Big Data

Turning massive data into powerful insights for smarter decisions.

Sudhir Lama

BIG DATA

- Big Data is a massive collection of structured, unstructured, and semi-structured data that is growing exponentially over time.
- It is a data set that is so large and complex that traditional data management tools cannot store or process it efficiently.
- Big data is a type of data that is extremely large in size.
- It enhances decision-making and business intelligence, providing organizations a competitive edge in today's fast-paced environment.

EXAMPLE OF BIG DATA

- The New York Stock Exchange, for example, generates approximately one terabyte of new trade data per day.
- The statistic shows that ***500+terabytes*** of new data get ingested into the databases of social media site **Facebook**, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.
- A single **Jet engine** can generate ***10+terabytes*** of data in ***30 minutes*** of flight time. With many thousand flights per day, generation of data reaches up to many ***Petabytes.***

WHY IS BIG DATA IMPORTANT?

- Companies use big data in their systems to improve operations, provide better customer service, create personalized marketing campaigns and take other actions that, ultimately, can increase revenue and profits.
- Big data is also used by medical researchers to identify disease signs and risk factors and by doctors to help diagnose illnesses and medical conditions in patients.
- In addition, a combination of data from electronic health records, social media sites, the web and other sources gives healthcare organizations and government agencies up-to-date information on infectious disease threats or outbreaks.

Uses of Big Data in Various Industries

Healthcare

- Patient data analysis for better treatment plans. - Predictive analytics for disease outbreaks.

Finance

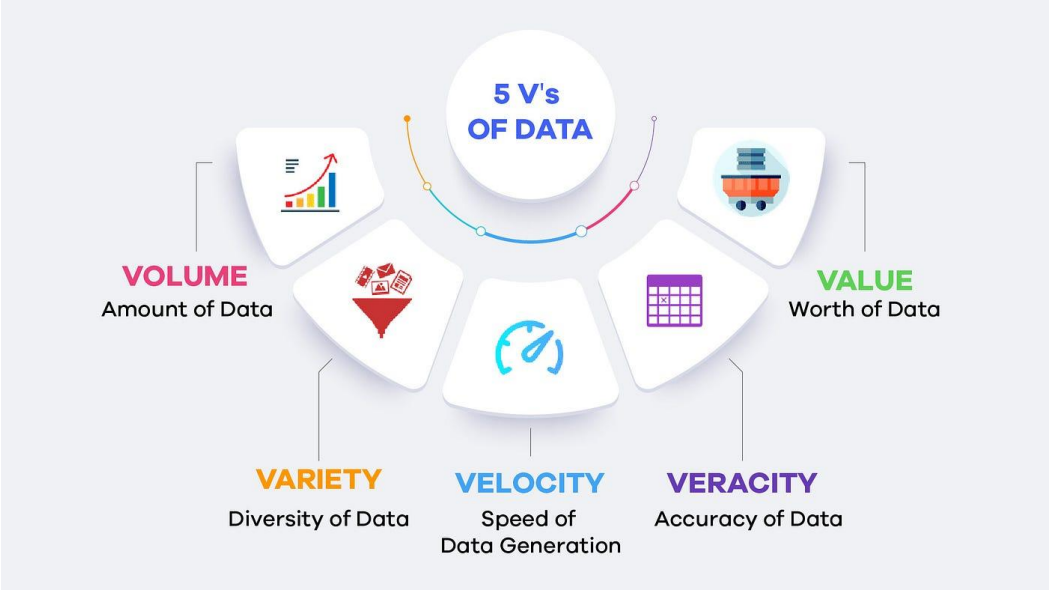
- Risk management and fraud detection. - Personalized financial services.

Marketing

- Customer behavior analysis for targeted campaigns. - Enhancing customer experiences and engagement.

Data Vs Big Data

- Big Data is just data with
  - More Volume
  - Faster Data data generation(Velocity)
  - Multiple Data format(Variety)



Distributed System in Big Data

- A distributed system is a network of independent computers that work together to appear as a single coherent system to the end-user.
- In the context of Big Data, distributed systems play a crucial role in processing and managing massive amounts of data efficiently.
- It is the backbone of Big Data processing and storage.
- It is crucial for managing large-scale applications that require high availability, fault tolerance, and scalability

Role of Distributed System in Big Data

- Data Storage
- Data Processing
- Load Balancing
- Fault Tolerance
- Real Time Data Processing
- Scalability

Challenges in Distributed System

- Complexity
- Latency
- Consistency
- Security

Big Data Analytics

BIG DATA ANALYTICS

- Big data analytics examines large amounts of data to uncover hidden patterns, correlations and other insights.
- Big data analytics helps organizations harness their data and use it to identify new opportunities.
- That, in turn, leads to smarter business moves, more efficient operations, higher profits and happier customers.

BENEFITS OF BIG DATA ANALYTICS

- Real-time forecasting and monitoring of business as well as the market.
- Identify crucial points hidden within large datasets to influence business decisions.
- Identify issues in systems and business processes in real-time.
- Dig in customer data to create tailor-made products, services, offers, discounts, etc.
- Facilitate speedy delivery of products/services that meet and exceed client expectations.

TYPES OF BIG DATA

- Structured
- Unstructured
- Semi-structured

STRUCTURED

- Structured Data is used to refer to the data which is already stored in databases, in an ordered manner.
- There are two sources of structured data;
  - Human-Generated
  - Machine-Generated
- All the data received from sensors, web logs and financial systems are classified under machine-generated data.
- Human-generated structured data mainly includes all the data a human input a computer, such as his name.

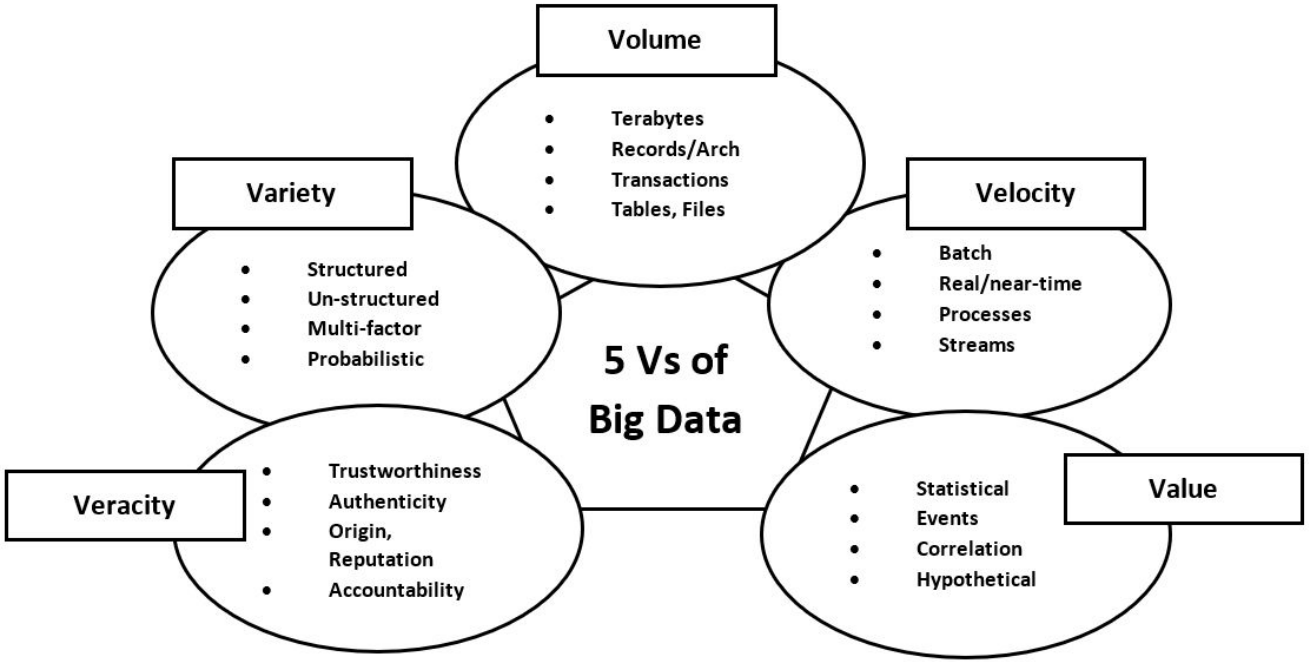
UN-STRUCTURED

- Unstructured data is defined as any data with an unknown form or structure.
- Aside from its massive size, unstructured data presents a number of challenges in terms of processing and extracting value from it.
- A heterogeneous data source containing a mix of simple text files, images, videos, and so on is an example of unstructured data.

SEMI-STRUCTURED

- Semi-structured data can contain both types of information.
- Semi-structured data appears to be structured, but it is not defined in the same way that a table definition in a relational DBMS is.
- A data representation in an XML file is an example of semi-structured data.

CHARACTERISTICS OF BIG DATA



VOLUME

- The name Big Data itself is related to a size which is enormous.
- Size of data plays a very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is dependent upon the volume of data.
- Hence, **Volume** is one characteristic which needs to be considered while dealing with Big Data solutions.
- For example;
    - Organizational data
    - Social media data

VELOCITY

- The term **‘velocity’** refers to the speed of generation of data.
- How fast the data is generated and processed to meet the demands, determines real potential in the data.
- Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks, and social media sites, sensors, Mobile devices, etc.
- The flow of data is massive and continuous.

VERACITY

- When we are dealing with a high volume, velocity and variety of data, it is not possible that all of the data is going to be 100% correct, there will be dirty data.
- The quality of the data being captured can vary greatly.
- The data accuracy of analysis depends on the veracity of the source data.

VALUE

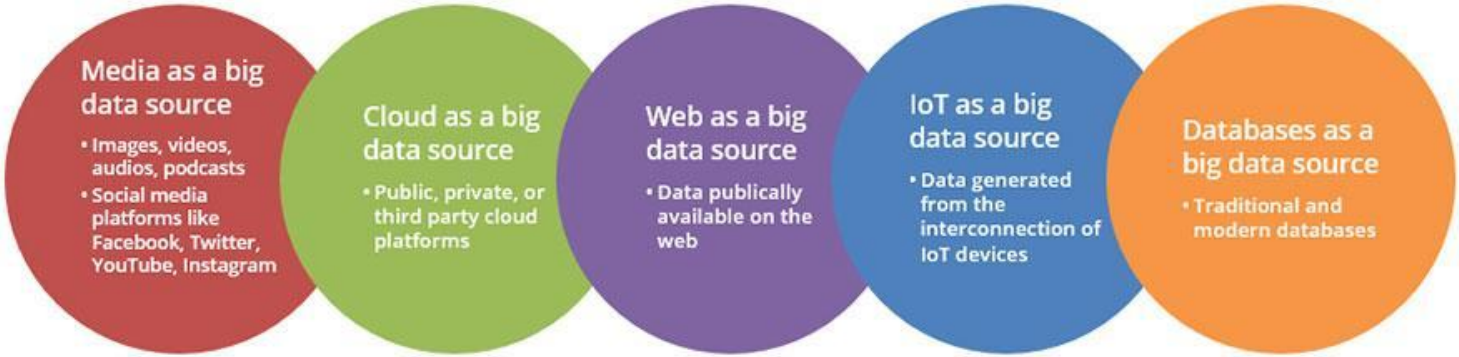
- Value is the most important aspect in the big data.
- Though, the potential value of the big data is huge.
- It is all well and good having access to big data but unless we can turn it into value it is become useless.
- It becomes very costly to implement IT infrastructure systems to store big data, and businesses are going to require a return on investment.

VARIETY

- Big data is not always structured data and it is not always easy to put big data into a relational database.
- This means that the category to which Big Data belongs to is also a very essential fact that needs to be known by the data analysis.
- Dealing with a variety of structured and unstructured data greatly increases the complexity of both storing and analyzing Big Data.
- 90% of data generated is data is in unstructured form.

PRIMARY SOURCE OF BIG DATA

- Primary sources of Big Data are;



MEDIA AS A BIG DATA SOURCE

- Media is the most popular source of big data, as it provides valuable insights on consumer preferences and changing trends.
- Since it is self-broadcasted and crosses all physical and demographical barriers, it is the fastest way for businesses to get an in-depth overview of their target audience, draw patterns and conclusions, and enhance their decision-making.
- Media includes social media and interactive platforms, like Google, Facebook, Twitter, YouTube, Instagram, as well as generic media like images, videos, audios, and podcasts that provide quantitative and qualitative insights on every aspect of user interaction.

CLOUD AS A BIG DATA SOURCE

- Today, companies have moved ahead of traditional data sources by shifting their data on the cloud.
- Cloud storage accommodates structured and unstructured data and provides business with real-time information and on-demand insights.
- The main attribute of cloud computing is its flexibility and scalability.
- As big data can be stored and sourced on public or private clouds, via networks and servers, cloud makes for an efficient and economical data source.

WEB AS A BIG DATA SOURCE

- The public web constitutes big data that is widespread and easily accessible.
- Data on the Web or ‘Internet’ is commonly available to individuals and companies alike.
- Moreover, web services such as Wikipedia provide free and quick informational insights to everyone.
- The enormity of the Web ensures for its diverse usability and is especially beneficial to start-ups and SME’s, as they don’t have to wait to develop their own big data infrastructure and repositories before they can leverage big data.

IOT AS A BIG DATA SOURCE

- Machine-generated content or data created from IoT constitute a valuable source of big data.
- This data is usually generated from the sensors that are connected to electronic devices.
- The sourcing capacity depends on the ability of the sensors to provide real-time accurate information.
- IOT is now gaining momentum and includes big data generated, not only from computers and smartphones, but also possibly from every device that can emit data.
- With IoT, data can now be sourced from medical devices, vehicular processes, video games, meters, cameras, household appliances, and the like.

DATABASES AS A BIG DATA SOURCE

- Businesses today prefer to use an incorporation of traditional and modern databases to acquire relevant big data.
- This integration paves the way for a hybrid data model and requires low investment and IT infrastructure costs.
- Furthermore, these databases are deployed for several business intelligence purposes as well.
- These databases can then provide for the extraction of insights that are used to drive business profits.
- Popular databases include a variety of data sources, such as MS Access, DB2, Oracle, SQL, and Amazon Simple, among others.

BIG DATA TOOLS AND SOFTWARE

- Hadoop
- Apache Spark
- MongoDB
- TensorFlow
- Cassandra
- Kaggle
- CouchDB
- Apache Kafka

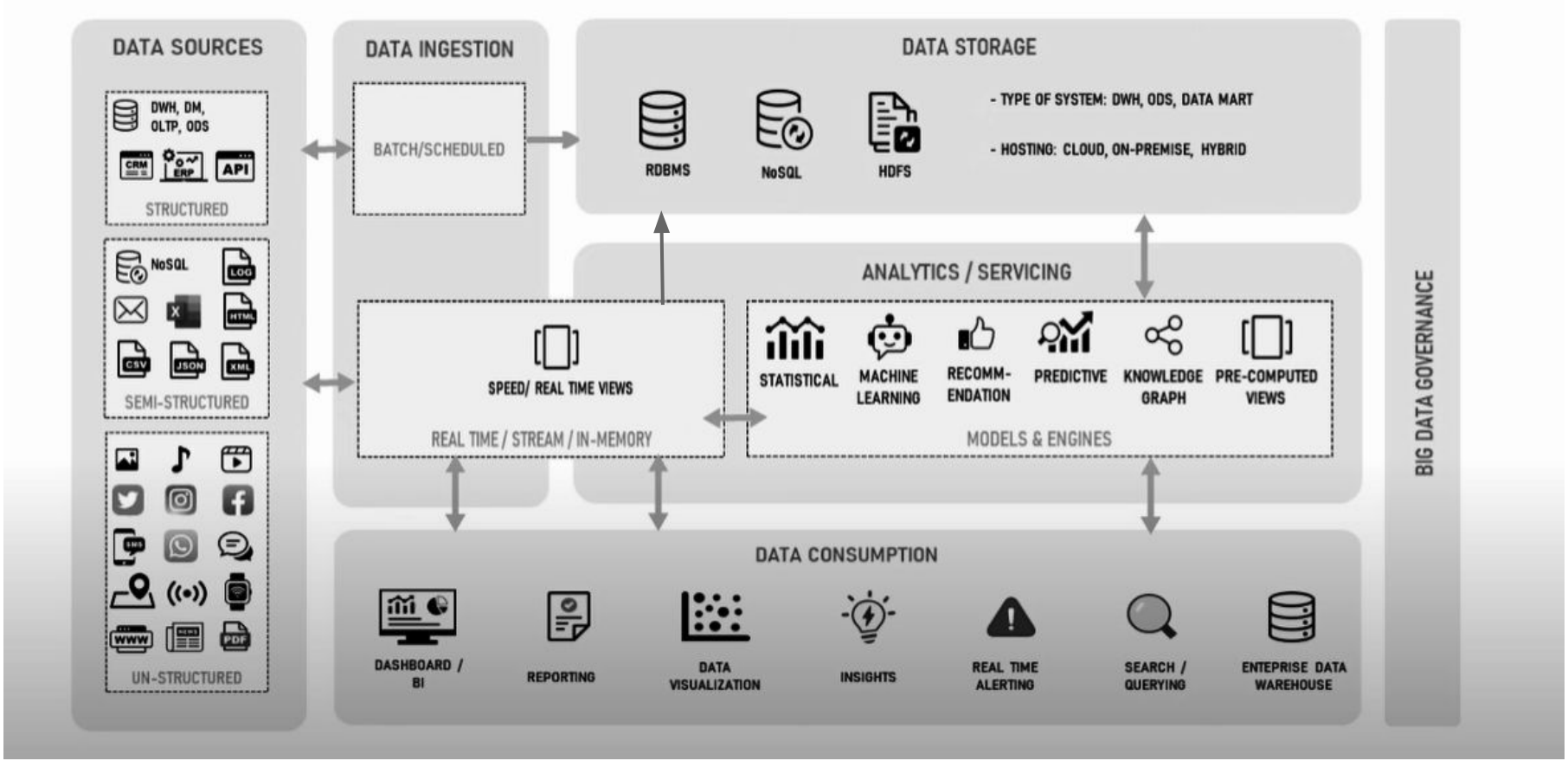
BIG DATA MINING

- Big data mining is referred to the collective data mining or extraction techniques that are performed on large sets /volume of data or the big data.
- Big data mining is primarily done to extract and retrieve desired information or pattern from humongous quantity of data.
- Big data mining works on data searching, refinement , extraction and comparison algorithms.

TOP TRENDS IN BIG DATA

- More data, increased data diversity drive advances in processing and the rise of edge computing.
- Big data storage needs spur innovations in cloud and hybrid cloud platforms, growth of data lakes.
- DataOps and data governance are becoming more prominent.
- Adoption of advanced analytics, machine learning and other AI technologies increases dramatically.
    -
    -

Big Data General Architecture



Database

Early Database Systems

Database	Advantages	Limitation
Flat File System	Simplest form of database,storing data in plain text files	Limited in handling complex relationships between data
Hierarchical Database System	Organized data in a tree-like structure with parent-child relationship	Changes in the structure were challenging to implement
Network Database System	Modeled data using a graph structure, allowing many to many relationship	Complexity in designing and maintaining the database schema
Relational Database System	Organized data into tables with rows and columns	Faces scalability challenges in big data with high traffic app.

Relational Database

- A Relational Database is a collection of data items which are organized in the form of tables of information, which can be easily accessed.
- This concept was introduced by E.F.Codd a researcher at IBM in 1970 .
- In Relational Databases the data is stored using rows and columns in the form of a table.

Features of a Relational Database

- **Structured Format:** Data is stored in tables, where each table represents a specific entity (e.g., customers, orders).
- **Relationships:** Tables can be linked (related) through foreign keys, allowing efficient data organization and retrieval.
- **Data Integrity**: Ensures data accuracy and consistency through constraints like primary keys, foreign keys, unique constraints, and more.
- **Query Languag**e: Relational databases use Structured Query Language (SQL) for managing and manipulating data.
- **ACID Properties**: Ensures reliable transactions through Atomicity, Consistency, Isolation, and Durability.

Advantages of Relational Database

- **Flexibility**: Easy to update, insert, and delete data.
- **Scalability**: Suitable for small-scale to large-scale applications.
- **Data Consistency**: Maintains data integrity through constraints.
- **Standardized Queries**: SQL provides a uniform way to interact with data.

Relational Database Advantage

But...

- ❑ Relational databases were not built for **distributed applications.**

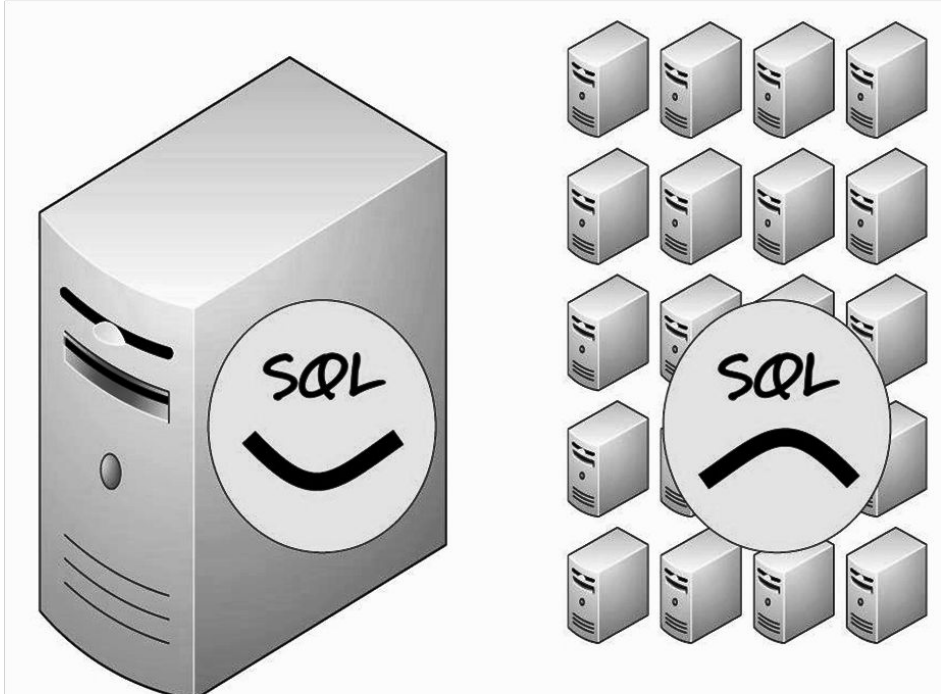
Because...

- ❑ Joins are expensive
- ❑ **Hard to scale horizontally**
- ❑ Impedance mismatch occurs
- ❑ Expensive (product cost, hardware, Maintenance)

And....

It's weak in:

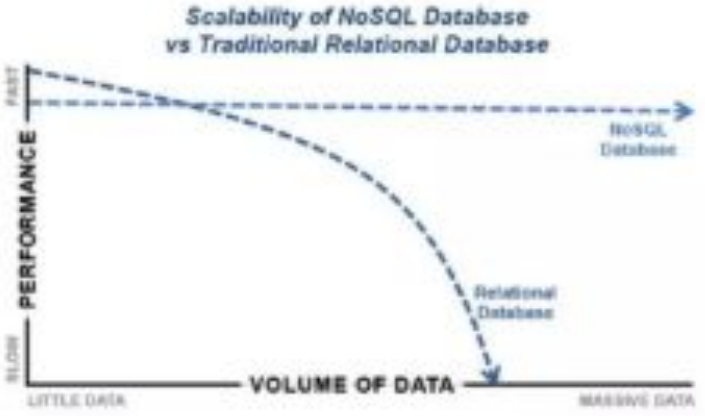
- ❑ Speed (performance)
- ❑ High availability
- ❑ Partition tolerance



Limitations

RDBMS Limitations

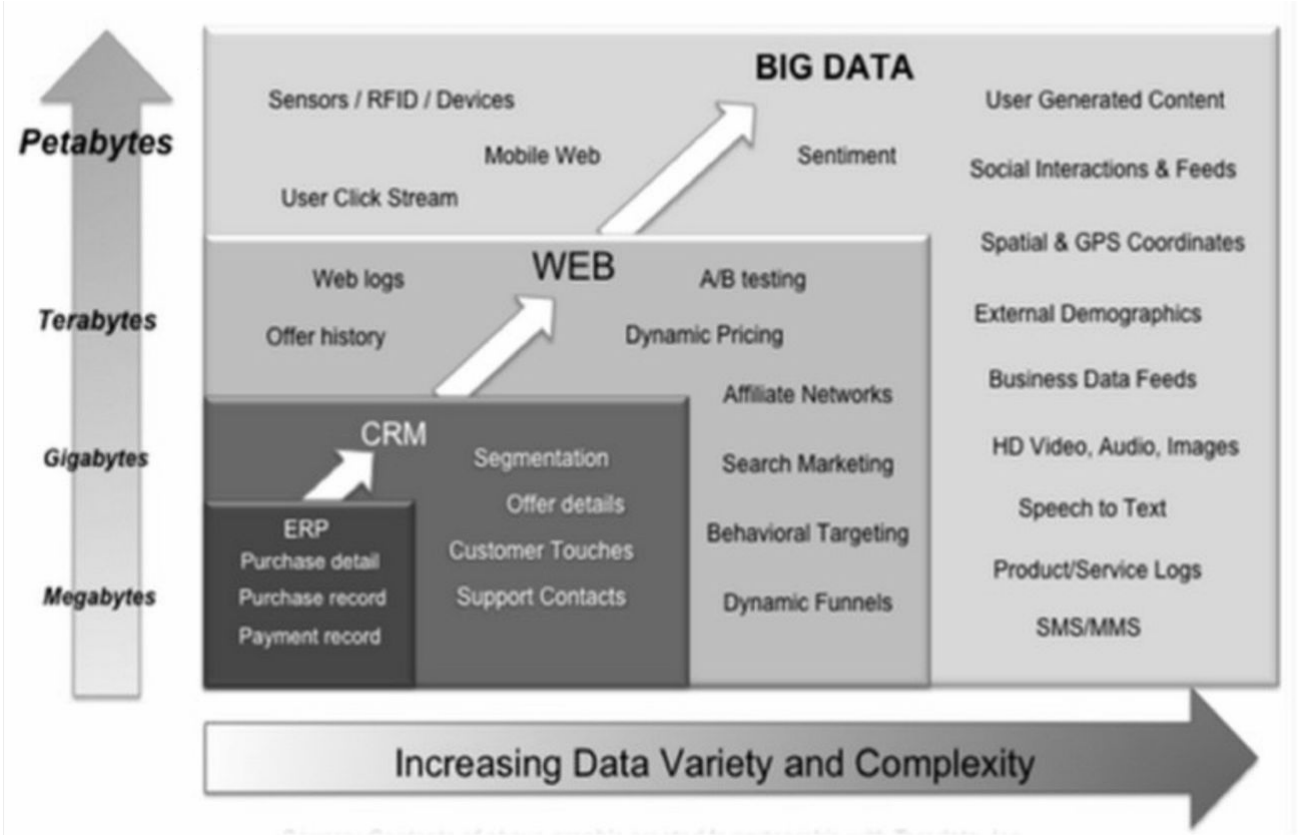
- Many issues while scaling up for massive datasets
- Not designed for distributed computing
- Expensive specialized hardware
- Multi-node databases considered as solutions - Known as 'scaling out' or 'horizontal scaling'
  - Master-slave
  - Sharding



Rise of Big Data

Three V(s) of Bigdata:

- Volume
- Velocity
- Variety



The Rise Of NoSQL Database

- NoSQL databases have emerged as a crucial component in the realm of modern data management.
- In today's world, characterized by massive data volumes and the demand for real-time applications, traditional relational databases face limitations in terms of scalability and flexibility.
- NoSQL databases address these challenges by providing a dynamic and scalable solution.

What is NoSQL Database ?

- A NoSQL database (Not Only SQL) is a type of database management system that provides a mechanism for storing, retrieving, and managing data that does not follow the traditional relational database model.
- Unlike relational databases, NoSQL databases are designed to handle unstructured, semi-structured, or structured data, providing greater flexibility and scalability for certain types of applications.

Characteristics of NoSQL Database

1) **Schema-less Design**

   NoSQL databases typically do not enforce a fixed schema. This allows developers to insert data without first defining its structure, making them more adaptable to evolving data models.

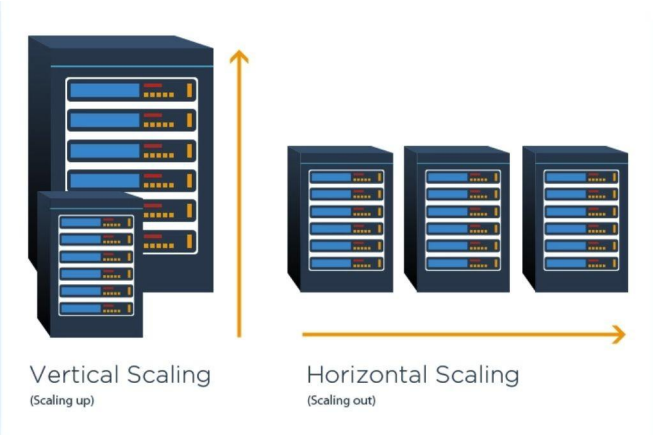
2) **Variety of Data Models**

   NoSQL databases support various data models, including document-oriented, key-value pairs, wide-column stores, and graph databases. This versatility allows developers to choose the best-fit data model for their specific application needs.

Characteristics of NoSQL Database

3) Horizontal Scalability

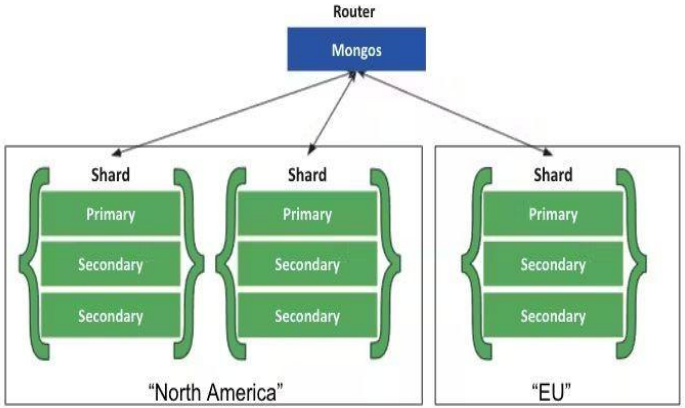
NoSQL databases are designed to scale horizontally, meaning they can handle increased data volumes and traffic by adding more servers or nodes to the database system. This is in contrast to the vertical scaling often associated with traditional relational databases.



Characteristics of NoSQL Database

4) Distributed Architecture

Many NoSQL databases are built to operate in distributed environments, allowing them to distribute data across multiple nodes or servers. This design enhances fault tolerance and ensures data availability.



Characteristics of NoSQL Database

5) **Dynamic and Unstructured Data**

NoSQL databases are well-suited for managing unstructured or semi-structured data, such as JSON or XML documents. This makes them versatile in handling diverse data formats.

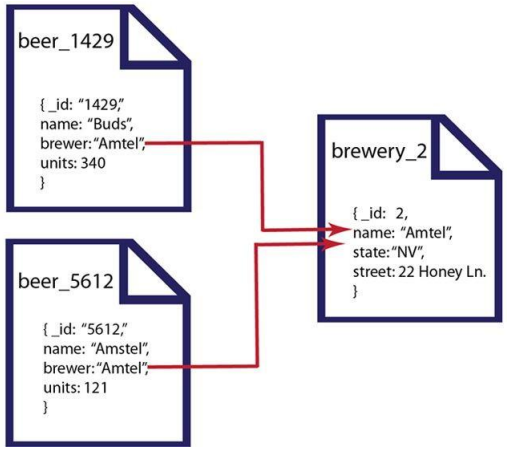
Categories of NoSQL Database

NoSQL databases can be broadly categorized into four main types, each with its own unique data model and characteristics. These categories are based on the way data is organized and stored within the database. The four main categories of NoSQL databases are:

1) **Document-oriented database**
2) **Key-Value stores**
3) **Column-oriented database**
4) **Graph-based database**

Document-oriented Database

- Data is stored as documents, typically in JSON or BSON format.
- Each document is a self-contained unit that may contain nested structures.
- Documents are often organized in collections.
- Examples : MongoDB, CouchDB, Elasticsearch

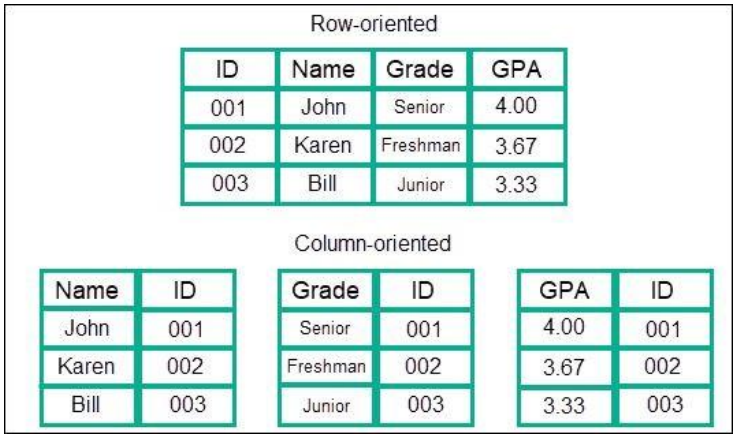


Key-value Stores

- Basic data model with a collection of key-value pairs.
- Data is stored as unstructured values or blobs associated with a unique key.
- Simple and fast retrieval of values based on keys.
- Examples: Redis, Amazon DynamoDB, Riak

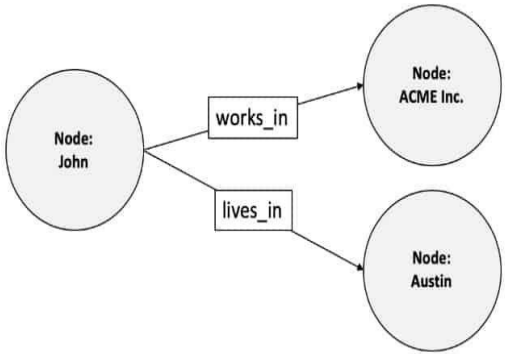
Column-oriented Database

- Data is organized into columns instead of rows.
- Columns are grouped into column families, and each column family can have a different set of columns.
- Well-suited for read and write intensive workloads.
- Example: Apache Cassandra, HBase , Amazon SimpleDB

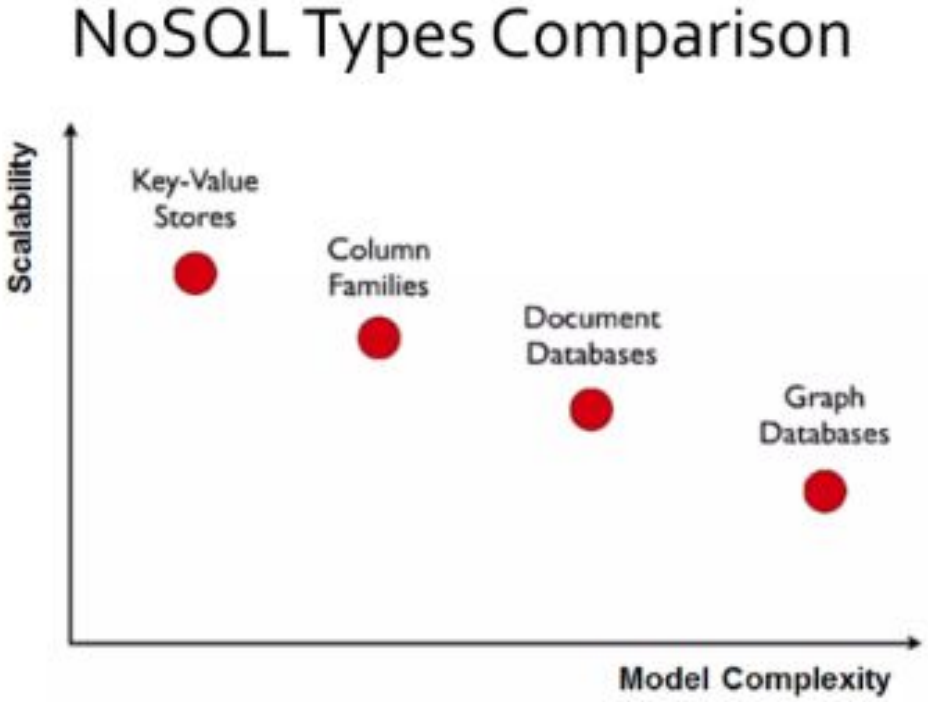


Graph-based database

- Designed for handling relationships between data points.
- Data is represented as nodes, edges, and properties.
- Efficient for traversing and querying complex networks or graphs.
- Examples: Neo4j, ArangoDB, Amazon Neptune



Comparison


NoSQL Types Comparison
Scalability
Key-Value Stores
Column Families
Document Databases
Graph Databases
Model Complexity

Advantages of NoSQL Database

- **Schema Flexibility**
  - NoSQL databases allow for a flexible and dynamic schema, accommodating changes to the data model without requiring predefined structures. This flexibility is beneficial in scenarios with evolving or dynamic data requirements.
- **Scalability**
  - NoSQL databases are designed for horizontal scalability, making it easier to handle large volumes of data and increased traffic by adding more nodes.

  Traditional SQL databases may face challenges in scaling horizontally.

Advantages of NoSQL Database

- **Optimized for Big Data**
  - NoSQL databases are often designed to handle large volumes of data efficiently, making them suitable for big data scenarios and real-time analytics.
- **Developer Friendly**
  - NoSQL databases often provide a more developer-friendly environment, allowing for quicker development cycles. They may have less rigid data requirements, making it easier to adapt to changing application needs.

Advantages of NoSQL Database

- **Cloud Computing**
    - NoSQL databases are often well-suited for cloud environments, aligning with distributed and scalable architectures commonly used in cloud computing.

Challenges and Considerations

- **Consistency and ACID Compliance Trade-offs**
  - NoSQL databases may sacrifice strict consistency for better scalability.Developers need to consider the trade-offs in terms of data consistency and ACID compliance.
- **Learning Curve**
  - Developers accustomed to RDBMS may face a learning curve when transitioning to NoSQL databases.Training and resources may be required for a smooth transition.

Challenges and Considerations

- **Data Migration Challenges**

  Migrating from traditional databases to NoSQL can pose challenges.Planning and executing data migration strategies are crucial for a successful transition.

NoSQL Popularity

NoSQL Summary

- NoSQL databases offer unparalleled advantages in terms of scalability, flexibility, and performance.
- Ideal for modern applications with dynamic data requirements and high scalability needs.
- NoSQL databases represent a paradigm shift in the way we approach and manage data.

Data Cleaning and Visualization

- Data Cleaning:
  - This is a critical preprocessing step where errors, inconsistencies, and inaccuracies in the dataset are identified and corrected or removed to ensure the data is suitable for analysis.
- Data Visualization:
  - It involves the representation of data in a graphical format, which allows for easier interpretation of trends, patterns, and insights.
- Tools Used:
- Pandas:
  - A Python library that provides data structures and data analysis tools.
- matplotlib:
  - A plotting library for creating static, interactive, and animated visualizations in Python.

Data Cleaning Overview

- Data cleaning involves removing or correcting inaccurate, incomplete or irrelevant data within a dataset.
- Clean data is essential for making reliable and accurate inferences. It ensures that the results of any data analysis are not skewed by flawed data.
- **Common Data Issues:**
    - Missing Values: Data entries that are not recorded.
    - Duplicates: Multiple identical rows or entries that can bias the analysis.
    - Incorrect Data Types: Mismatched data types (e.g., numeric data stored as text).
    - Inconsistent Formats: Variations in data representation (e.g., date formats).

.

Data Cleaning Overview

- Data cleaning involves removing or correcting inaccurate, incomplete or irrelevant data within a dataset.
- Clean data is essential for making reliable and accurate inferences. It ensures that the results of any data analysis are not skewed by flawed data.
- **Common Data Issues:**
    - Missing Values: Data entries that are not recorded.
    - Duplicates: Multiple identical rows or entries that can bias the analysis.
    - Incorrect Data Types: Mismatched data types (e.g., numeric data stored as text).
    - Inconsistent Formats: Variations in data representation (e.g., date formats).

.

Data Cleaning Overview

Student ID	Name	Age	Gender	Grade	Score	Address
1	Sita	20	F	A	90	Kathmandu
2	Shyam	22	M	B	80	Lalitpur
3	Hari	21	M	C	70	Bhaktapur
4	Sita	20	F	A	90	Kathmandu
5	Rita	23	F	B	85	Pokhara
6	Ram	22	M	NULL	NULL	NULL

Data Cleaning Overview

- A dataset containing student information:
  - Student ID: A unique number assigned to each student.
  - Name: Full name of the student.
  - Age: Age of the student, expected as an integer.
  - Gender: Gender of the student, often as 'M' or 'F'.
  - Grade: Academic grade, such as 'A', 'B', 'C'.
  - Score: Numeric test scores.
- Potential Issues:
  - Missing values in the Score column.
  - Duplicates in student records.
  - Age or Score stored as strings instead of integers or floats.

Data Cleaning Process

- Remove Duplicates:
  - Use df.drop_duplicates() to remove any repeated rows in the dataset.
- Handle Missing Values:
  - Use df['Score'].fillna(df['Score'].mean()) to replace missing scores with the mean of the available scores.
- Correct Data Types:
  - Convert columns to appropriate data types using pd.to_numeric() or df['column'].astype(type).
- Standardize Formats:
  - Ensure columns like Gender are consistent (e.g., 'Male'/'Female' vs. 'M'/'F').

.

Data Visualization

- Data visualization is the process of translating information into a visual context, such as a graph or map, to make data easier to understand.
- Simplifies data interpretation by converting raw data into visual elements.
- Identifies trends, patterns, and outliers effectively.
- Enhances communication of insights from data analysis.
- Visualization Tools:
    - While matplotlib is used here, other libraries like seaborn can also be used for more advanced visualizations.

.

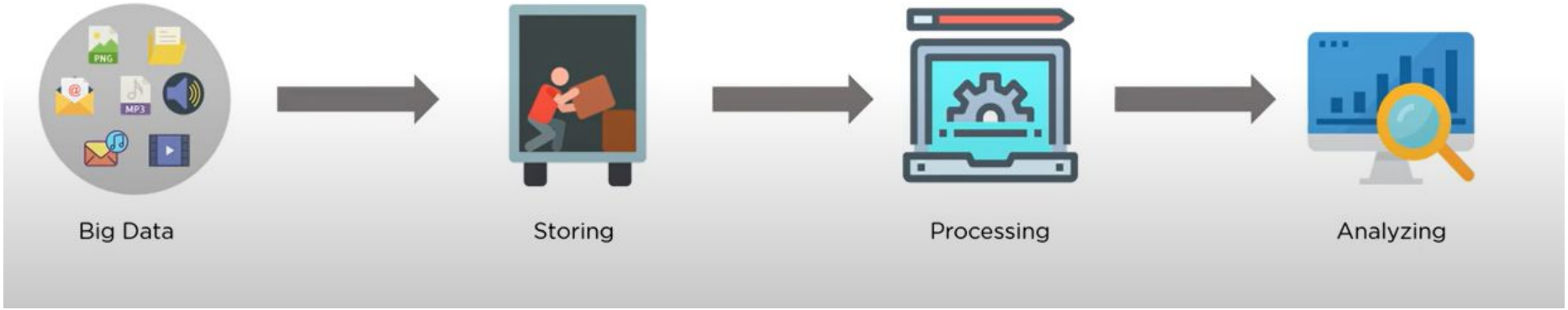
Visualization type

- Bar Plot:
  - Useful for comparing different categories or groups.
- Pie Chart:
  - Shows proportions of a whole, helpful in understanding percentage distribution.
- Histogram:
  - Represents the distribution of numerical data by showing the frequency of data points in successive intervals.
- Scatter Plot:
  - Displays values for typically two variables for a set of data and helps identify correlations.

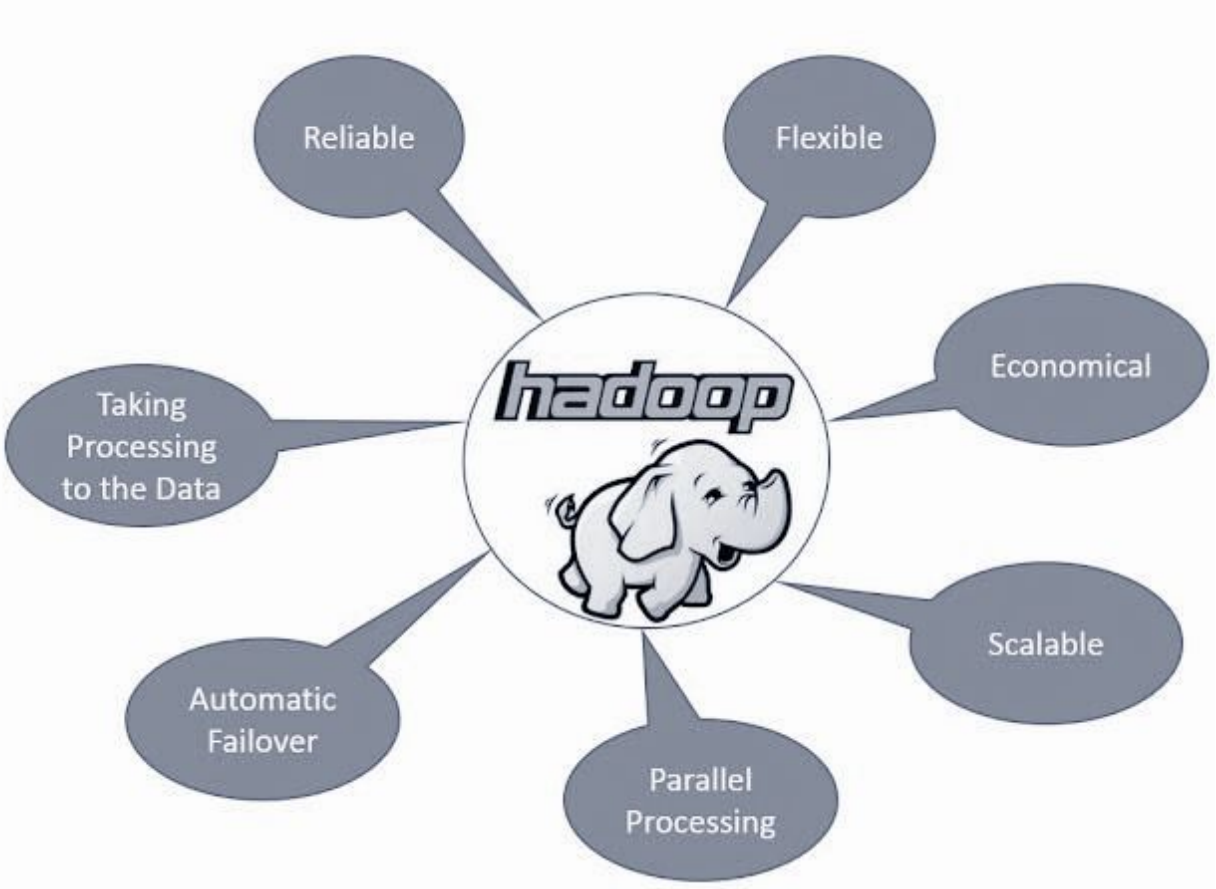
.

Introduction to Hadoop

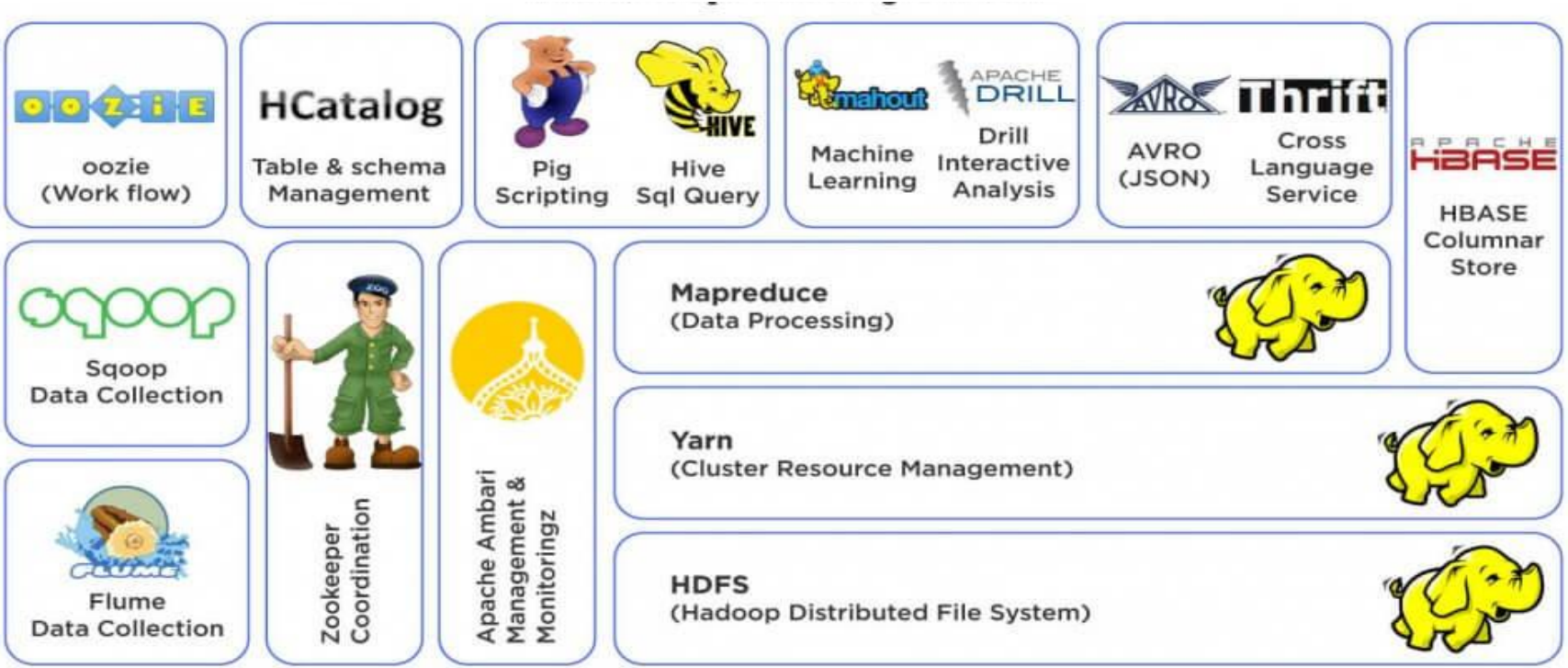
- Hadoop is a framework that manages big data storage in a distributed way and processes it in parallel.
- Hadoop allows for the distribution of datasets across a cluster of commodity hardware. Processing is performed in parallel on multiple servers simultaneously.



Features of Hadoop



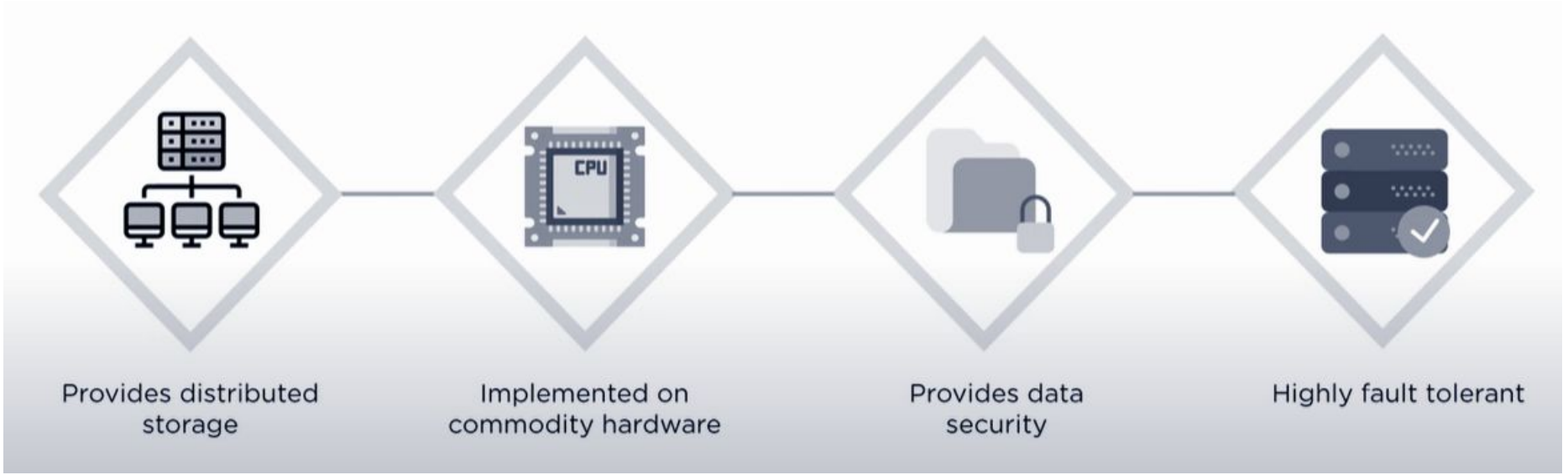
Hadoop Ecosystem



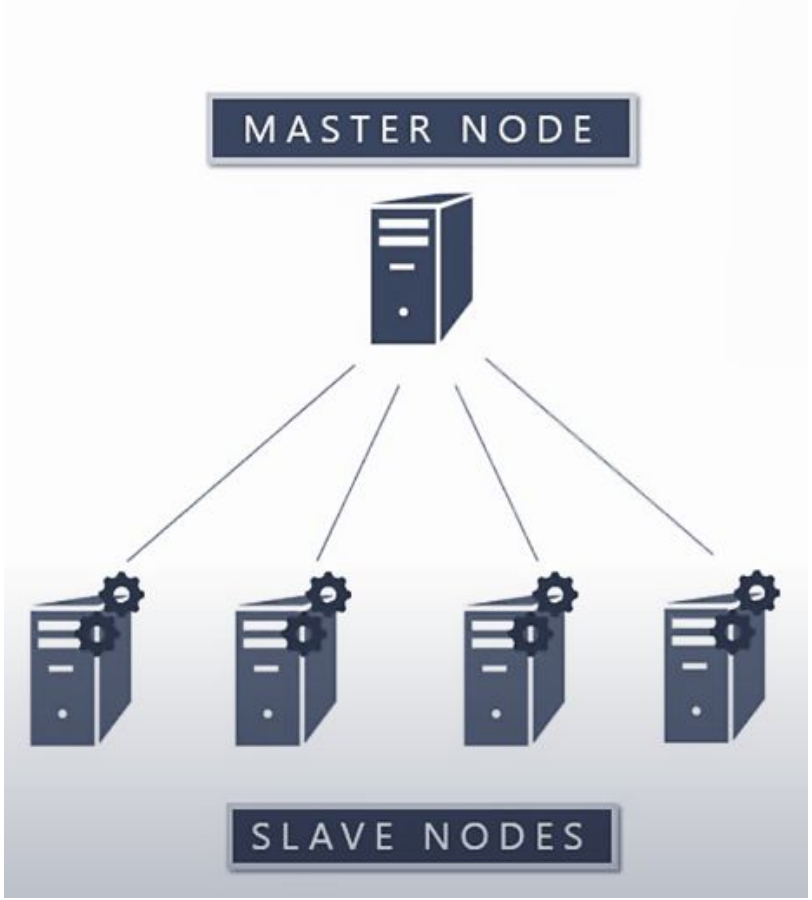
Hadoop Distributed File System (HDFS)

- HDFS is a distributed file system designed to store large data across clusters of commodity hardware.
- It enables the reliable and scalable storage of large datasets across a cluster of machines by dividing files into large blocks and distributing them across multiple nodes
- It provides high-throughput access to data and is optimized for batch processing rather than real-time access.
- It employs a master/slave architecture with a NameNode (master) and multiple DataNodes (slaves).

Features of HDFS



HDFS Architecture



HDFS Architecture

NameNode [Master Node]

- NameNode is a daemon that maintains and operates all DataNodes (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.
- The NameNode frequently receives heartbeats and block reports from the data nodes in the cluster to ensure they are working and live.

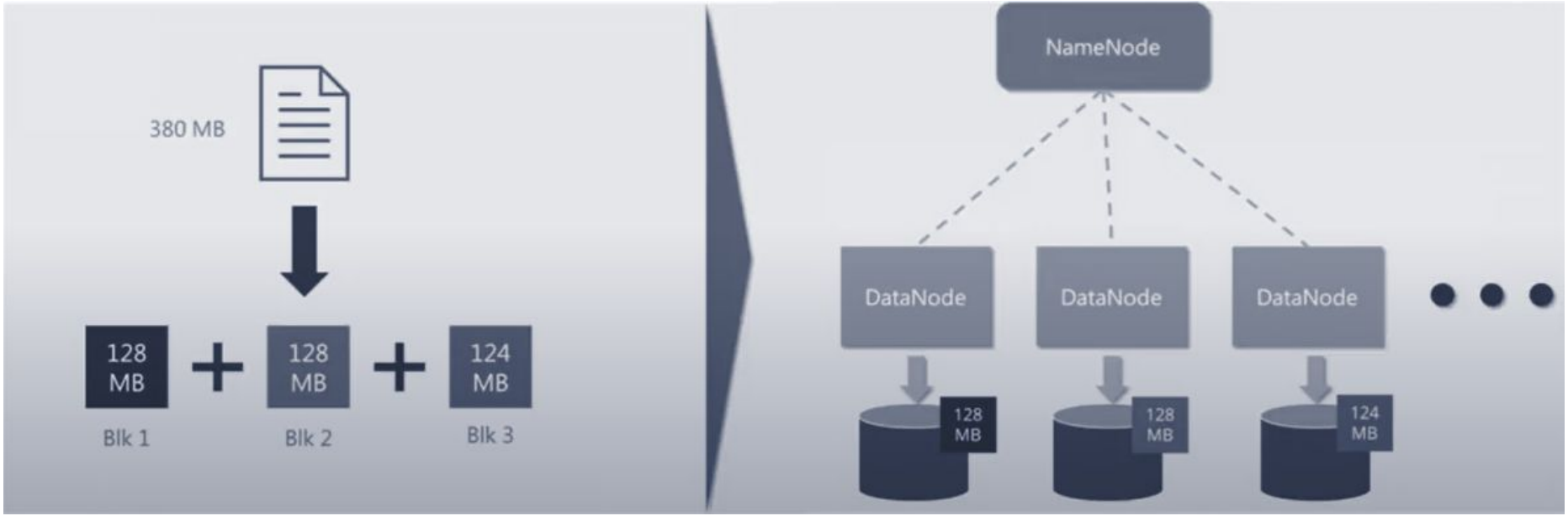
HDFS Architecture

DataNodes [Slave Nodes]

- 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 requests 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 every 3 seconds.

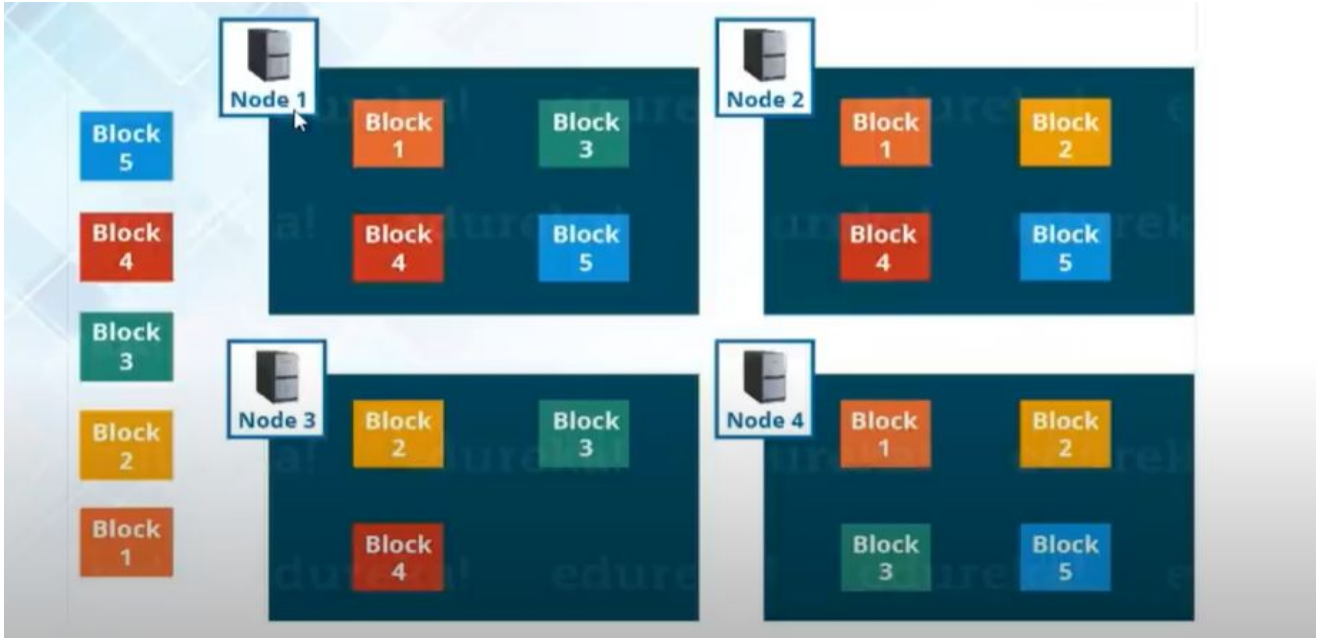
HDFS Data Block

- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB



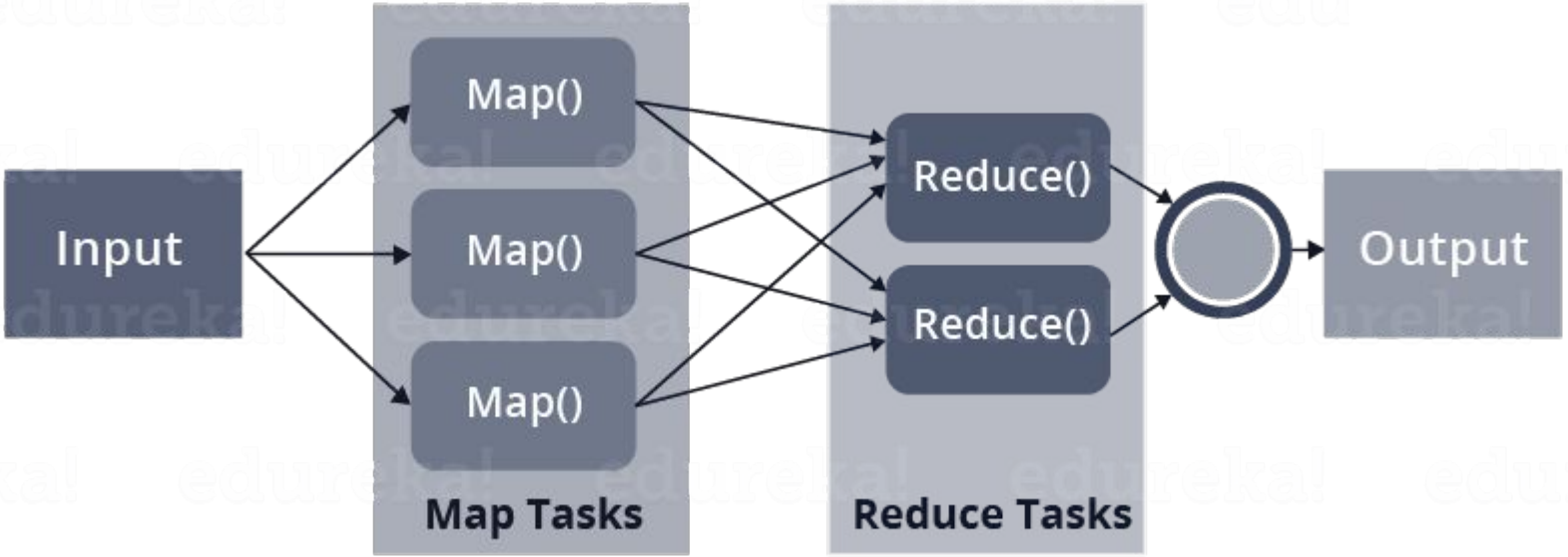
Block Replication

- In HDFS (Hadoop Distributed File System), block replication is a key feature for fault tolerance and data reliability.
- When a file is stored in HDFS, it is split into blocks (default block size = 128 MB).
- Each block is replicated to multiple DataNodes where default replication factor is 3

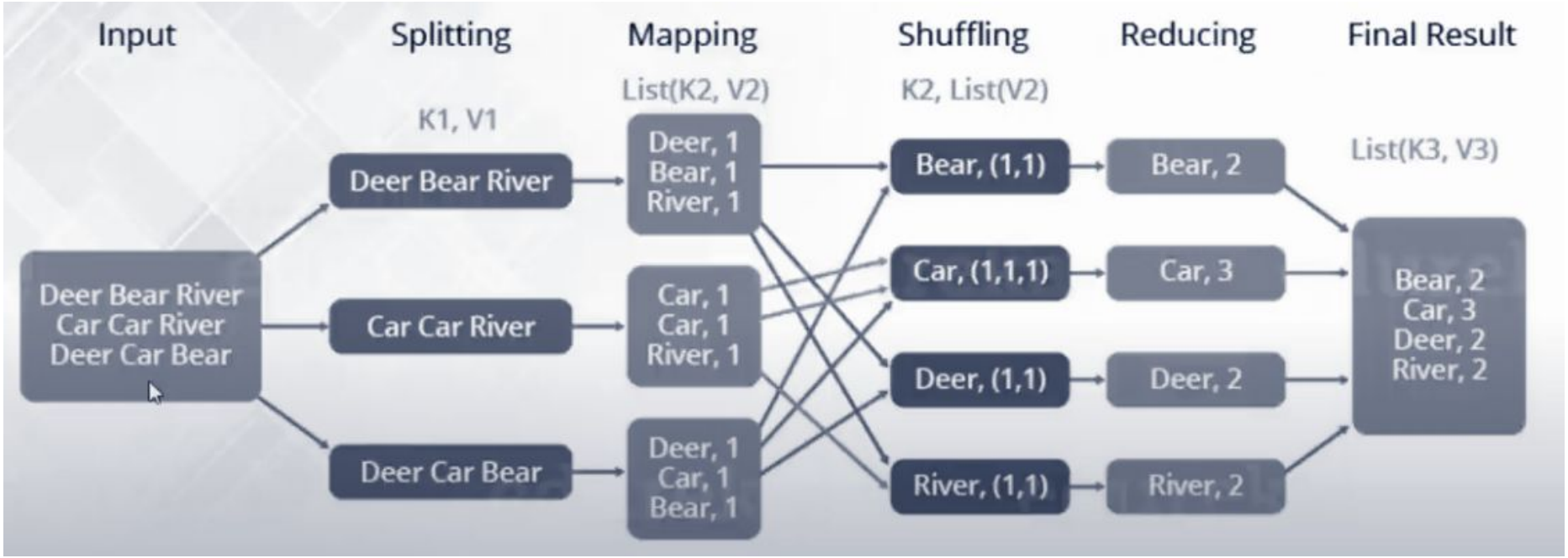


MapReduce

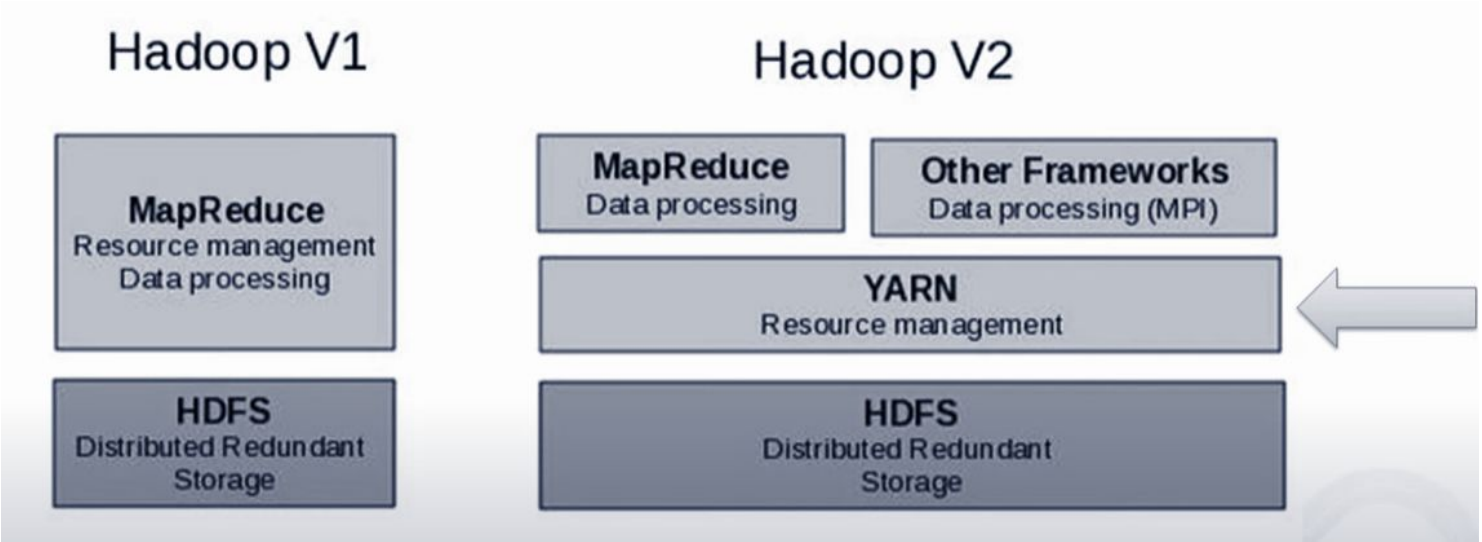
- MapReduce is a programming model and processing component used for handling and analyzing large data sets in a distributed computing environment.
- It processes data in parallel in a distributed system and handles massive datasets across multiple nodes.
- It provides fault tolerance and scalability and simplifies complex distributed processing.



MapReduce Process—with Word count example



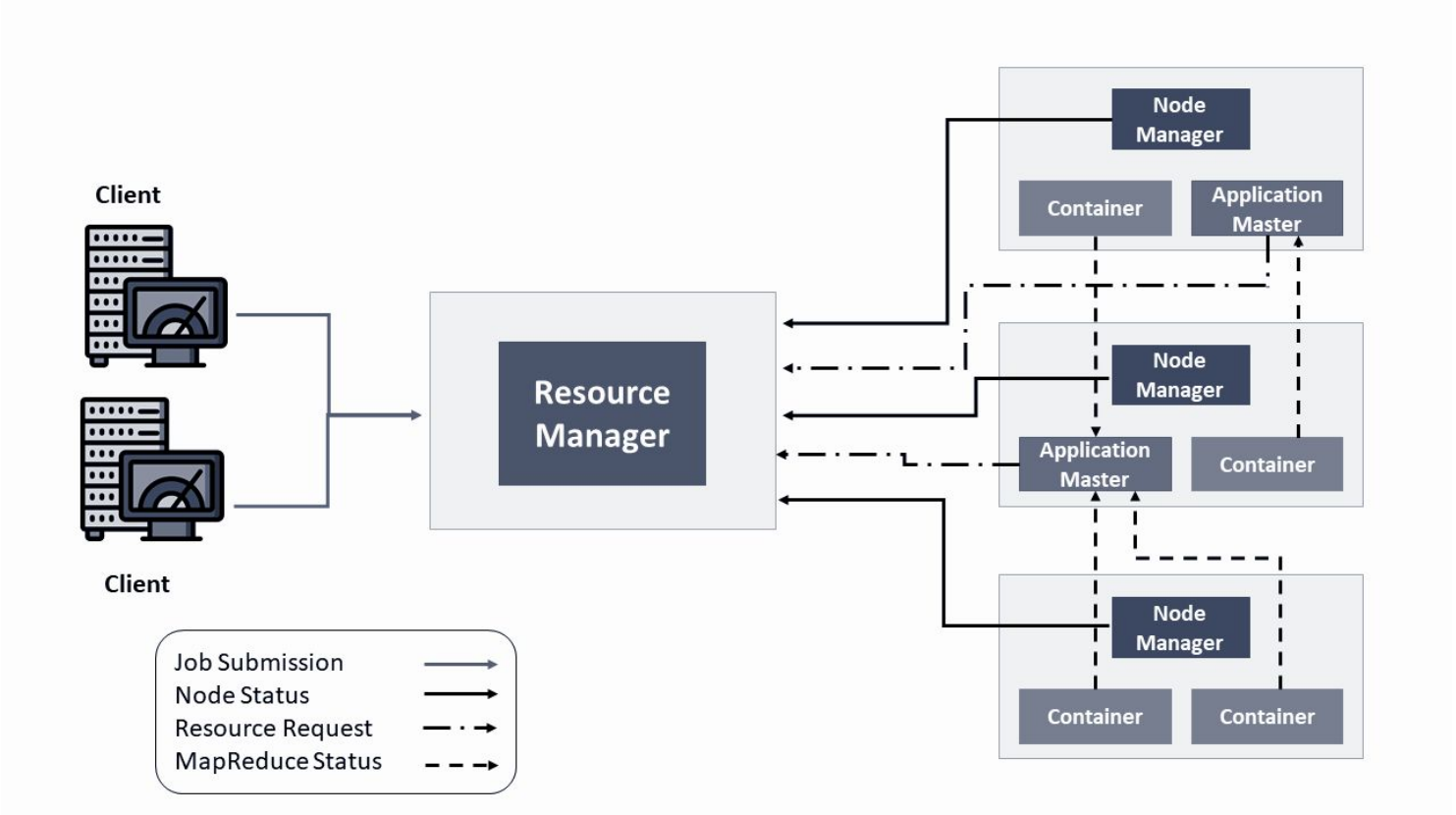
YARN [Yet Another Resource Negotiator] Overview



YARN

- YARN (Yet Another Resource Negotiator) is the resource management and job scheduling component of the Hadoop ecosystem
- It is the Cluster Resource Management Layer of Hadoop which separates resource management from data processing.
- It supports various processing engines beyond MapReduce, including Spark, Tez, Hive, Flink,
- It allocates CPU and memory to applications and balances load across the cluster.
- It enables multiple frameworks to run simultaneously, allowing for better resource utilisation.
- It decouples resource management from MapReduce, making Hadoop a general-purpose distributed computing platform.
- It enables fault-tolerant and parallel processing.

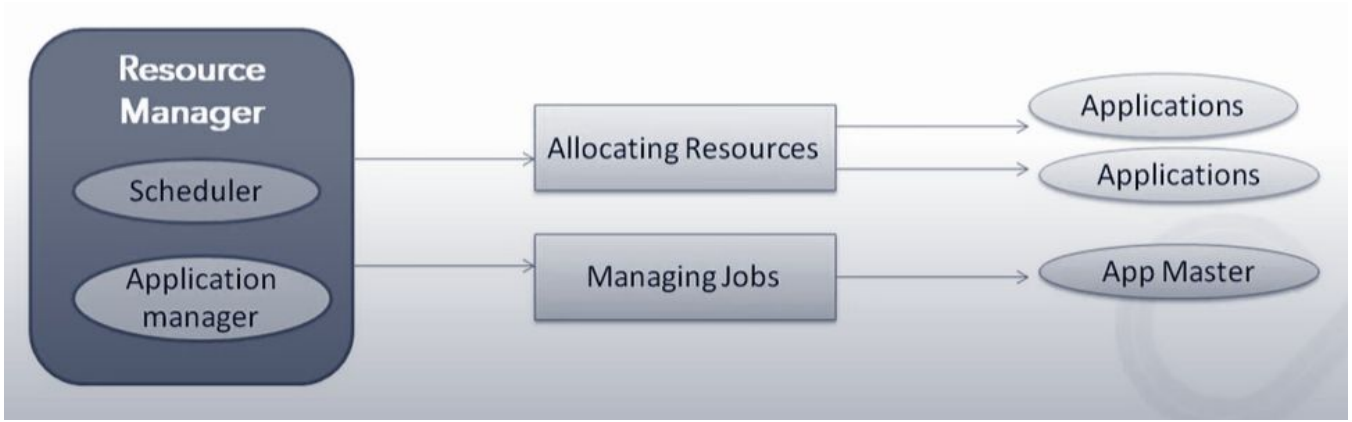
YARN Architecture



YARN Components

ResourceManager (RM)

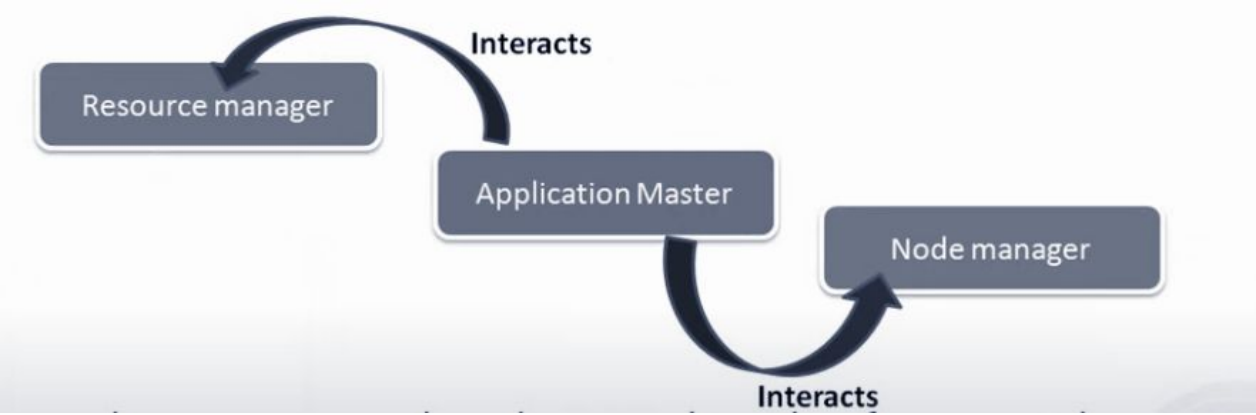
- The master daemon that manages and allocates all cluster resources.
- It keeps track of available resources across all nodes and allocates them to applications.
- It's a global resource scheduler
- RM consists of two components
    - **Scheduler:** Allocates resources based on policies like capacity or fairness.
    - **ApplicationManager:** Manages application submission and monitors their status.



YARN Components

ApplicationMaster (AM)

- It manages application life cycle and task scheduling.
- One per application (job) running in the cluster.
- Coordinates the execution of tasks for the application.
- Requests resources from the ResourceManager and works with NodeManagers to start/stop cc
- Tracks progress and handles failures within the application lifecycle.



YARN Components

NodeManager (NM)

- It manages single-node resource allocations.
- Runs on each worker node in the cluster.
- Responsible for managing resources (CPU, memory, disk, network) on its node.
- Launches and monitors containers as instructed by the ResourceManager.
- Reports the health status and resource usage of the node back to the ResourceManager.

YARN Components

Container

- A container is a bundle of resources (CPU, memory) allocated by the ResourceManager to run a task.
- The NodeManager launches the actual process inside the container on the node.
- Containers provide resource isolation for running applications.

Apache Spark in Hadoop

- Apache Spark is an open-source, distributed processing engine designed for Big Data analytics.
- It is designed to process large-scale data much faster than traditional MapReduce.
- It uses in-memory computing, which reduces the time-consuming disk read/write operations.
- It supports both **batch** and **real-time** stream processing.
- It supports a wide range of workloads in a single platform:
    - Batch processing (e.g., ETL jobs)
    - Streaming processing (real-time logs, sensor data)
    - Machine learning (via MLlib)
    - Graph processing (via GraphX)
    - SQL queries (via Spark SQL)
- Integrates well with tools like Kafka, Hive, and HBase
- Runs on Hadoop YARN and supports HDFS
- Uses RDDs with lineage information to recover lost data automatically.