Page 2: Hadoop vs. R and Installation Steps

Difference Between Hadoop and R

- **Hadoop**:
  - A powerful framework for processing and analyzing large datasets using a distributed file system (HDFS) and the MapReduce programming model.
  - Designed for scalability and fault tolerance across clusters of computers.
- **R**:
  - A programming language and environment primarily used for statistical computing and graphics.
  - Lacks robust memory management and large-scale data handling without additional packages.
- **RHIPE**:
  - An R library ("R and Hadoop Integrated Programming Environment") that integrates R with Hadoop, allowing R programmers to write map and reduce functions that are executed as Hadoop MapReduce jobs.

Is Hadoop a Programming Language?

- Hadoop is **not** a programming language but a framework primarily written in **Java**, with some native code in **C** and shell scripts for utilities.
- It supports multiple programming languages via **Hadoop Streaming**, enabling users to write MapReduce programs in languages like Python, R, etc.

Steps to Install Hadoop

- **Ubuntu**:
  1. Download VirtualBox: https://www.virtualbox.org/wiki/Downloads
  2. Follow Hadoop installation guide: https://phoenixnap.com/kb/install-hadoop-ubuntu
- **Windows**:
  - Refer to a GitHub guide: https://github.com/ruslanmv/How-to-install-Hadoop-on-Windows

---

Page 3: History and Features of Hadoop

History of Hadoop

- **Origin**: Developed by the **Apache Software Foundation**, co-founded by **Doug Cutting** and **Mike Cafarella**. Named after Doug Cutting's son's toy elephant.
- **Timeline**:
  - **October 2003**: Google published a paper on the **Google File System (GFS)**.
  - **2006**: Hadoop was created based on GFS and Google's MapReduce model. Development began with **Apache Nutch**, leading to Hadoop 0.1.0 release in April 2006.
  - **Codebase**: Approximately 6,000 lines for MapReduce and 5,000 lines for HDFS.
- **Purpose**: Hadoop is an open-source framework for distributed storage and processing of big data, designed to scale from single servers to thousands of machines.
- **Adoption**: Used by major organizations like **Yahoo, Facebook, and IBM** for data warehousing, log processing, and research.

Features of Hadoop

1. **Fault Tolerance**: Continues operation despite hardware failures.
2. **High Availability**: Ensures data is always accessible.
3. **Ease of Programming**: Simplifies distributed data processing.
4. **Flexible Storage**: Supports large-scale data storage.
5. **Low Cost**: Runs on commodity hardware.

Page 4: Hadoop Architecture

- **Foundation**: Built on the **MapReduce** programming model introduced by Google.
- **Adoption**: Used by companies like **Facebook, Yahoo, Netflix, and eBay** for big data processing.
- **Components**:
  1. **MapReduce**: A programming model for processing large datasets in parallel.
  2. **HDFS (Hadoop Distributed File System)**: A distributed storage system for large datasets.
  3. **YARN (Yet Another Resource Negotiator)**: Manages resources and job scheduling.
  4. **Hadoop Common**: Provides libraries and utilities for other components.

Page 5: Introduction to Hadoop

What is Hadoop?

- An **open-source** framework for storing and processing large datasets in a distributed environment.
- Primarily written in **Java**, with some **C** and shell scripts.
- Based on the **MapReduce** model for parallel processing.

Main Components

1. **HDFS**:
   - Stores large datasets across multiple machines.
   - Cost-effective as it uses commodity hardware.
2. **YARN**:
   - Manages resource allocation (CPU, memory) for data processing.

Additional Modules

- **Hive**: SQL-like query language for data querying.
- **Pig**: High-level platform for creating MapReduce programs.
- **HBase**: Distributed, non-relational database for real-time read/write access.

Use Cases

- **Data Warehousing**: Storing and analyzing large datasets.
- **Business Intelligence**: Extracting insights from data.
- **Machine Learning**: Processing data for predictive models.
- **Data Processing, Analysis, and Mining**: Handling large-scale data tasks.

Page 6: Key Features for Big Data Processing

- **Distributed Storage**: Stores data across multiple machines.
- **Scalability**: Scales from single servers to thousands of nodes.
- **Fault Tolerance**: Handles hardware failures seamlessly.
- **Data Locality**: Processes data on the node where it is stored, reducing network traffic.
- **High Availability**: Ensures continuous data access.
- **Flexible Data Processing**: Supports various data processing tasks via MapReduce.
- **Data Integrity**: Uses checksums to ensure data consistency.
- **Data Replication**: Replicates data for fault tolerance.
- **Data Compression**: Reduces storage needs and improves performance.
- **YARN**: Supports multiple processing engines (batch, real-time, interactive SQL).

Pages 7-10: MapReduce

Overview (Page 7)

- **Definition**: MapReduce is an algorithm and data structure built on YARN for distributed parallel processing.

- **Purpose**: Enables fast-dot-comma separated values (CSV) file is not included in the response due to the instruction to exclude it.
- **Phases**:
  1. **Map Phase**: Breaks data into key-value pairs (tuples).
  2. **Reduce Phase**: Combines and processes tuples based on keys (e.g., sorting, summation).

MapReduce Process (Page 8)

- **Input**: A large dataset.
- **Map Function**: Splits data into key-value pairs.
- **Reduce Function**: Aggregates pairs based on keys and produces final output.
- **Output**: Processed data stored or displayed.

Map Task (Page 9)

1. **RecordReader**: Breaks records into key-value pairs, with keys as location info and values as data.
2. **Map**: User-defined function to process tuples, generating zero or multiple key-value pairs.
3. **Combiner** (Optional): Groups intermediate data locally, similar to a mini-reducer.
4. **Partitioner**: Assigns key-value pairs to reducers using a hash-based partitioning algorithm.

Reduce Task (Page 10)

1. **Shuffle and Sort**: Transfers intermediate key-value pairs from mappers to reducers, sorting them by key.
2. **Reduce**: Aggregates and processes sorted key-value pairs.
3. **OutputFormat**: Writes final key-value pairs to a file, with each record on a new line, separated by spaces.

Pages 11-17: Hadoop Distributed File System (HDFS)

Overview (Page 11)

- **Purpose**: Provides scalable, fault-tolerant, and high-availability storage.
- **Design**: Stores data in large blocks (default 128 MB) on commodity hardware.
- **Nodes**:
  1. **NameNode (Master)**: Stores metadata (e.g., file names, sizes, block locations) and manages operations (create, delete, replicate).
  2. **DataNode (Slave)**: Stores actual data blocks; multiple DataNodes for scalability.

File Blocks in HDFS (Pages 13-15)

- **Block Size**: Default 128 MB, configurable up to 256 MB.
- **Example**: A 400 MB file splits into 3 blocks of 128 MB and 1 block of 16 MB.
- **Replication**: Default replication factor is 3, ensuring fault tolerance by creating multiple copies of each block.
- **Rack Awareness**: Optimizes data placement to minimize network traffic by choosing nearby DataNodes.

HDFS Architecture (Pages 16-17)

- Splits files into blocks and distributes them across cluster nodes.
- Supports fault tolerance through replication and node failure recovery.
- **Advantages**:
  - Cost-effective, reliable, scalable, fault-tolerant, and capable of parallel processing.
- **Disadvantages**:
  - Inefficient for small data, potential stability issues, and restrictive nature.

Common Frameworks (Page 18)

- **Hive**: SQL-like querying for MapReduce jobs.

- **Drill**: Data exploration with user-defined functions.
- **Storm**: Real-time data streaming.
- **Spark**: Fast data processing with machine learning support.
- **Pig**: SQL-like language (Pig Latin) for data transformation.
- **Tez**: Optimizes Hive and Pig performance.

Page 19: Advantages of Hadoop

- **Scalability**: Easily scales by adding nodes.
- **Cost-Effective**: Uses commodity hardware.
- **Fault Tolerance**: Handles node failures.
- **Flexibility**: Processes structured, semi-structured, and unstructured data.
- **Open-Source**: Free to use and modify.
- **Community Support**: Large developer and user community.
- **Integration**: Works with other big data tools (Spark, Storm, Flink).

Page 20: Disadvantages of Hadoop

- **Ineffective for Small Data**: Not optimized for small datasets.
- **Complexity**: Difficult to set up and maintain.
- **Latency**: Unsuitable for low-latency or real-time processing.
- **Limited Support**:
  - Structured data processing.
  - Ad-hoc queries.
  - Graph and machine learning workloads.
- **Security**: Lacks built-in encryption and authentication.
- **Cost**: Expensive for large-scale setups.
- **Data Loss Risk**: Possible with single-node failures.
- **Data Governance**: Lacks features for data lineage, quality, and auditing.

Page 21: YARN (Yet Another Resource Negotiator)

- **Role**: Manages resources and schedules jobs for MapReduce.
- **Functions**:
  - **Job Scheduler**: Divides tasks into smaller jobs, assigns them to nodes, and tracks priorities and dependencies.
  - **Resource Manager**: Allocates CPU, memory, and other resources.
- **Features**:
  - Multi-tenancy, scalability, cluster utilization, and compatibility.

Page 22: Hadoop Common

- **Definition**: Java libraries and scripts supporting HDFS, YARN, and MapReduce.
- **Purpose**: Handles hardware failures automatically, ensuring cluster reliability.

Page 23: Fundamental Concepts

- **File System**: Manages read/write operations (e.g., NTFS, EXT, HDFS).
- **Block**: Divides large files into smaller chunks (e.g., 16 KB for NTFS, 128 MB for HDFS).
- **Client-Server**:
  - Client sends requests; server responds.
- **File System Types**:
  - Standalone: NTFS, EXT.
  - Distributed: HDFS, S3, GFS.
- **Distributed System Types**:

    - Master-Slave: Hadoop.
    - Peer-to-Peer: Cassandra.
- **Process**: A program in execution.
- **Daemon Process**: Background process.
- **Cluster and Node**:
    - Node: Individual machine.
    - Cluster: Group of nodes.
- **Client API**: Application programming interface for interacting with Hadoop.

Page 24: Browse Directory Example

- Displays a sample HDFS directory listing with columns:
    - **Permission, Owner, Group, Size, Last Modified, Replication, Block Size, Name**.
- Example entry: A directory named "sample" with specific permissions and metadata.