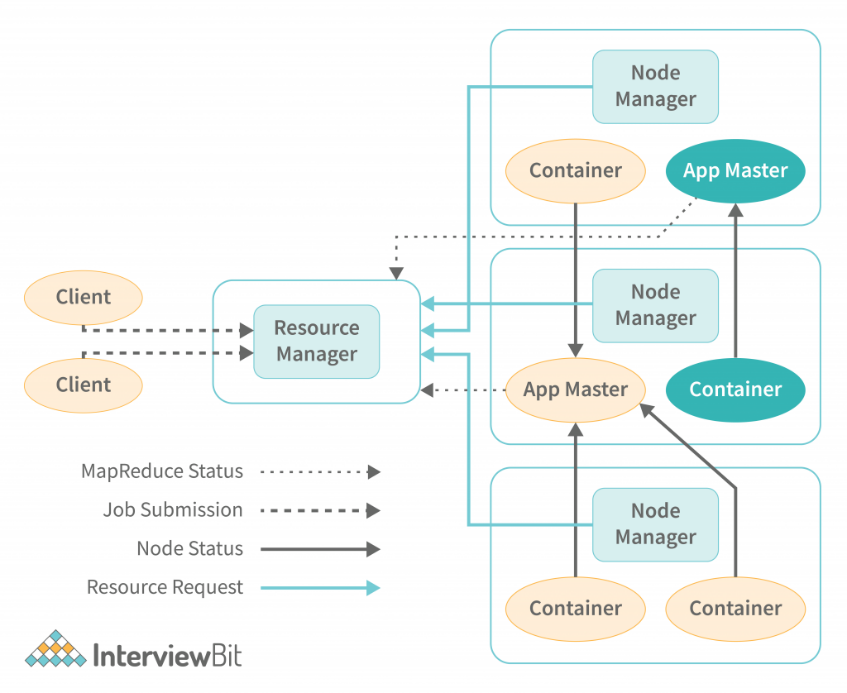
**🔹 2. HDFS (Hadoop Distributed File System) Architecture – Storage Layer**

* A distributed file system that stores data across multiple machines.
* **Components of HDFS:**

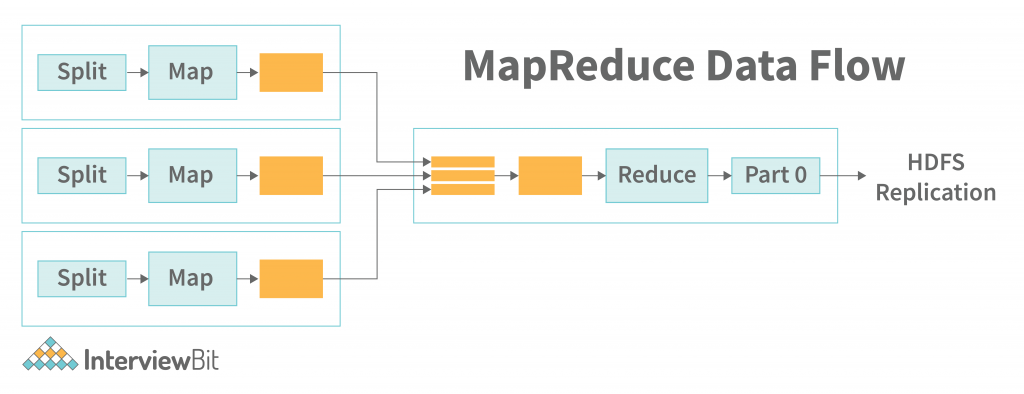


* + NameNode – Manages metadata and directory structure (e.g., file names, permissions).
  + DataNode – Stores the actual data blocks.
  + Secondary NameNode – Periodically merges the namespace image and edit logs to help NameNode recovery (not a backup).

**🔹 3. YARN (Yet Another Resource Negotiator) – Resource Management Layer**

* Manages resources and job scheduling for processing data.
* **Components of YARN:**
  + ResourceManager (RM) – Global resource manager and job scheduler.
  + NodeManager (NM) – Manages resources on a single node.
  + ApplicationMaster (AM) – Manages the execution of a specific job.

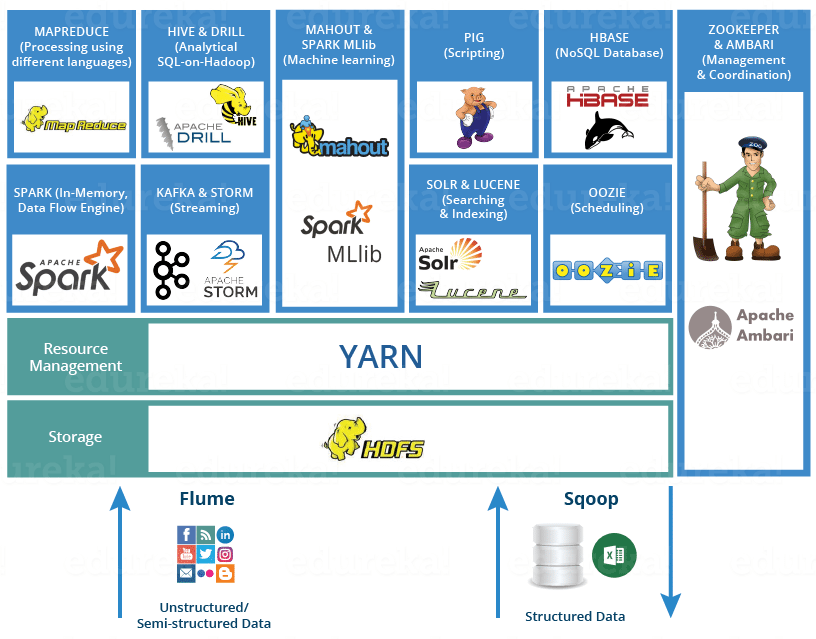
**🔹 4. MapReduce – Processing Layer**

* A programming model for processing large-scale data in parallel.
* Two phases:
  + Map – Filters and sorts data.
  + Reduce – Aggregates the results from the Map phase.

**🔹 5. Additional Ecosystem Tools**

* These are not core parts of Hadoop but commonly used with it:
  + Apache Hive – SQL-like querying over large datasets.
  + Apache Pig – High-level scripting for data transformation.
  + Apache HBase – NoSQL database on top of HDFS.
  + Apache Sqoop – Transfers data between Hadoop and relational databases.
  + Apache Flume – Collects and transfers large amounts of log data.
  + Apache Oozie – Workflow scheduler for Hadoop jobs.
  + Apache Zookeeper – Coordination service for distributed systems.

**🔹 Apache Hadoop Ecosystem**



| **Tool** | **Use** |
| --- | --- |
| HDFS | Storage |
| MapReduce | Processing |
| YARN | Resource management |
| Hive | SQL-like querying |
| Pig | Scripting language for analysis |
| HBase | NoSQL database |
| Sqoop | Transfer data between RDBMS & Hadoop |
| Flume | Ingest logs & streaming data |
| Oozie | Workflow scheduler |