Presentation
Design

HADOOP FRAMEWORK

Module 4

Apache Hadoop

Definition

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

Purpose

Hadoop is designed to handle massive amounts of data across clusters of commodity hardware, making it suitable for big data processing.

Origins

It was originally created by Doug Cutting and Mike Cafarella in 2005 and is now maintained by the Apache Software Foundation

Assumptions

All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

More about Hadoop

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume.

Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing. It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more.

The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Why Hadoop?

01. Flexibility

Hadoop manages both structured and unstructured data efficiently, supporting various data types and formats. It is language-agnostic and runs on multiple operating systems.

02. Scalability

Hadoop allows easy scalability by adding nodes without altering data formats or existing applications. It operates on commodity hardware and is fault-resistant.

03. Efficient Data Economy

Hadoop revolutionizes big data mining, acting as a reservoir for vast data inflows. It transforms the economics of data storage and analysis.

04. Robust Ecosystem

Hadoop provides a versatile ecosystem with initiatives like MapReduce, Hive, HBase, and more, catering to diverse computational needs.

05. Real-Time Capabilities

Hadoop adapts to real-time data processing, offering standardized approaches for various big data analytics APIs.

06. Cost-Effectiveness

Hadoop's cost-effective data analysis involves massively parallel processing on commodity servers, significantly reducing the cost per terabyte of storage.

History of Hadoop

2002

Doug Cutting and Mike Cafarella begin working on the Apache Nutch project, an open-source web crawler, dealing with substantial amounts of data.

2003

Google introduces the Google File System (GFS), a proprietary distributed file system designed for efficient data access.

2004

Google releases a white paper on MapReduce, a technique simplifying data processing on large clusters.

2005

Doug Cutting and Mike Cafarella introduce the Nutch Distributed File System (NDFS), incorporating MapReduce.

2006

Doug Cutting leaves Google, joins Yahoo, and introduces the Hadoop project based on Nutch, with the Hadoop Distributed File System (HDFS). Hadoop version 0.1.0 is released.

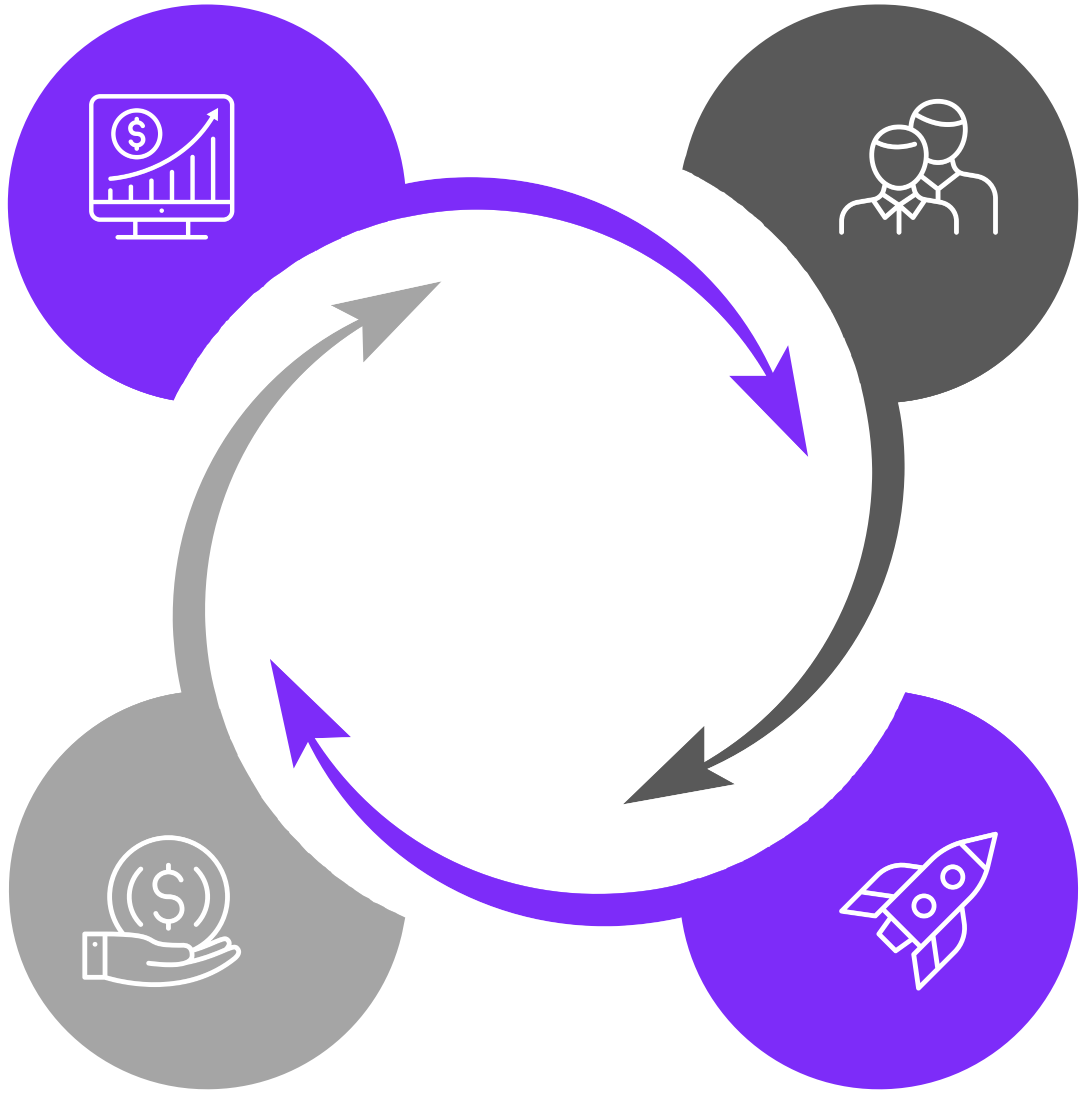
2007

Yahoo operates two clusters of 1000 machines running Hadoop.

History of Hadoop

2008

Hadoop becomes the fastest system to sort 1 terabyte of data on a 900-node cluster within 209 seconds.



2017

Hadoop 3.0 is released, reflecting advancements and updates in the Hadoop ecosystem.

2013

Hadoop 2.2 is released, showcasing ongoing development.

The journey of Hadoop, originating from the challenges faced during the Apache Nutch project, led to the creation of a powerful and scalable framework that revolutionized big data processing.

Components of HADOOP

Definition: Hadoop is a Big Data storage and management system that leverages distributed storage and parallel processing to handle vast amounts of data efficiently.

Hadoop Common:

Functionality: Common libraries and utilities shared across all Hadoop modules.

Role: Provides a foundation for other Hadoop modules to build upon.

Hadoop MapReduce:

Functionality: Processing unit in Hadoop.

Role: Executes parallel processing of data, dividing tasks into Map and Reduce phases.

Hadoop HDFS (Hadoop Distributed File System):

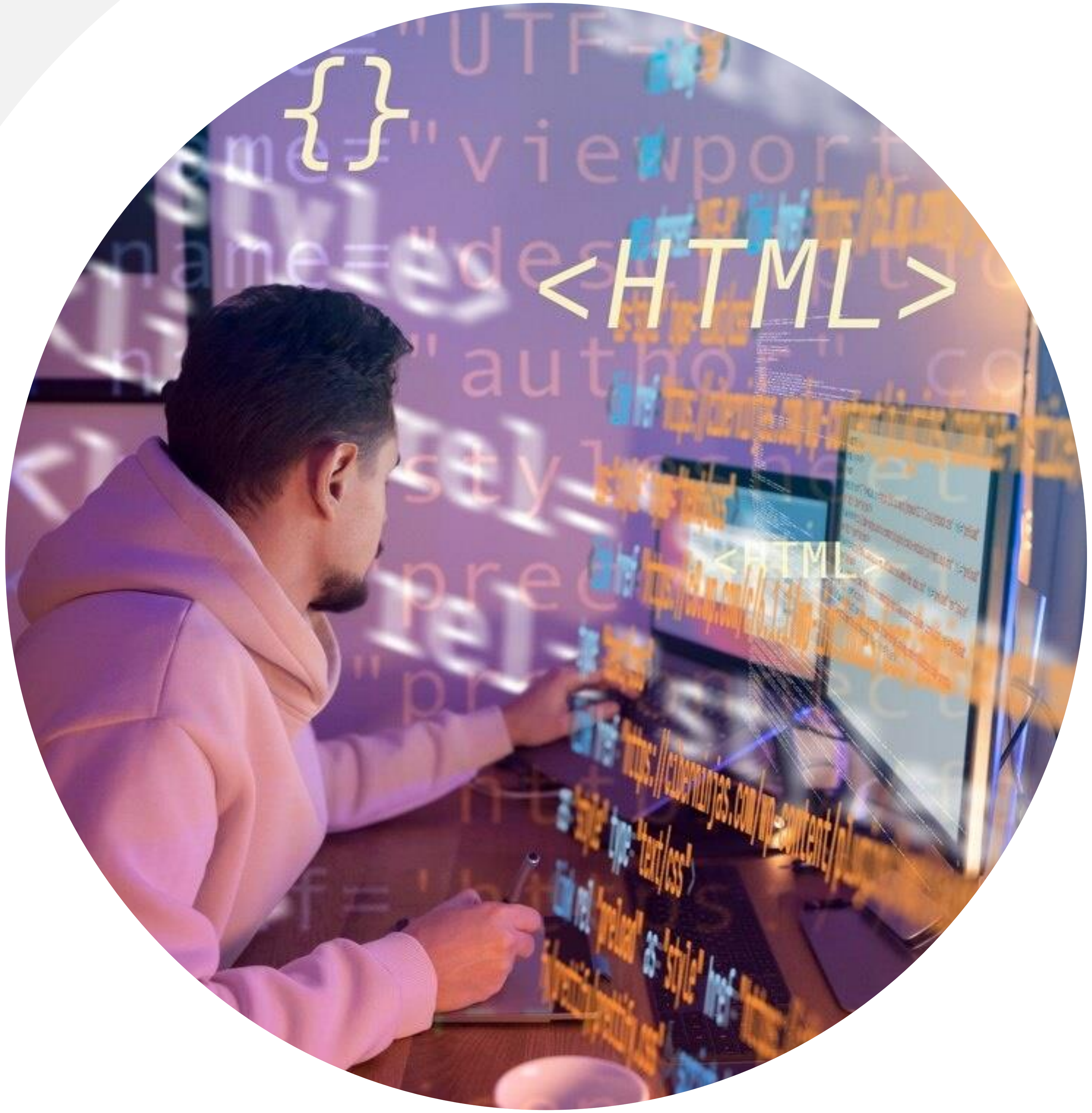
Functionality: Storage unit in Hadoop.

Role: Distributes and replicates data across multiple nodes for reliability and accessibility.

Hadoop YARN (Yet Another Resource Negotiator):

Functionality: Resource management unit in Hadoop.

Role: Manages and allocates resources for various applications running on the Hadoop cluster.



Hadoop Common

Hadoop Common, also known as Hadoop Core, is a module within the Hadoop ecosystem that provides shared utilities, libraries, and essential components required by other modules in the Hadoop framework.

Hadoop Common includes a **set of Java libraries** and scripts that serve as the foundation for the entire Hadoop ecosystem. These utilities are fundamental for the functioning of various components within the Hadoop cluster.

It maintains a **shared codebase** that encapsulates functionalities common to HDFS, YARN, MapReduce, and other Hadoop modules. This shared codebase ensures consistency and interoperability across different components.

One crucial role of Hadoop Common is to **address hardware failures within a Hadoop cluster**. Given that hardware failures are common in large-scale distributed systems, Hadoop Common provides mechanisms to handle and mitigate these failures automatically through software.

Hadoop HDFS (Hadoop Distributed File System)

HDFS is the default data storage system in Hadoop, serving as a repository for data before processing. It employs a distributed approach, dividing data into blocks and distributing them across the cluster.

Data in HDFS is organized into blocks, typically 128MB in size (configurable). These blocks are replicated for fault tolerance and distributed across the cluster, ensuring consistent and convenient access.

Key Components of HDFS

Name Node

Acts as the master node, controlling the storage system.

Data Node

Serves as a slave node, managing various structures in the Hadoop cluster.

Secondary Name Node

Assists the Namenode in its operations.

Master-Slave Architecture:

HDFS follows a Master-Slave architecture, where the Name node is the master controlling the storage, and Data Nodes are slaves managing the cluster's structures.

Cost-Effective Storage:

Designed for storing large datasets on commodity hardware, HDFS offers a cost-effective alternative to enterprise-level servers. Commodity devices can serve as Data Nodes, significantly reducing costs.

HDFS stores data in blocks, with a default size of 128MB (configurable). For example, a 400MB file would be divided into four blocks (128MB + 128MB + 128MB + 16MB). Replication ensures data availability, and the default Replication Factor is set to 3.

File Block in HDFS:

HDFS Features

Replication

HDFS employs replication for data availability. The Replication Factor, by default 3, determines the number of copies made for each file block. Adjusting the Replication Factor allows customization based on specific requirements.

Rack Awareness

Rack Awareness involves the physical grouping of nodes into racks within the Hadoop cluster. Large clusters consist of multiple racks. The Namenode utilizes rack information to choose the closest Data Node, optimizing read/write performance and reducing network traffic.

Why HDFS?

In essence, HDFS forms the backbone of Hadoop's storage capabilities, utilizing distributed and replicated blocks for efficient data handling and fault tolerance. Its cost-effective design and master-slave architecture contribute to the scalability and resilience of the Hadoop ecosystem.

HDFS Features

1. Distributed Storage:

HDFS offers distributed storage, meaning that data is divided into blocks and distributed across multiple nodes in a cluster. This approach allows for parallel processing and efficient storage utilization.

2. Implementation on Commodity Hardware

HDFS is designed to function on commodity hardware, providing a cost-effective solution for storing large datasets. It allows organizations to build their Hadoop clusters on standard, readily available hardware rather than relying on expensive, enterprise-level servers.

3. Data Protection:

HDFS ensures data protection through replication. It creates multiple copies (replicas) of each data block and distributes them across the cluster. This redundancy safeguards against data loss due to hardware failures or other issues.

4. High Fault Tolerance:

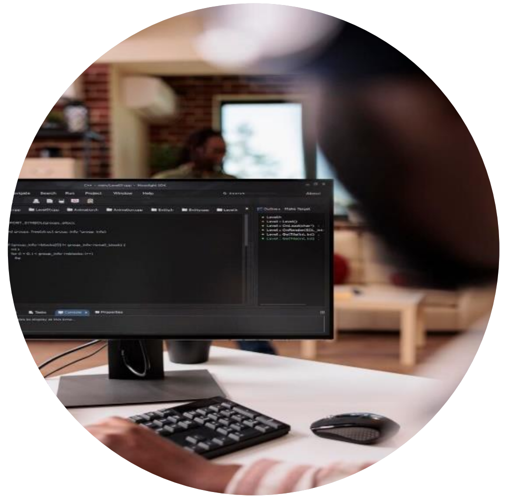
HDFS exhibits high fault tolerance by detecting and responding to system failures. If a machine or node breaks down, the data stored on that machine is seamlessly transferred to other nodes that contain replicas of the same data. This ensures continuous availability and reliability in the face of hardware failures.

Slave Nodes - Datanodes:

a. **Responsibility:** The slave nodes in HDFS are referred to as Datanodes.
b. **Operations:** Datanodes are responsible for performing tasks such as reading, writing, processing, and replicating data.
c. **Heartbeats:** Datanodes regularly send signals, known as heartbeats, to the Namenode. These heartbeats indicate the status and availability of the Datanodes.
d. **Status Indication:** The health and operational status of Datanodes are monitored by the Namenode through these heartbeats.

Master Node - Namenode:

a. **Responsibility:** The master node in HDFS is called the Namenode.
b. **Operations:** Namenode manages the operations of the data nodes, overseeing tasks such as reading, writing, and replicating data.
c. **Metadata:** It keeps track of metadata, which includes information about the location and health of data blocks, among other details.
d. **Centralized Control:** The Namenode serves as the central control point for the HDFS cluster.



Name Node

Role: The NameNode operates as the master in a Hadoop cluster, providing guidance to the Datanodes (Slaves).

Function: Primarily used for storing metadata, which includes information about the data stored in the cluster.

Metadata Examples: Transaction logs tracking user activity, file names, sizes, and details about the location of Datanodes (block numbers, block IDs).

Communication: Instructs DataNodes on operations such as deletion, creation, replication, etc.

Efficiency: Helps in identifying the closest DataNode for faster communication by storing location information.

Data Node

Role: DataNodes serve as slave nodes in the Hadoop cluster and are responsible for storing data.

Number of DataNodes: The Hadoop cluster can have varying numbers of DataNodes, typically ranging from 1 to 500 or more.

Storage Capacity: Higher storage capacity in DataNodes allows the cluster to store more data efficiently.

Replication: Data is replicated among multiple DataNodes by the NameNode for fault tolerance.

Data Replication: The data, represented in blue, grey, and red in the image, is replicated across three DataNodes.

Default Replication: By default, data is replicated three times to ensure redundancy and availability.

Fault Tolerance: If a machine fails, a new machine with the same data can replace it, thanks to data replication.

Hadoop Mapreduce

Simple and powerful

Hadoop MapReduce serves as the processing unit in the Hadoop ecosystem.

Processing occurs on slave nodes, and the final output is sent to the master node.

Data Handling Code

Uses a small, coded data containing code to handle large volumes of raw data.

The coded data is typically very small compared to the raw data.

Key Features

Integral part of Apache Hadoop, enabling programmers to manage massive data while writing programs.

Written in Java, MapReduce processes vast data by dividing it into small, parallelizable bits.

MapReduce Algorithm

Comprises two main parts: Map and Reduce.

Map function processes data, converting it into tuples.

Reduce function takes the Map output, combines it with another set of tuples, and generates a new set of tuples.

Parallel Processing

Core to Hadoop's functionality, leveraging MapReduce for parallel processing.

Enables big data processing on multiple computers within the same cluster.

Map Stage:

Input data is converted using the mapper tool.

Data can be stored in various formats in HDFS (e.g., folders or directories).

Sequentially, the entire dataset passes through the Map Function, transforming it into tuples.

Reduce Stage:

Data is shuffled and partially reduced.

Uses the Map function's output for data processing.

Generates a new output stored in the Hadoop Distributed File System (HDFS).

YARN (Yet Another Resource Negotiator):

Resource Control and Scheduling Separation

YARN separates resource control and work scheduling functions into different daemons.

Resource Manager:

Responsibility: Delegates work to all applications in the system.
Resource Allocation: Allocates resources to applications, manages overall cluster resources.
Work Scheduling: Ensures proper scheduling of jobs in the cluster.

Resource Allocation

Responsible for allocating resources to various applications within the Hadoop cluster.

Node Manager:

Responsibility: Manages containers and tracks their resource usage (CPU, disk, memory, network).
Information to Resource Manager: Sends resource usage information to the Resource Manager.

Hadoop YARN Overview:

Acronym for Yet Another Resource Negotiator.

Serves as Hadoop's resource management unit, introduced in Hadoop version 2.

Functions as an operating system for Hadoop, utilizing HDFS as a foundation.

Roles of Hadoop YARN:

Preventing Overloading: Manages cluster resources to avoid overloading a single server.
Work Schedule Management: Ensures proper scheduling of jobs to appropriate nodes.

03. Infogaphics Design

Job Request from Client: Client computer requests the execution of a query or retrieval of code for data processing.

Resource Manager (Hadoop YARN): Responsible for resource allocation and management. Receives the job request from the client.

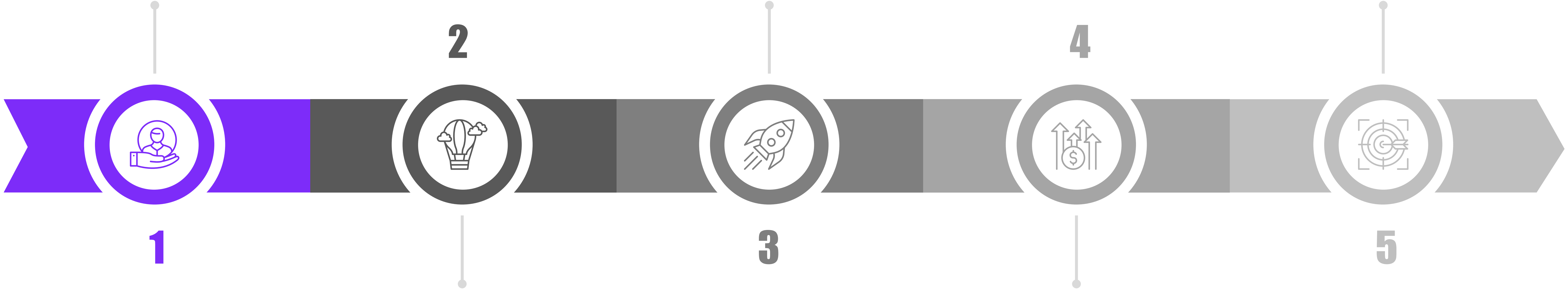
Node Managers: Each node has its Node Manager. Monitors resource usage of nodes, including RAM, CPU, and hard drives.

Containers: Physical resources (RAM, CPU, etc.) are encapsulated within containers. Node Managers manage containers and their resource usage.

App Master: When a job request is received, the Application Master requests a container from the Node Manager.

Resource Return: After completing the job, the resource is returned to the Resource Manager through the Node Manager.

Components of YARN



Global Resource Manager

Accepts user work submissions and schedules them by allocating resources.

Central authority for resource allocation in the YARN cluster

Node Manager

Each Node has a Node Manager that reports back to the Resource Manager on the performance of each node.

Monitors and reports the resource usage of individual nodes to the Resource Manager.

Application Master

Each framework has its own Application Master.

Aids the Node Manager in executing and monitoring tasks.

Facilitates a smoother resource allocation process within the cluster.

Resource Container

Operated by Node Managers, these containers distribute system resources allocated to individual applications.

Ensures efficient utilization of system resources for each application.

The components of Hadoop YARN work collaboratively to manage resource allocation, monitor node performance, execute tasks, and ensure the efficient distribution of system resources across various applications within the YARN cluster.

Advantages of HADOOP

Open Source:

Hadoop is open-source, providing free access to its source code. This facilitates transparency, modification, and adaptation based on industry requirements.

Highly Scalable Cluster:

Hadoop employs a highly scalable model, distributing large datasets across multiple inexpensive machines in a cluster. The number of nodes can be adjusted according to enterprise needs., Traditional RDBMS struggles to scale effectively with large data volumes.

Fault Tolerance:

Hadoop utilizes commodity hardware that can fail unexpectedly., Data is replicated on various DataNodes, ensuring availability even if a node fails. Default replication is set to 3 copies, configurable in the hdfs-site.xml file.

High Availability:

Fault tolerance contributes to high availability in Hadoop clusters., High availability is achieved through multiple Name Nodes, including an Active and Passive (standby) NameNode. If the Active NameNode fails, the Passive takes over seamlessly.

Cost-Effective:

Hadoop's open-source nature eliminates licensing costs, Hadoop's utilization of cost-effective commodity hardware contrasts with the expensive infrastructure needed by traditional Relational databases for handling Big Data.

Advantages of HADOOP

Flexibility:

Hadoop efficiently processes structured (e.g., MySQL data), semi-structured (e.g., XML, JSON), and unstructured data (e.g., images, videos). Its flexibility allows businesses to analyze diverse datasets from sources like social media and emails, enabling insights for various applications such as log processing, data warehousing, and fraud detection.

Easy to Use:

Hadoop is user-friendly as developers are relieved from managing intricate processing tasks, handled by Hadoop itself. The extensive Hadoop ecosystem includes tools like Hive, Pig, Spark, HBase, Mahout, etc., enhancing ease of use and expanding functionality.

Data Locality:

Hadoop employs the data locality concept, ensuring computational logic is placed near data, reducing the need to move data to computation logic. This minimizes bandwidth usage and speeds up processing by utilizing the proximity of data and computation.

Data Processing:

Hadoop utilizes HDFS for distributed file storage. Large files are broken into smaller blocks, distributed across nodes, and processed in parallel. This distributed processing model enhances performance compared to traditional Database Management Systems.

Support for Multiple Data Formats:

Hadoop supports various data formats like CSV, JSON, Avro, etc., simplifying the handling of diverse data sources. This feature is beneficial for developers and data analysts dealing with large volumes of data in different formats.

Advantages of HADOOP

High Processing Speed:	Integration with Other Tools:	Secure
Hadoop's distributed processing model enables high-speed data processing. Distributing data across multiple nodes and processing it in parallel contributes to faster data processing compared to traditional database systems.	Hadoop seamlessly integrates with popular tools such as Apache Spark, Apache Flink, and Apache Storm. This integration allows developers and data analysts to use preferred tools for building data processing pipelines and handling large datasets effectively.	Hadoop incorporates essential security features like authentication, authorization, and encryption. These features ensure data protection, permitting access only to authorized users and making Hadoop a secure platform for processing sensitive data.

Difference between Hadoop and RDBMS

S.No.	RDBMS	Hadoop
1.	Traditional row-column based databases, basically used for data storage, manipulation and retrieval.	An open-source software used for storing data and running applications or processes concurrently.
2.	In this structured data is mostly processed.	In this both structured and unstructured data is processed.
3.	It is best suited for OLTP environment.	It is best suited for BIG data.
4.	It is less scalable than Hadoop.	It is highly scalable.
5.	Data normalization is required in RDBMS.	Data normalization is not required in Hadoop.
6.	It stores transformed and aggregated data.	It stores huge volume of data.
7.	It has no latency in response.	It has some latency in response.
8.	The data schema of RDBMS is static type.	The data schema of Hadoop is dynamic type.
9.	High data integrity available.	Low data integrity available than RDBMS.
10.	Cost is applicable for licensed software.	Free of cost, as it is an open source software.

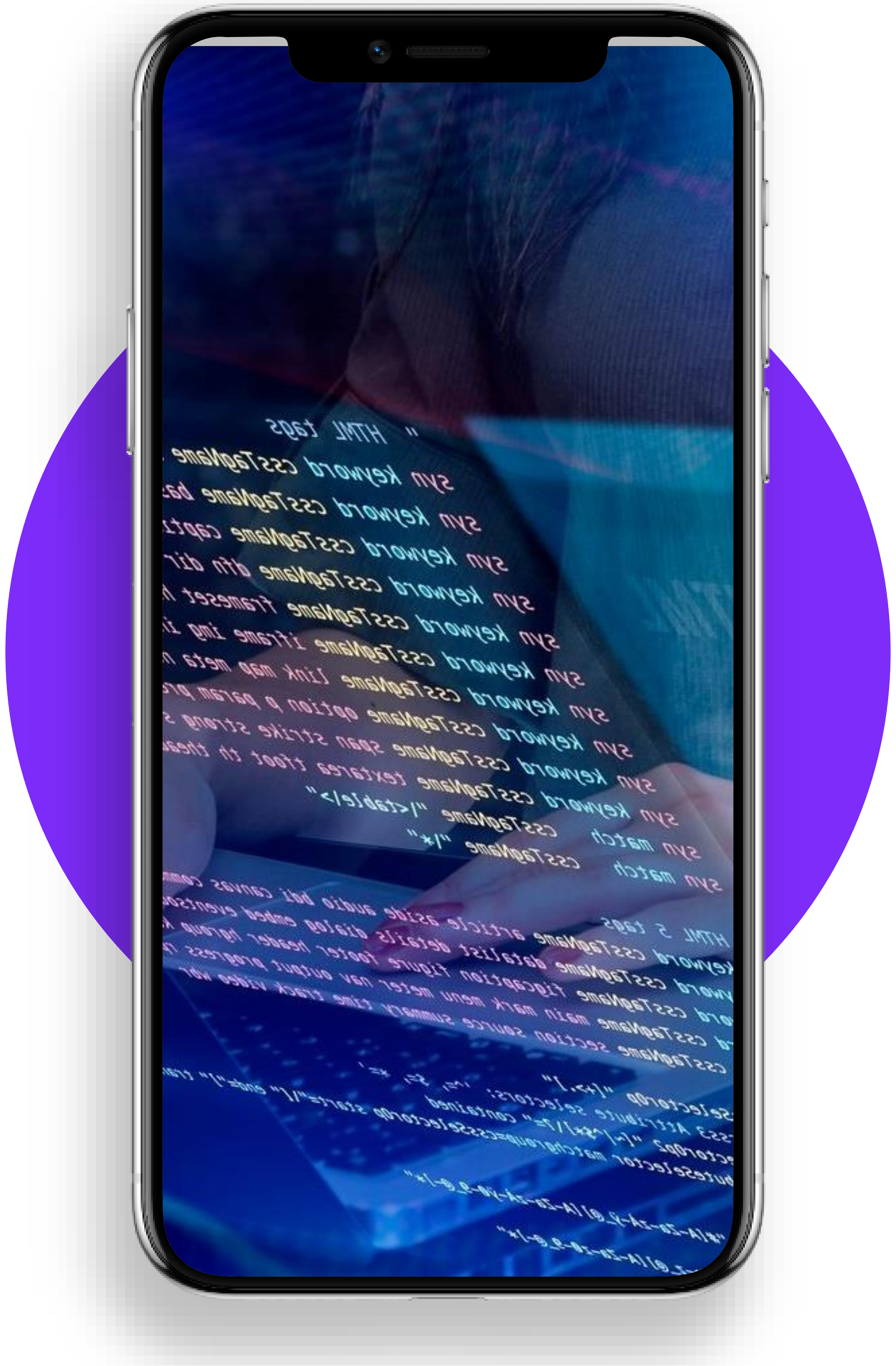
Distributed Computer Systems

A **distributed system** is an assembly of independent computers or digital devices communicating and coordinating actions through networked message passing. These systems collaborate to achieve common goals, such as processing extensive data, offering web services, or managing intricate applications.

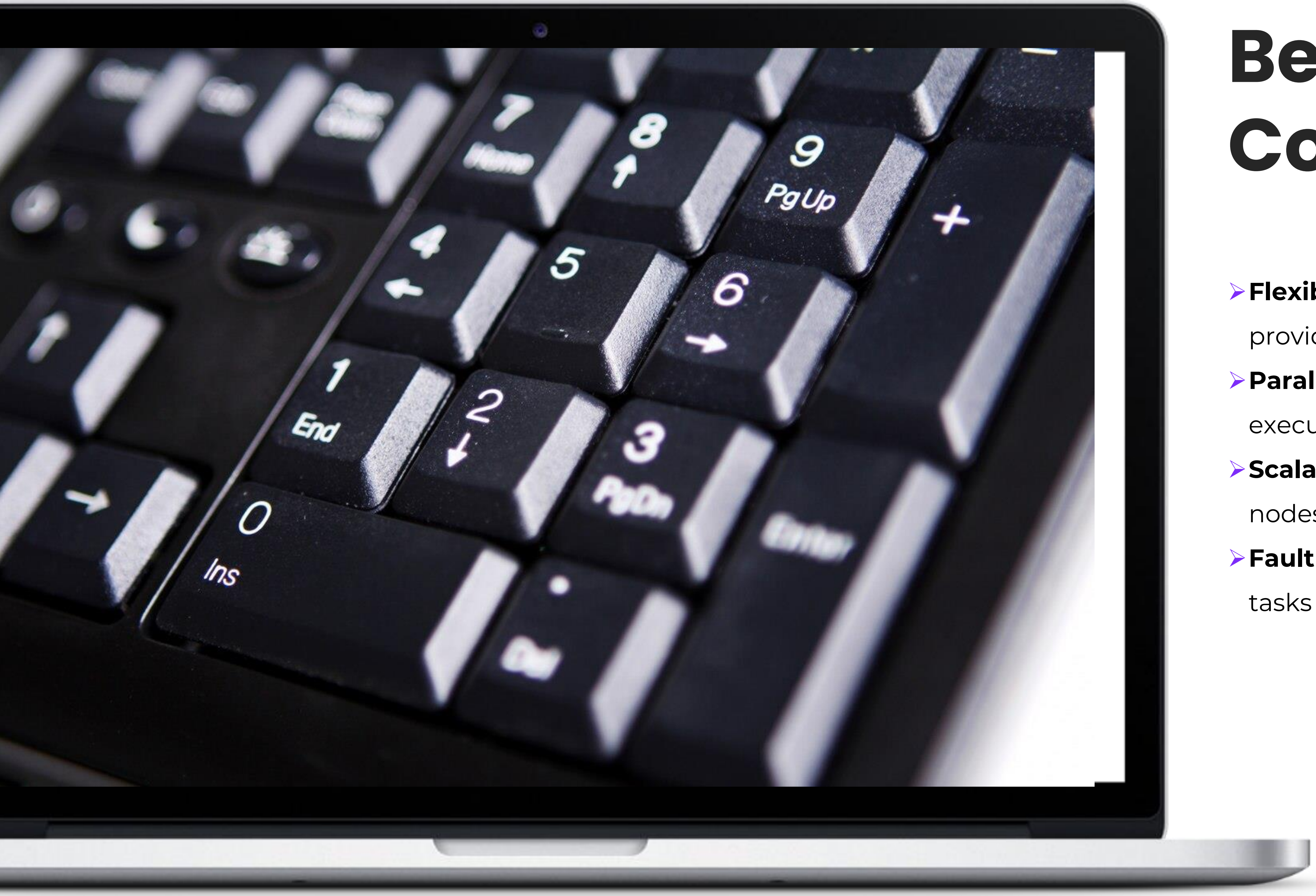
Nodes: Each computer in the system, termed a node, executes specific tasks and interacts with other nodes for information exchange, enabling coordinated actions.

var packTvCheck = packSlider.find('.pack
margin: 0,
nav: true,
var tvSwiper = new Swiper('.tv-swiper
effect: 'fade',
crossFade: true
pagination',
disableOnInteraction: false,
breakpoints: {

Systems Characteristics of Distributed Computer Systems



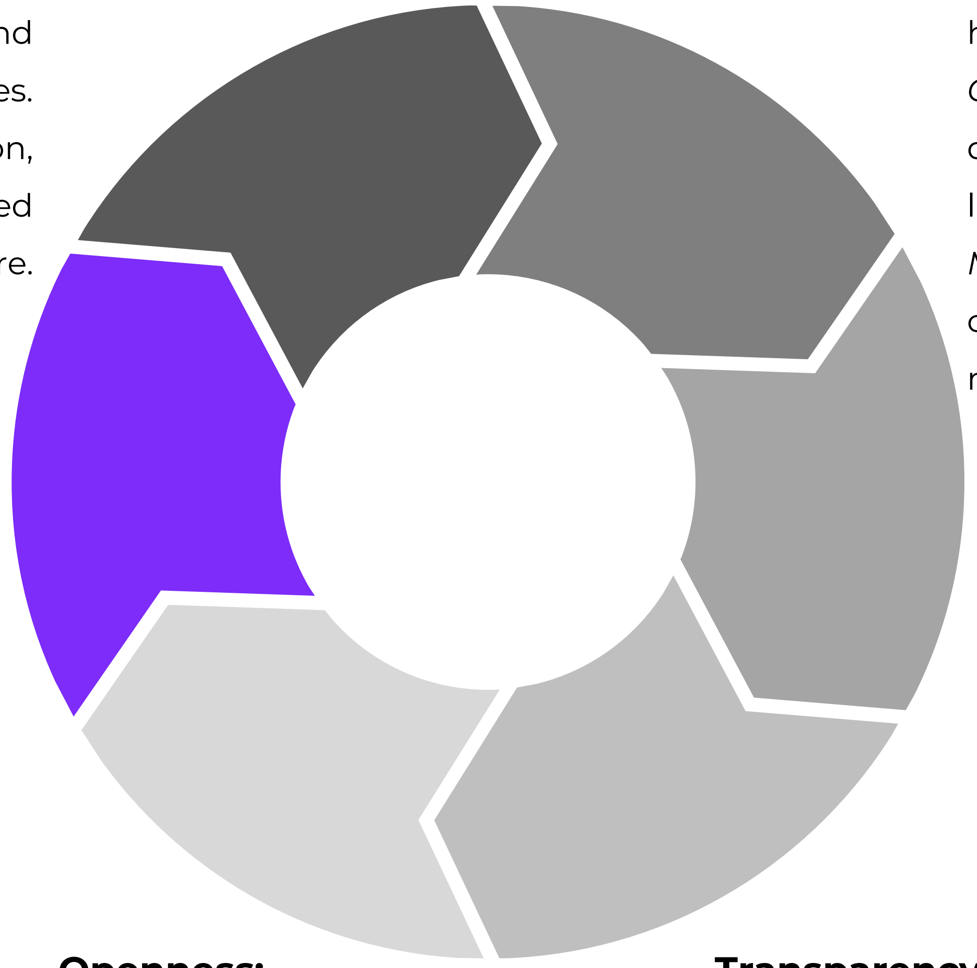
- Common Goal: Collaboration toward shared objectives, exemplified by data processing or application management.
- Communication: Nodes communicate through networked channels, facilitating information exchange.
- Node Flexibility: Nodes can be geographically dispersed with varied hardware configurations and operating systems, enhancing system adaptability.



Benefits of Distributed Computer Systems

- **Flexibility:** Geographic and hardware diversity provides system adaptability.
- **Parallel Processing:** Enables concurrent execution of tasks for enhanced efficiency.
- **Scalability:** Easily accommodates additional nodes for increased system capacity.
- **Fault Tolerance:** Redundancy and distributed tasks enhance system resilience.

Challenges in Distributed Computer Systems



Heterogeneity:

Definition: Varied hardware, software, or network configurations among nodes.

Challenges: Communication and coordination difficulties.

Mitigation: Middleware, virtualization, standardization, and service-oriented architecture.

Security:

Definition: Ensuring confidentiality, integrity, and authentication in a distributed and heterogeneous system.

Challenges: Unauthorized access, data integrity during transmission, diverse authentication locations.

Mitigation: Encryption, digital signatures, checksums, secure authentication mechanisms.

Scalability:

Definition: Difficulty in maintaining performance and availability as systems grow.

Challenges: Security, data consistency, network latency, resource allocation, and node balancing.

Mitigation: Robust fault tolerance mechanisms, efficient resource allocation algorithms.

Concurrency:

Definition: Simultaneous data processing on different nodes.

Challenges: Race conditions, communication, and synchronization complexities.

Mitigation: Efficient synchronization mechanisms, fault-tolerant concurrency control

Openness:

Definition: Achieving standards for seamless communication between systems with different protocols.

Challenges: Ensuring compatibility, data exchange, transparency, and security.

Mitigation: Standardization, protocols, and data format agreements.

Transparency:

Definition: Concealing complexity from users; failures should not impact overall performance.

Challenges: Handling diverse hardware and software configurations; security concerns.

Mitigation: Abstraction layers, fault-tolerant mechanisms, robust security measures.

Thanks!

What does Harvard and IIBS have in common?

Nothing! They know what they are doing!