Advanced Data Science

Hadoop

Session 9

Kiran Waghmare
Program Manager
C-DAC Mumbai

Agenda

Hadoop Architecture
- Introduction to Hadoop;
- Framework;

Modules:
- Hadoop Common,
- Hadoop YARN,
- Hadoop Distributed File Systems (HDFS),
- Hadoop MapReduce;
- Architecture;
- Environment Setup;
- Operation Modes;

HDFS Overview:
- Features of HDFS,
- Architecture,
- Goals;
- HDFS Operations;
- Hadoop Command Reference;

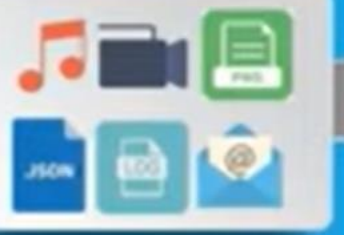
MapReduce:
- Algorithms,
- Terminologies,
- MapReduce Command and Jobs;

Streaming:
- Mapper Phase Code,
- Reducer Phase Code;
- Multi-Node Cluster.

5 V's of Big Data

Volume

Data is being generated at an accelerating speed



Variety

Different kinds of data is being generated from various sources

Velocity

Data is being generated at an alarming rate

Value

Mechanism to bring the correct meaning out of the data

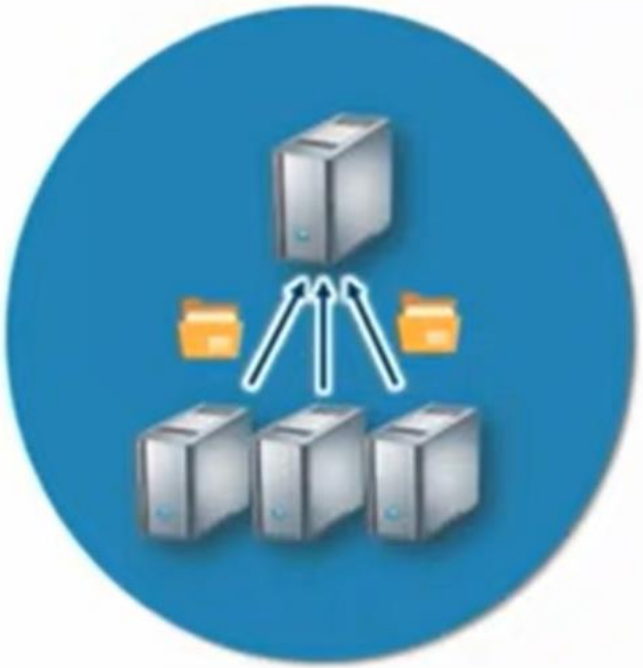
Veracity

Uncertainty and inconsistencies in the data

Problems with Big Data

Storing huge and exponentially growing datasets

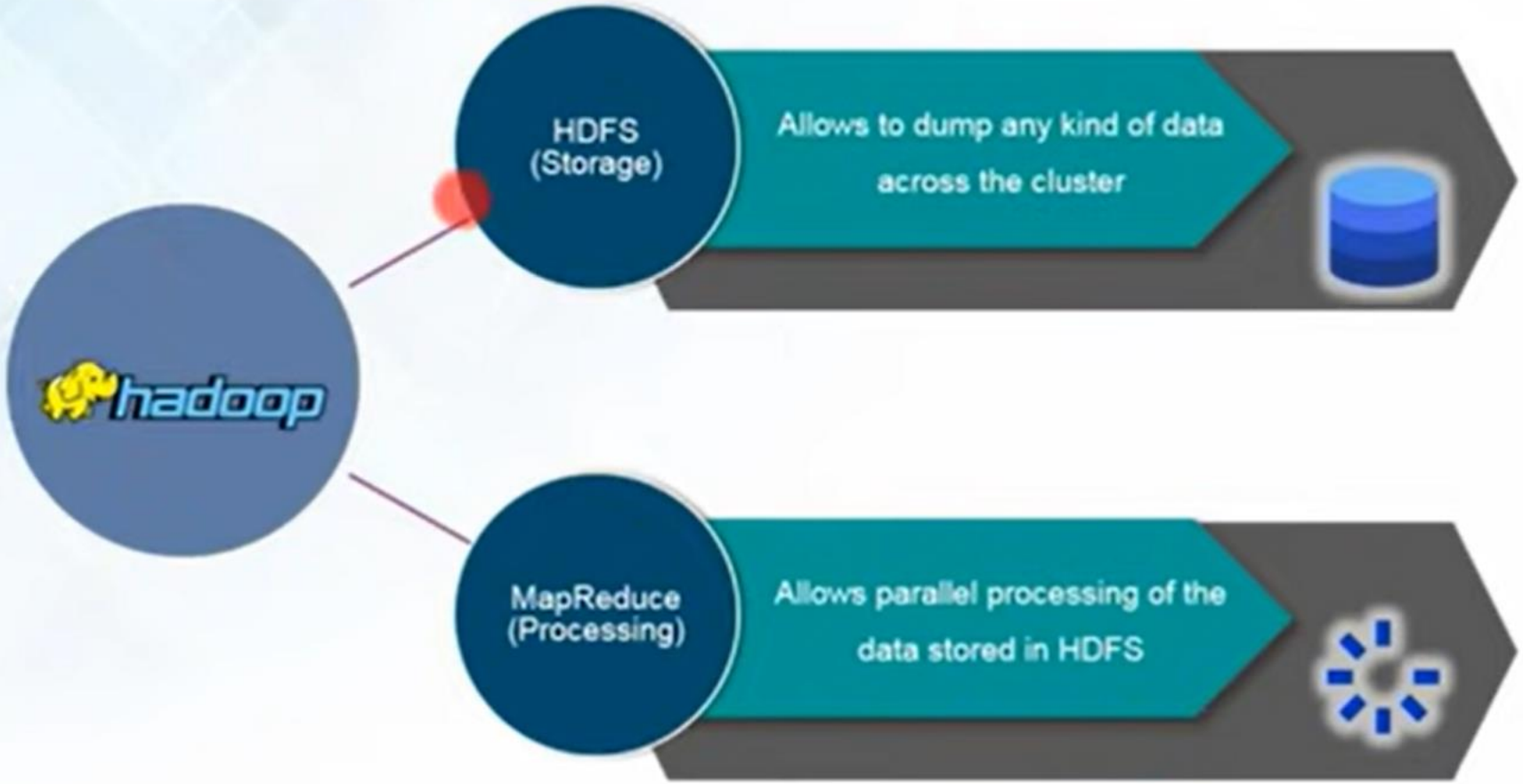
Processing data having complex structure (structured, un-structured, semi-structured)

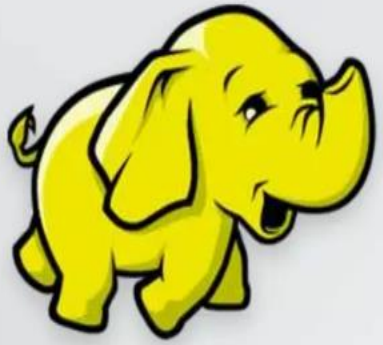


Bringing huge amount of data to computation unit becomes a bottleneck

Hadoop

Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion





What is Hadoop?

- Hadoop is an open source software programming framework for storing a large amount of data and performing the computation.
- Its framework is based on Java programming with some native code in C and shell scripts.
- Hadoop is used for some advanced level of analytics, which includes Machine Learning and data mining

Need for Hadoop

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination

Programmers

No longer need to worry about

Q: Where file is located?

Q: How to handle failures & data lost?

Q: How to divide computation?

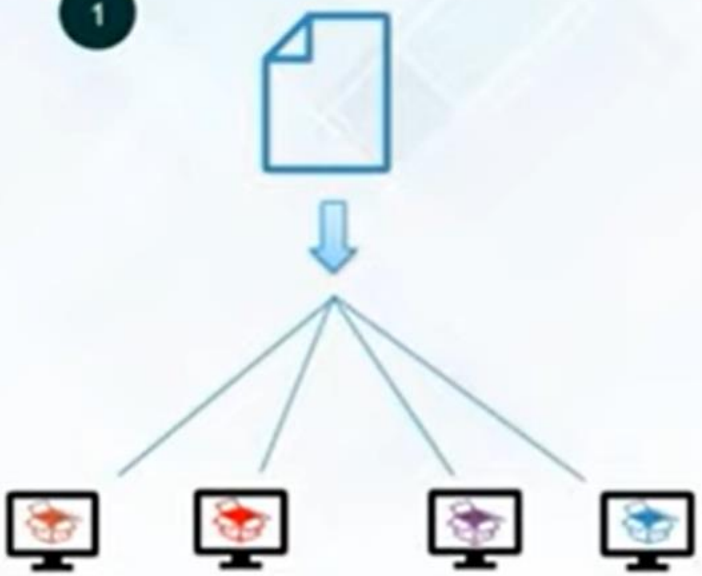
Q: How to program for scaling?

Hadoop

Storing exponentially growing huge datasets

HDFS, storage unit of Hadoop is a Distributed File System

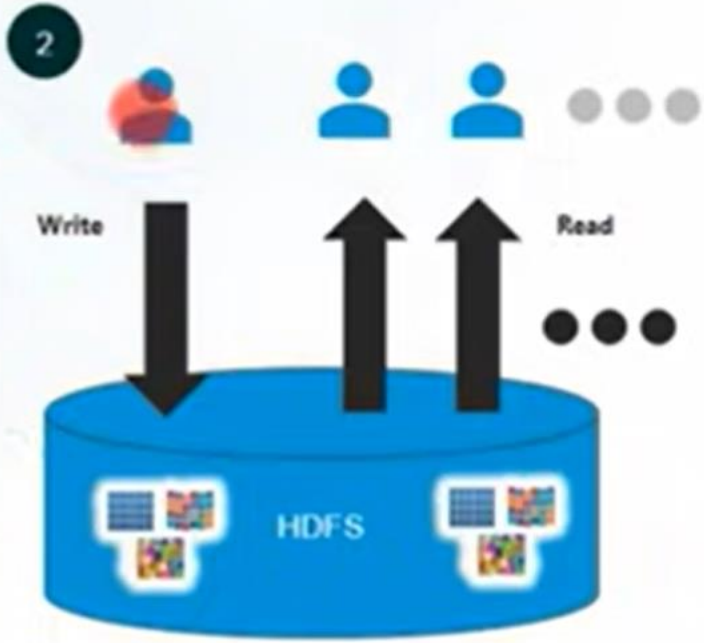
1



Storing unstructured data

Allows to store any kind of data, be it structured, semi-structured or unstructured

2

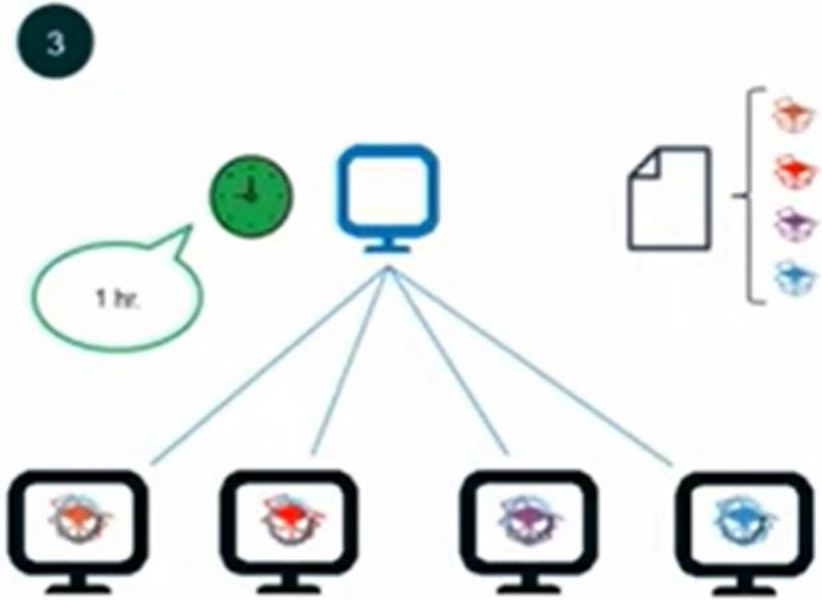


Processing data faster

Provides parallel processing of data present in HDFS

Allows to process data locally i.e. each node works with a part of data which is stored on it

3



History of Hadoop

- **Apache Software Foundation** is the developers of Hadoop, and it's co-founders are **Doug Cutting** and **Mike Cafarella**.
- It's co-founder Doug Cutting named it on his son's toy elephant. In October 2003 the first paper release was Google File System.
- In January 2006, MapReduce development started on the Apache Nutch which consisted of around 6000 lines coding for it and around 5000 lines coding for HDFS.
- In April 2006 Hadoop 0.1.0 was released.

hadoop overview

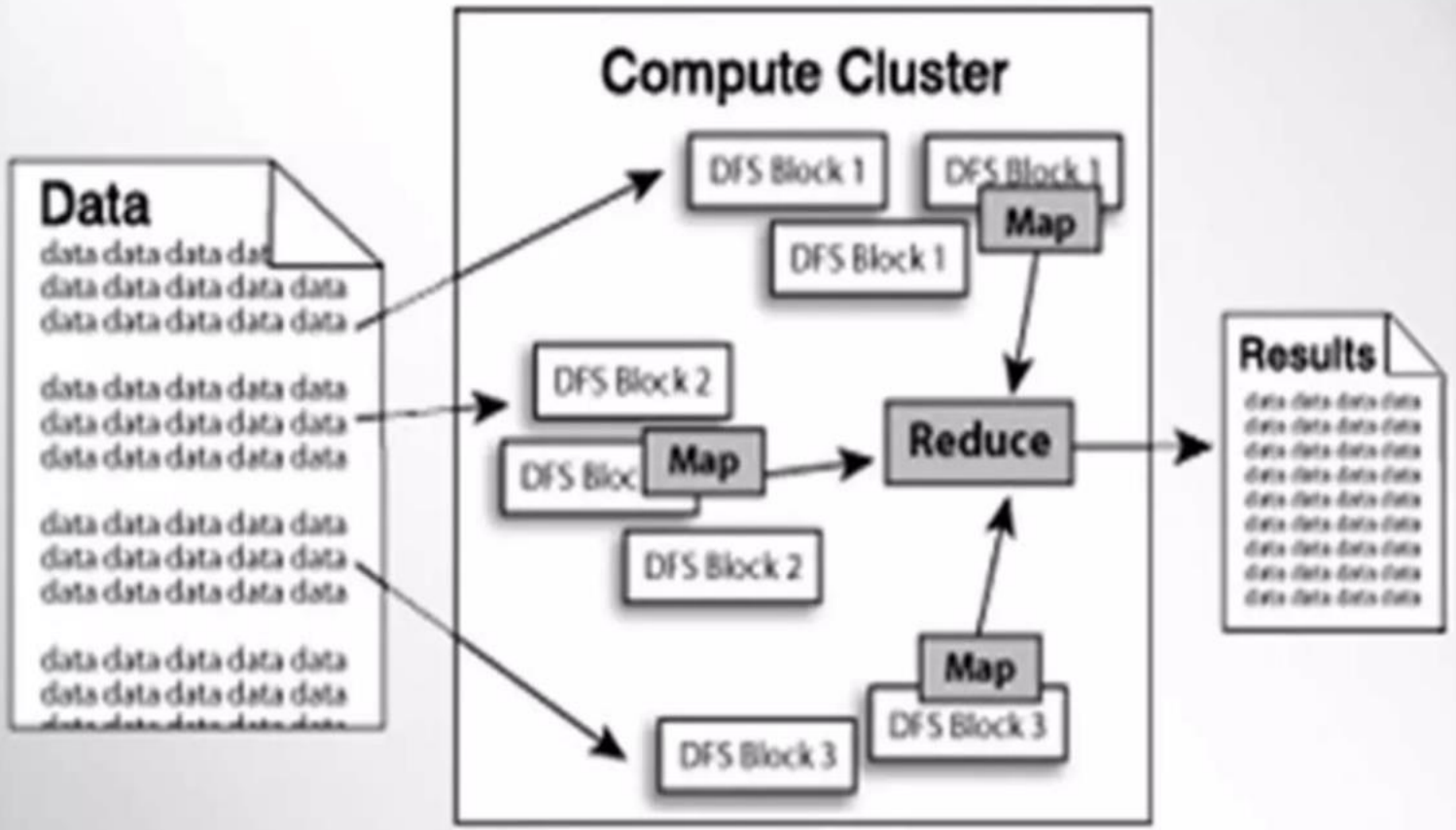


image courtesy of the Apache Software Foundation

Hadoop Framework

2.1 Core Modules

2.1.1 Hadoop Common

- **Description**: Provides the common utilities and libraries needed by other Hadoop modules.
- **Components**: Includes utilities for managing file systems, serialization, and Hadoop commands.
- **Functions**: Ensures interoperability among different Hadoop components.

2.1.2 Hadoop YARN (Yet Another Resource Negotiator)

- **Description**: Manages and schedules resources in the Hadoop cluster.
- **Components**:
  - **ResourceManager**: Manages cluster resources and job scheduling.
  - **NodeManager**: Manages resources and runs tasks on individual nodes.
  - **ApplicationMaster**: Manages the execution of individual applications.
- **Functions**: Allocates resources dynamically, optimizes resource usage.

Core Modules

2.1.3 Hadoop Distributed File System (HDFS)

- **Description**: A distributed file system designed to store large files across multiple nodes.
- **Components**:
  - **NameNode**: Manages the file system namespace and metadata.
  - **DataNode**: Stores the actual data blocks.
- **Functions**: Provides high-throughput access to application data, ensures fault tolerance by replicating data.

2.1.4 Hadoop MapReduce

- **Description**: A programming model and processing engine for parallel computation of large data sets.
- **Components**:
  - **Map**: Processes input data into intermediate key-value pairs.
  - **Reduce**: Aggregates and processes intermediate data to produce final results.
- **Functions**: Enables distributed data processing by dividing tasks into smaller chunks.

2.2 Architecture

- **Master-Slave Architecture**: Hadoop follows a master-slave architecture where:
  - **Master Nodes**: Handle management and coordination tasks.
  - **Slave Nodes**: Perform the actual data processing and storage.

2.3 Environment Setup

1. **Installation**:
   - Download Hadoop binaries from the Apache website.
   - Extract the binaries and configure environment variables (e.g., `HADOOP_HOME`).
   - Configure core-site.xml, hdfs-site.xml, mapred-site.xml, and yarn-site.xml files.
2. **Configuration**:
   - **Core Site Configuration**: Configures the Hadoop core system parameters.
   - **HDFS Configuration**: Configures HDFS properties.
   - **MapReduce Configuration**: Configures MapReduce properties.
   - **YARN Configuration**: Configures YARN properties.
3. **Start Services**:
   - Use commands like `start-dfs.sh` and `start-yarn.sh` to start HDFS and YARN services.

2.4 Operation Modes

- **Local Mode**: Hadoop runs in a single Java process, suitable for debugging and development.
- **Pseudo-Distributed Mode**: Each Hadoop daemon runs as a separate Java process on a single machine, simulating a multi-node environment.
- **Cluster Mode**: Hadoop runs on a cluster of machines, with HDFS and YARN managing data and resources across nodes.

Advantages and Disadvantages of Hadoop

Advantages:

- Ability to store a large amount of data.
- High flexibility.
- Cost effective.
- High computational power.
- Tasks are independent.
- Linear scaling.

Disadvantages:

- Not very effective for small data.
- Hard cluster management.
- Has stability issues.
- Security concerns.

3. HDFS Overview

3.1 Features of HDFS

- **High Throughput**: Optimized for large data sets and sequential read/write operations.
- **Fault Tolerance**: Data is replicated across multiple nodes to handle hardware failures.
- **Scalability**: Can scale horizontally by adding more nodes to the cluster.
- **Data Locality**: Moves computation close to the data to improve performance.

3.2 Architecture

- **NameNode**: The master server that manages the file system namespace and metadata.
- **DataNode**: The worker nodes that store the actual data blocks.
- **Secondary NameNode**: Maintains checkpoints and merges the namespace image.

3.3 Goals

- **Reliability**: Ensure data is reliably stored and retrieved.
- **Scalability**: Handle the growing volume of data by scaling horizontally.
- **Fault Tolerance**: Maintain data integrity and availability despite node failures.

3.4 HDFS Operations

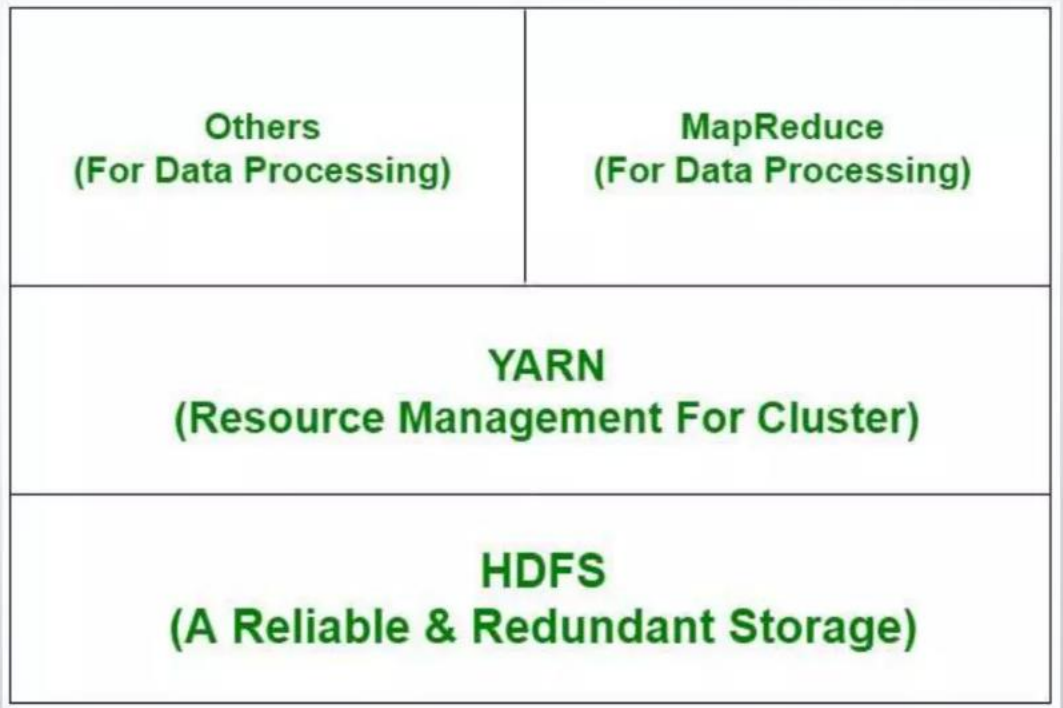
- **File Operations**: Creating, reading, writing, and deleting files.
- **Directory Operations**: Creating, listing, and deleting directories.
- **Replication Management**: Setting and monitoring replication factors for data blocks.

3.5 Hadoop Command Reference

- `hdfs dfs -ls /` : Lists files in the HDFS root directory.
- `hdfs dfs -put localfile /hdfs/path` : Uploads a local file to HDFS.
- `hdfs dfs -get /hdfs/path localdir` : Downloads a file from HDFS to a local directory.
- `hdfs dfs -rm /hdfs/path` : Deletes a file from HDFS.

Hadoop Distributed File System

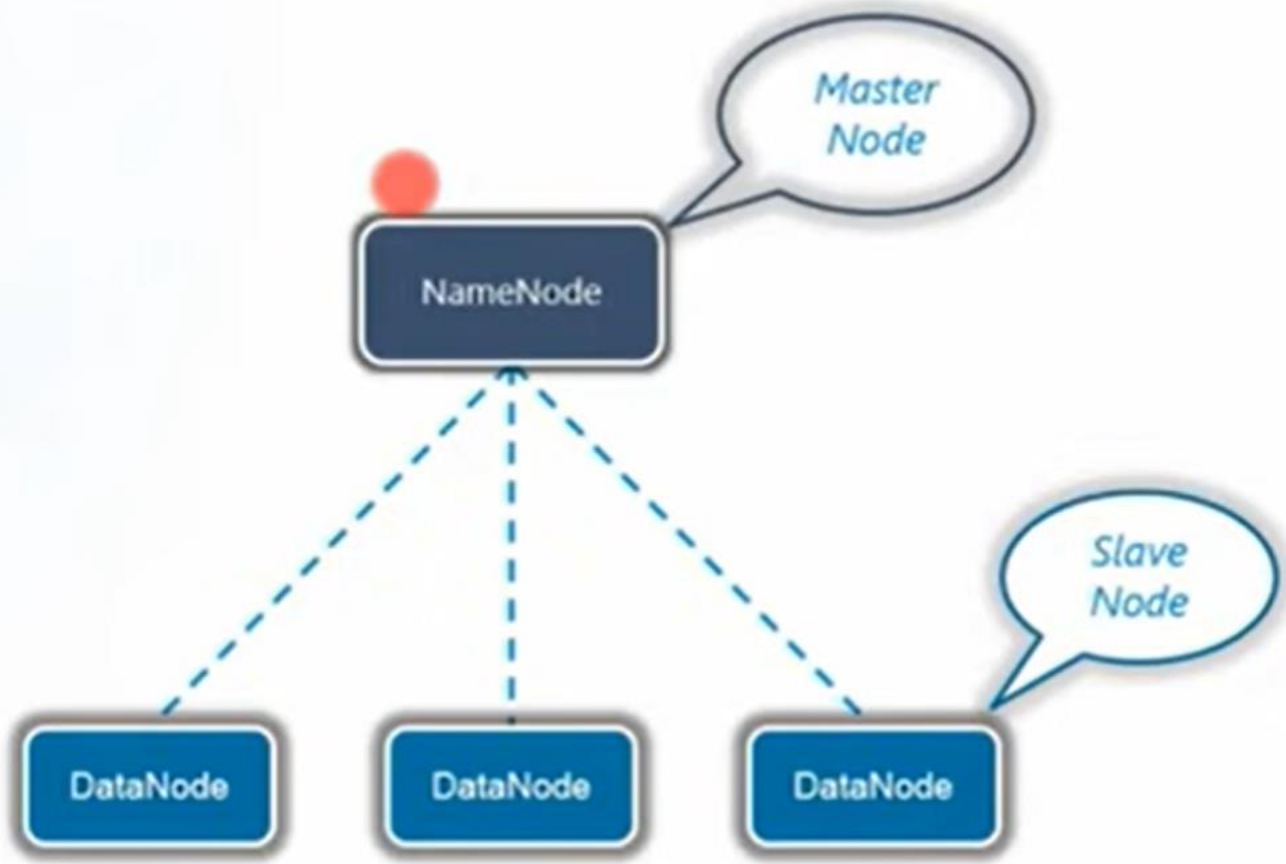
- It has distributed file system known as HDFS and this HDFS splits files into blocks and sends them across various nodes in form of large clusters.
- Also in case of a node failure, the system operates and data transfer takes place between the nodes which are facilitated by HDFS.



HDFS

HDFS

- Storage unit of Hadoop
- Distributed File System
- Divide files (input data) into smaller chunks and stores it across the cluster
- Vertical Scaling as per requirement
- Stores any kind of data
- No schema validation is done while dumping data

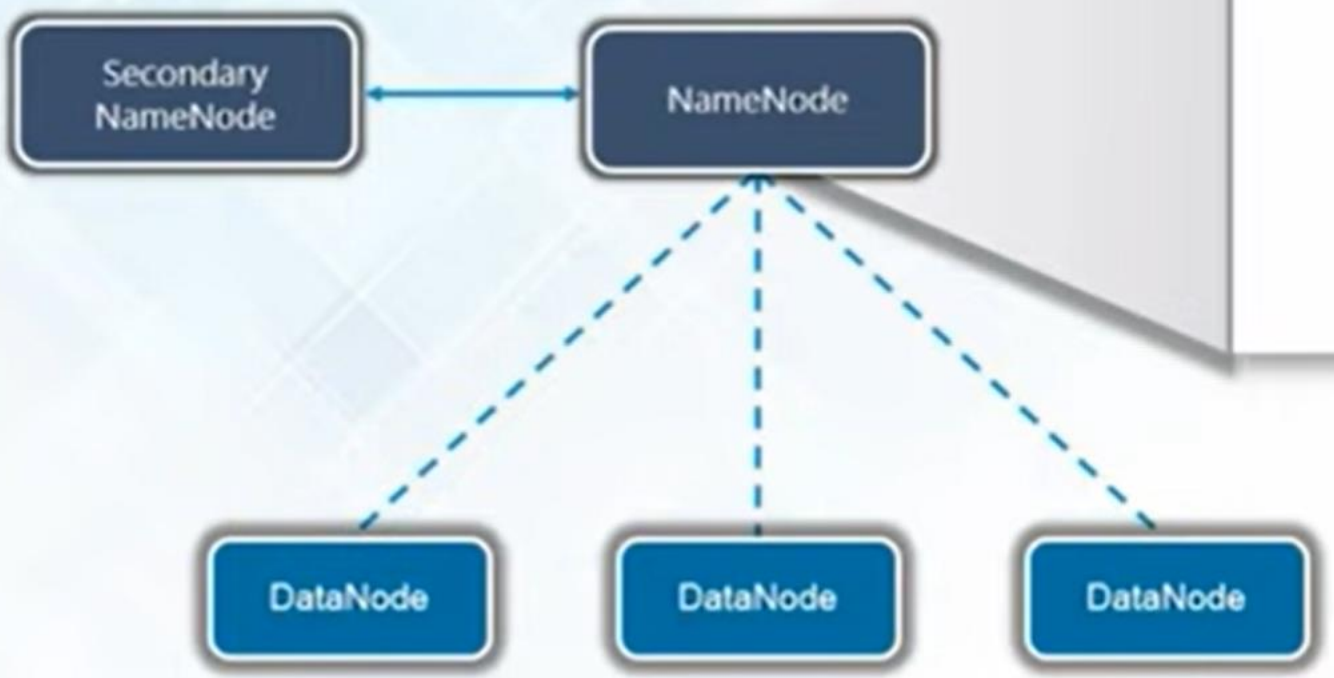


HDFS Block

- HDFS stores the data in form of blocks
- Block size can be configured base on requirements

Note: The default Block Size is 128 MB

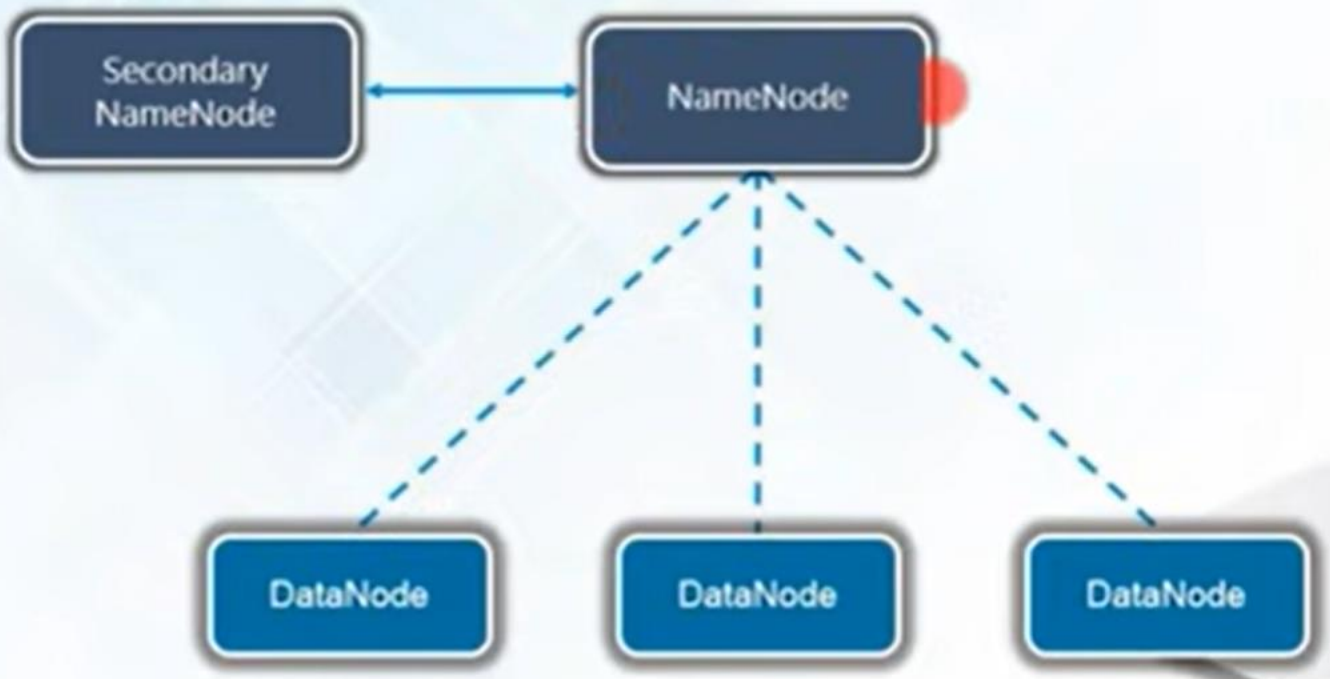
NameNode



NameNode

- Master daemon
- Maintains and Manages DataNodes
- Records metadata e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

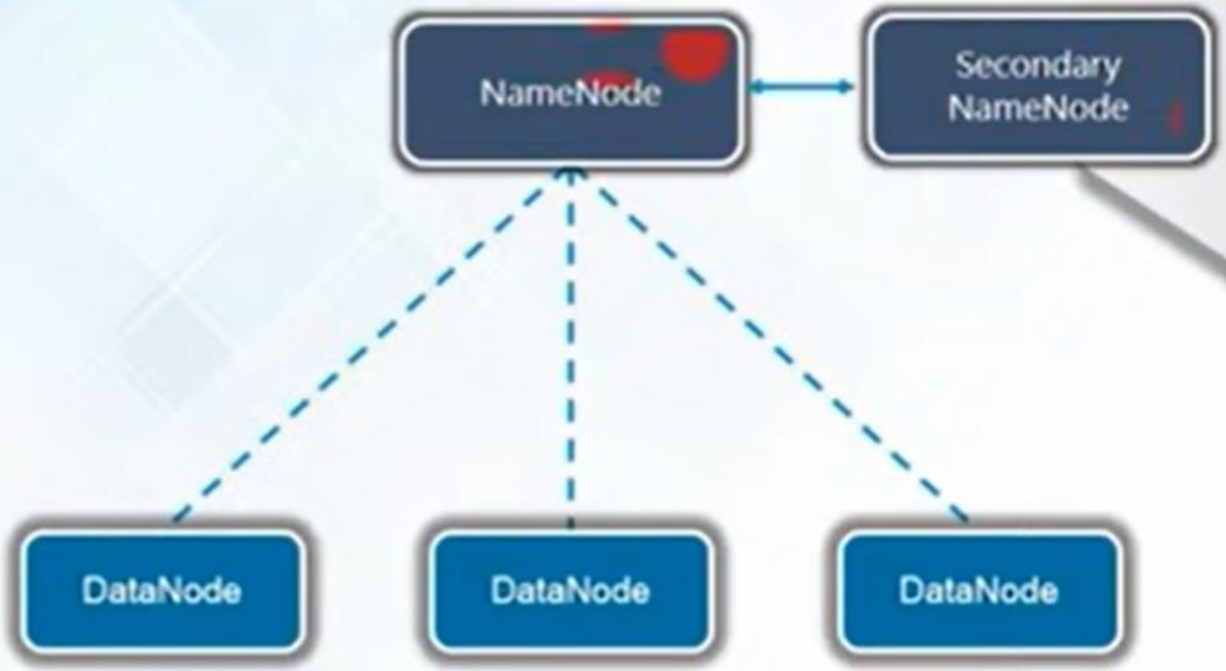
DataNode



DataNode

- Slave daemons
- Stores actual data
- Serves read and write requests

Secondary NameNode

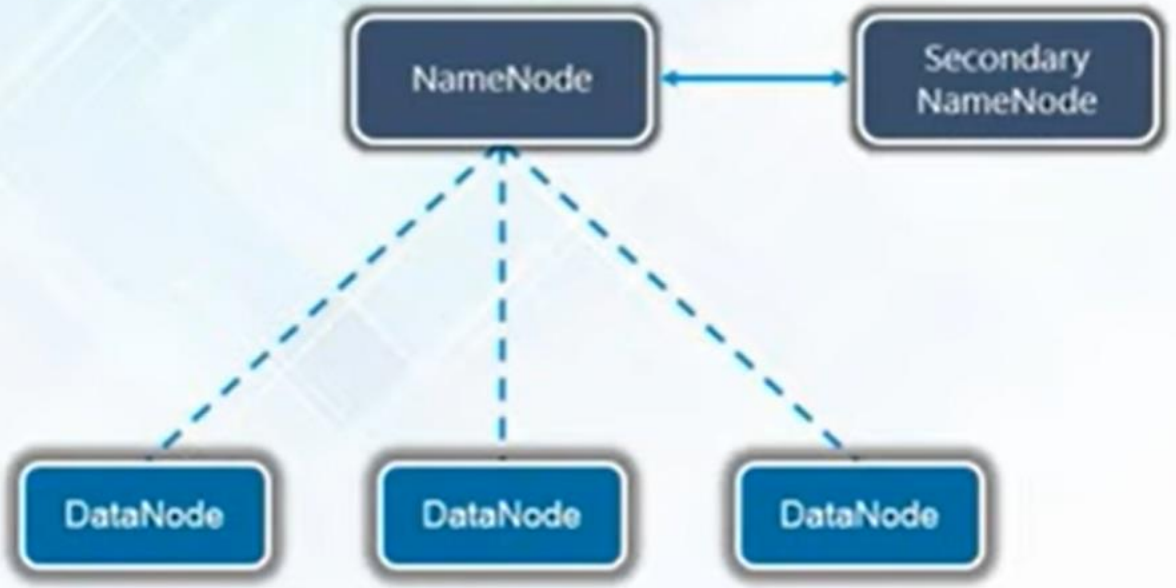


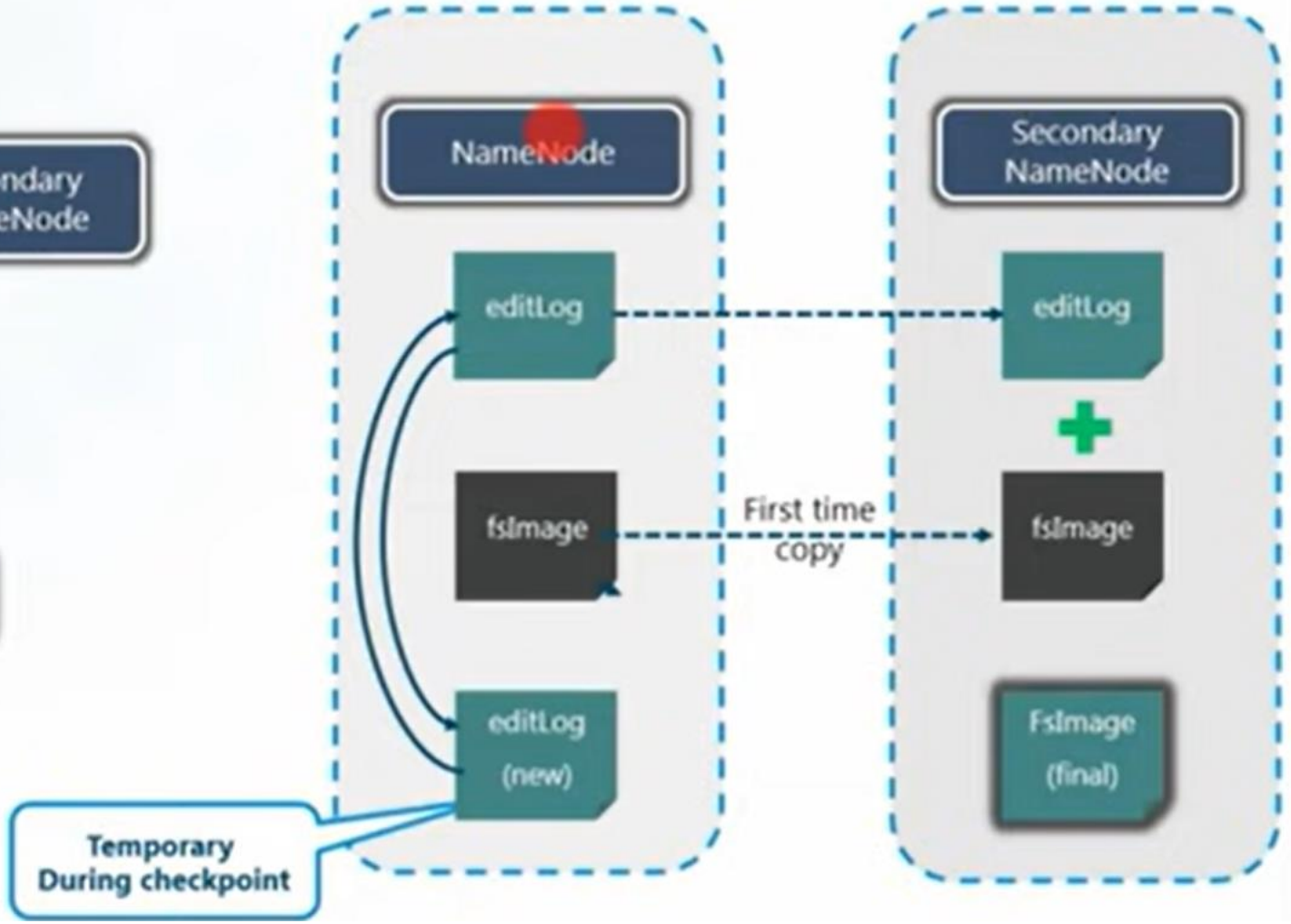
Secondary NameNode

- Checkpointing is a process of combining edit logs with FsImage
- Allows faster Failover as we have a back up of the metadata
- Checkpointing happens periodically (default: 1 hour)

Hadoop Distributed File System

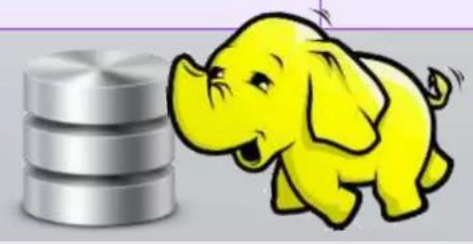
edureka!





Comparing: RDBMS vs. Hadoop

	Traditional RDBMS	Hadoop / MapReduce
Data Size	Gigabytes (Terabytes)	Petabytes (Hexabytes)
Access	Interactive and Batch	Batch – NOT Interactive
Updates	Read / Write many times	Write once, Read many times
Structure	Static Schema	Dynamic Schema
Integrity	High (ACID)	Low
Scaling	Nonlinear	Linear
Query Response Time	Can be near immediate	Has latency (due to batch processing)



Advantages of HDFS:

- It is inexpensive, immutable in nature, stores data reliably, ability to tolerate faults, scalable, block structured, can process a large amount of data simultaneously and many more.

Disadvantages of HDFS:

- It's the biggest disadvantage is that it is not fit for small quantities of data. Also, it has issues related to potential stability, restrictive and rough in nature.

Some common frameworks of Hadoop

- **Hive-** It uses HiveQl for data structuring and for writing complicated MapReduce in HDFS.
- **Drill**- It consists of user-defined functions and is used for data exploration.
- **Storm**- It allows real-time processing and streaming of data.
- **Spark**- It contains a Machine Learning Library(MLlib) for providing enhanced machine learning and is widely used for data processing. It also supports Java, Python, and Scala.
- **Pig**- It has Pig Latin, a SQL-Like language and performs data transformation of unstructured data.
- **Tez-** It reduces the complexities of Hive and Pig and helps in the running of their codes faster.

Modules of Hadoop frameworks

Hadoop framework is made up of the following modules:

1. **Hadoop MapReduce-** a MapReduce programming model for handling and processing large data.
2. **Hadoop Distributed File System-** distributed files in clusters among nodes.
3. **Hadoop YARN-** a platform which manages computing resources.
4. **Hadoop Common-** it contains packages and libraries which are used for other modules.

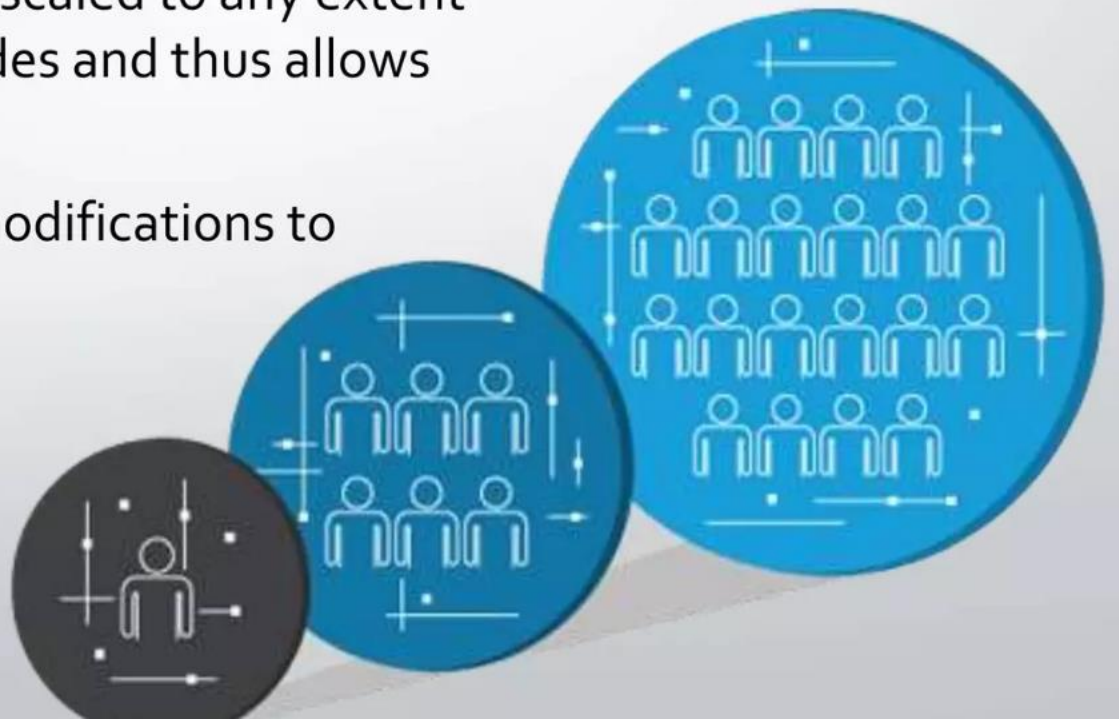
Features of 'Hadoop'

Suitable for Big Data Analysis

- As Big Data tends to be distributed and unstructured in nature, HADOOP clusters are best suited for analysis of Big Data.
- Since it is processing logic (not the actual data) that flows to the computing nodes, less network bandwidth is consumed.
- This concept is called as **data locality concept** which helps increase the efficiency of Hadoop based applications.

Scalability

- HADOOP clusters can easily be scaled to any extent by adding additional cluster nodes and thus allows for the growth of Big Data.
- Also, scaling does not require modifications to application logic.



- HADOOP ecosystem has a provision to replicate the input data on to other cluster nodes.
- That way, in the event of a cluster node failure, data processing can still proceed by using data stored on another cluster node.

WHY hadoop

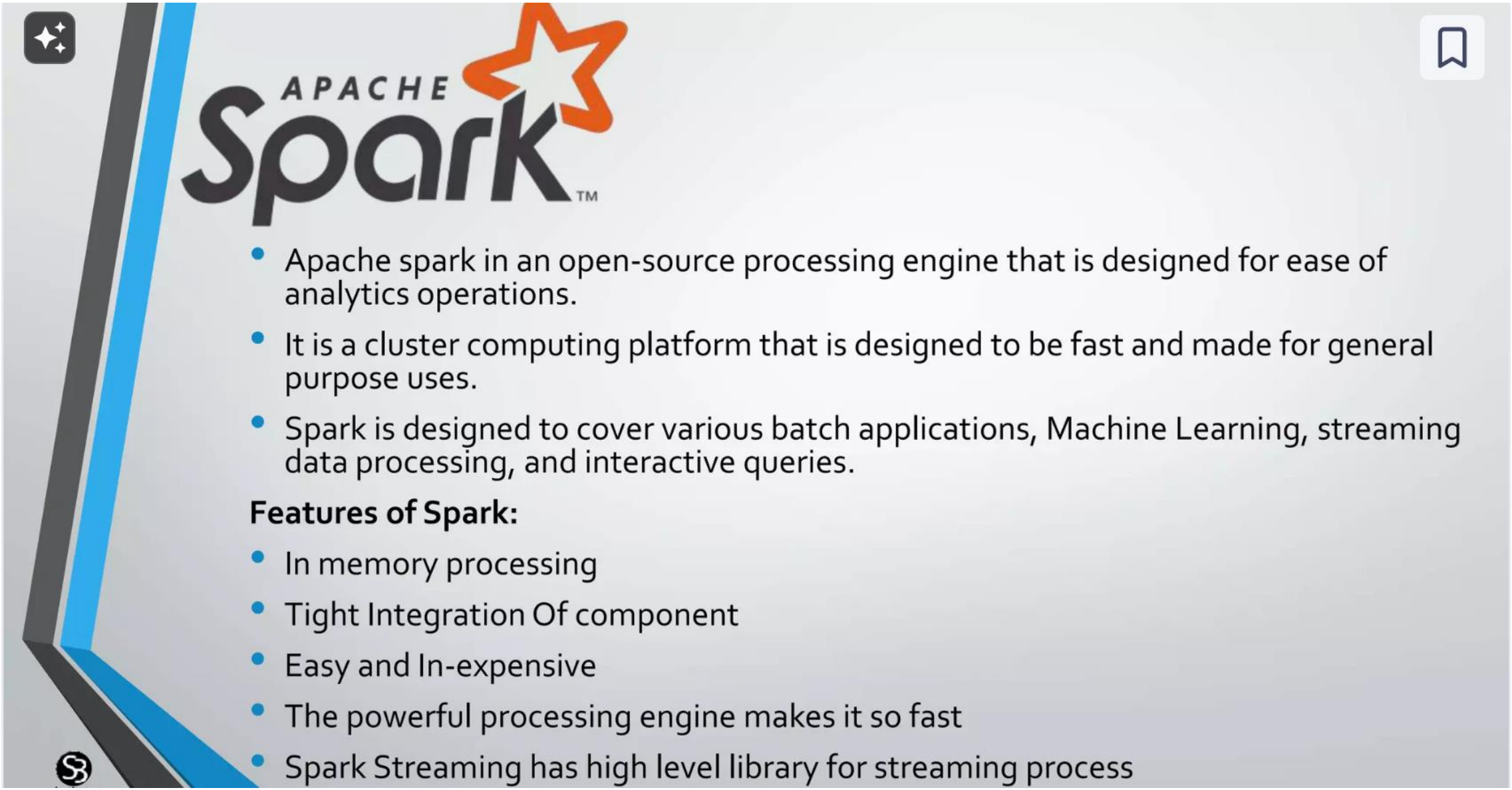
Parallel data processing

Suited for particular types of big data problems

Scales to Petabytes or more easily

Hadoop Analytics Tools

- There is a wide range of analytical tools available in the market that help Hadoop deal with the astronomical size data efficiently.
- Let us discuss some of the most famous and widely used tools one by one. Below are the top 10 Hadoop analytics tools for big data.


APACHE
Spark™
Apache spark in an open-source processing engine that is designed for ease of analytics operations.
It is a cluster computing platform that is designed to be fast and made for general purpose uses.
Spark is designed to cover various batch applications, Machine Learning, streaming data processing, and interactive queries.
Features of Spark:
In memory processing
Tight Integration Of component
Easy and In-expensive
The powerful processing engine makes it so fast
Spark Streaming has high level library for streaming process

- MapReduce is just like an Algorithm or a data structure that is based on the YARN framework.
- The primary feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster, which Makes Hadoop working so fast Because when we are dealing with Big Data, serial processing is no more of any use.

Features of Map-Reduce:

- Scalable
- Fault Tolerance
- Paraller Processing
- Tunable Replication
- Load Balancing

APACHE HIVE

- Apache Hive is a Data warehousing tool that is built on top of the Hadoop, and Data Warehousing is nothing but storing the data at a fixed location generated from various sources.
- Hive is one of the best tools used for data analysis on Hadoop.
- The one who is having knowledge of SQL can comfortably use Apache Hive.
- The query language of high is known as HQL or HIVEQL.

Features of Hive:

- Queries are similar to SQL queries.
- Hive has different storage type HBase, ORC, Plain text, etc.
- Hive has in-built function for data-mining and other works.
- Hive operates on compressed data that is present inside Hadoop Ecosystem.

Apache Impala

- Apache Impala is an open-source SQL engine designed for Hadoop.
- Impala overcomes the speed-related issue in Apache Hive with its faster-processing speed.
- Apache Impala uses similar kinds of SQL syntax, ODBC driver, and user interface as that of Apache Hive.
- Apache Impala can easily be integrated with Hadoop for data analytics purposes.

Features of Impala:

- Easy-Integration
- Scalability
- Security
- In Memory data processing

- The name *Mahout* is taken from the Hindi word **Mahavat** which means the elephant rider.
- Apache Mahout runs the algorithm on the top of Hadoop, so it is named Mahout.
- Mahout is mainly used for implementing various Machine Learning algorithms on our Hadoop like classification, Collaborative filtering, Recommendation.
- Apache Mahout can implement the Machine algorithms without integration on Hadoop.

Features of Mahout:

- Used for Machine Learning Application
- Mahout has Vector and Matrix libraries

Apache Pig

- This Pig was Initially developed by Yahoo to get ease in programming.
- Apache Pig has the capability to process an extensive dataset as it works on top of the Hadoop.
- Apache pig is used for analyzing more massive datasets by representing them as dataflow.
- Apache Pig also raises the level of abstraction for processing enormous datasets.
- Pig Latin is the scripting language that the developer uses for working on the Pig framework that runs on Pig runtime.

Features of Pig:

- Easy To Programme
- Rich set of operators
- Ability to handle various kind of data
- Extensibility

- HBase is nothing but a non-relational, NoSQL distributed, and column-oriented database. HBase consists of various tables where each table has multiple numbers of data rows.
- These rows will have multiple numbers of column family's, and this column family will have columns that contain key-value pairs.
- HBase works on the top of HDFS(Hadoop Distributed File System).
- We use HBase for searching small size data from the more massive datasets.

Features of HBase:

- HBase has Linear and Modular Scalability
- JAVA API can easily be used for client access
- Block cache for real time data queries

APACHE

- Sqoop is a command-line tool that is developed by Apache.
- The primary purpose of Apache Sqoop is to import structured data i.e., RDBMS(Relational database management System) like MySQL, SQL Server, Oracle to our HDFS(Hadoop Distributed File System).
- Sqoop can also export the data from our HDFS to RDBMS.

Features of Sqoop:

- Sqoop can Import Data To Hive or HBase
- Connecting to database server
- Controlling parallelism

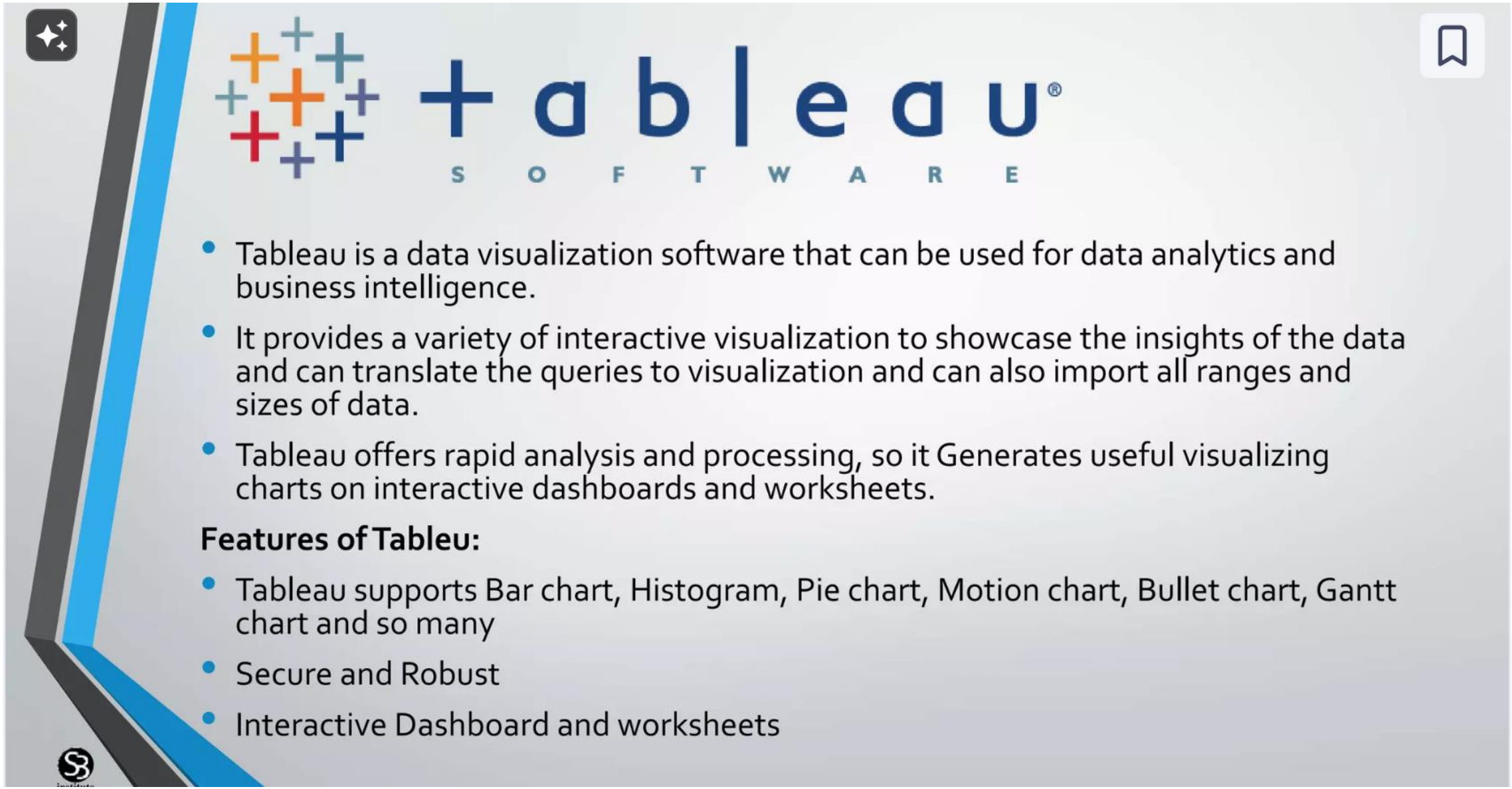
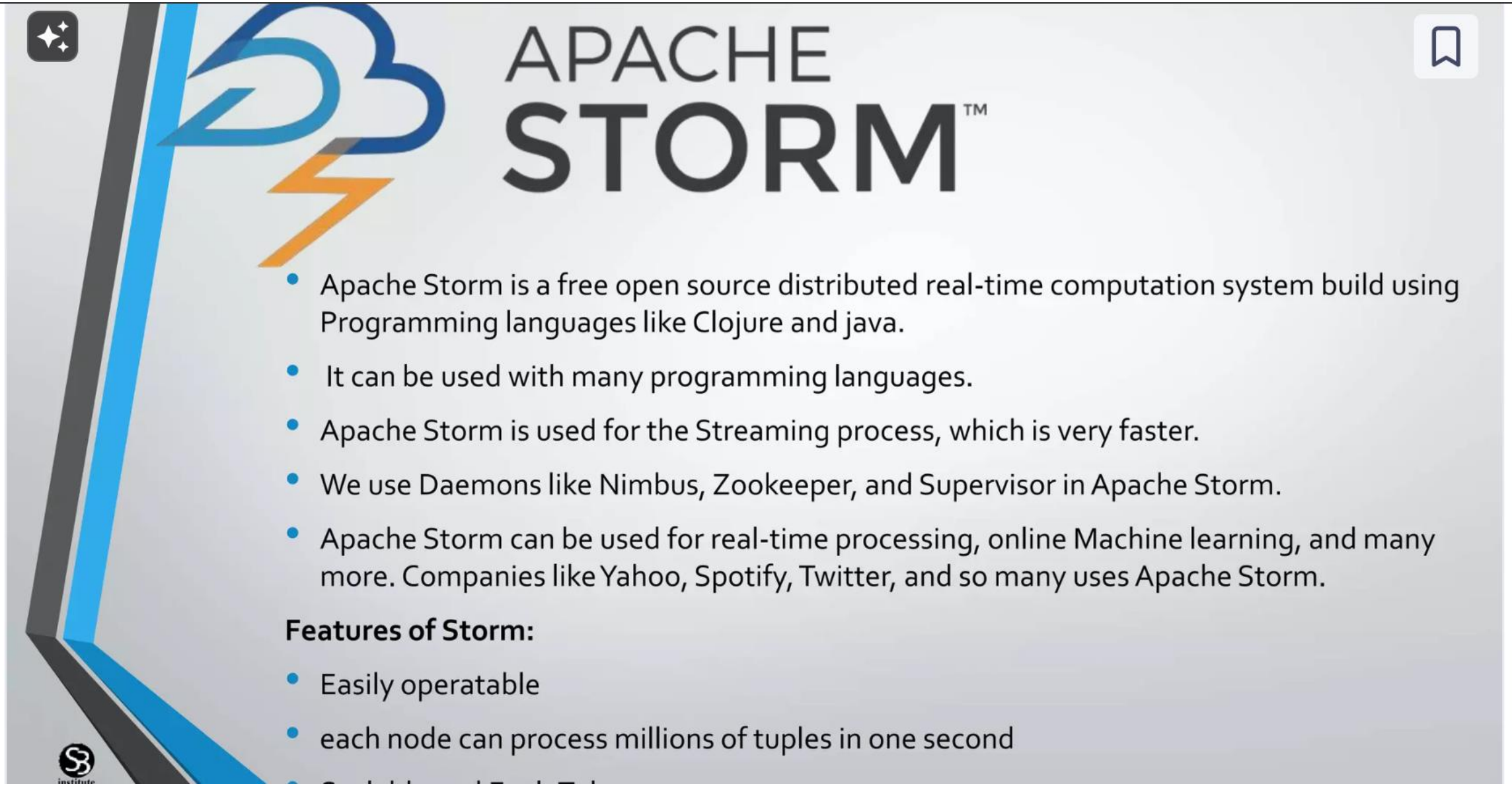


tableau
SOFTWARE
Tableau is a data visualization software that can be used for data analytics and business intelligence.
It provides a variety of interactive visualization to showcase the insights of the data and can translate the queries to visualization and can also import all ranges and sizes of data.
Tableau offers rapid analysis and processing, so it Generates useful visualizing charts on interactive dashboards and worksheets.
Features of Tableu:
Tableau supports Bar chart, Histogram, Pie chart, Motion chart, Bullet chart, Gantt chart and so many
Secure and Robust
Interactive Dashboard and worksheets

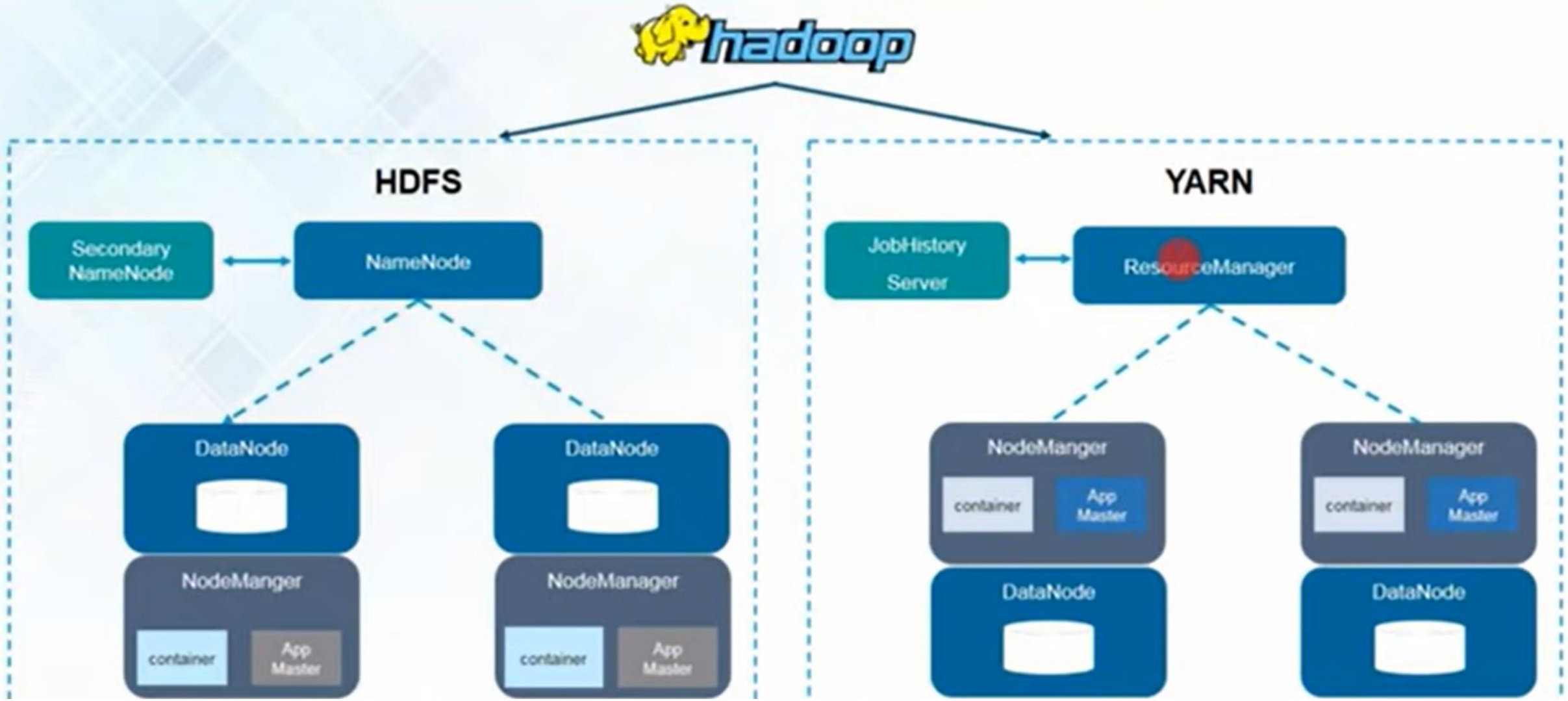

APACHE
STORM™
Apache Storm is a free open source distributed real-time computation system build using Programming languages like Clojure and java.
It can be used with many programming languages.
Apache Storm is used for the Streaming process, which is very faster.
We use Daemons like Nimbus, Zookeeper, and Supervisor in Apache Storm.
Apache Storm can be used for real-time processing, online Machine learning, and many more. Companies like Yahoo, Spotify, Twitter, and so many uses Apache Storm.
Features of Storm:
Easily operatable
each node can process millions of tuples in one second

Companies Using Hadoop

Common Hadoop Distributions

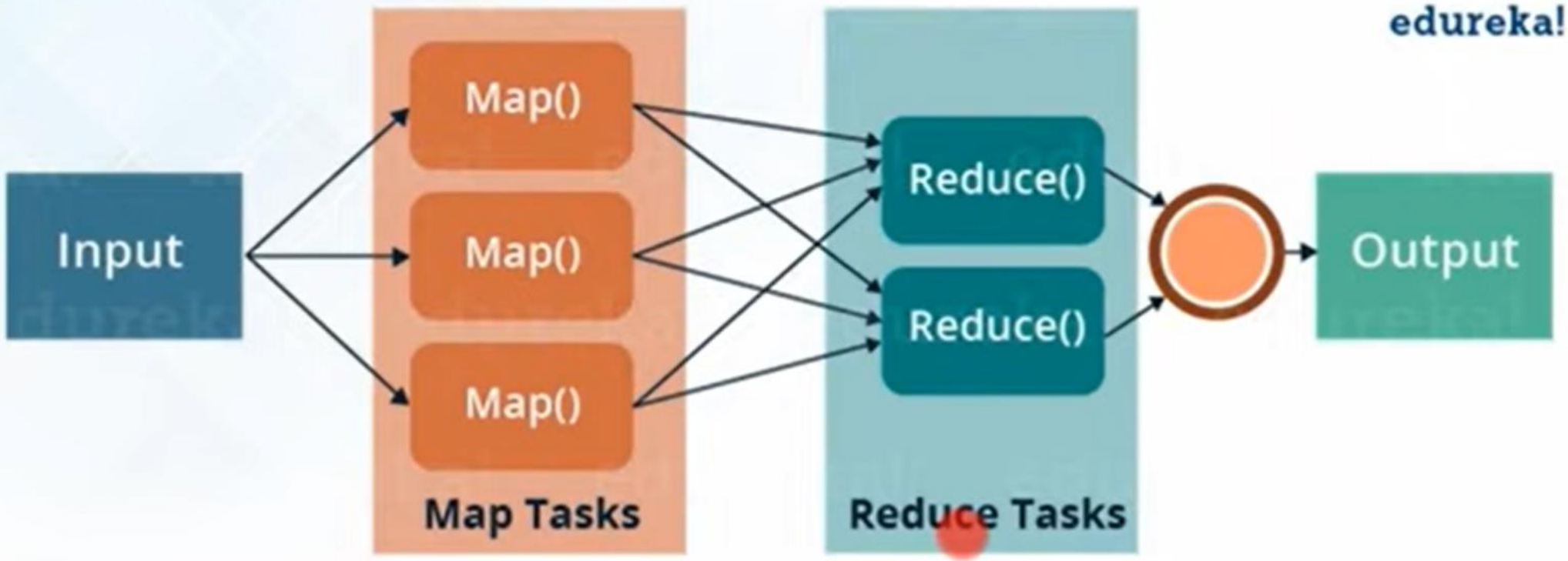
- Open Source
  - Apache
- Commercial
  - Cloudera
  - Hortonworks
  - MapR
  - AWS MapReduce
  - Microsoft Azure HDInsight (Beta)

Hadoop Architecture

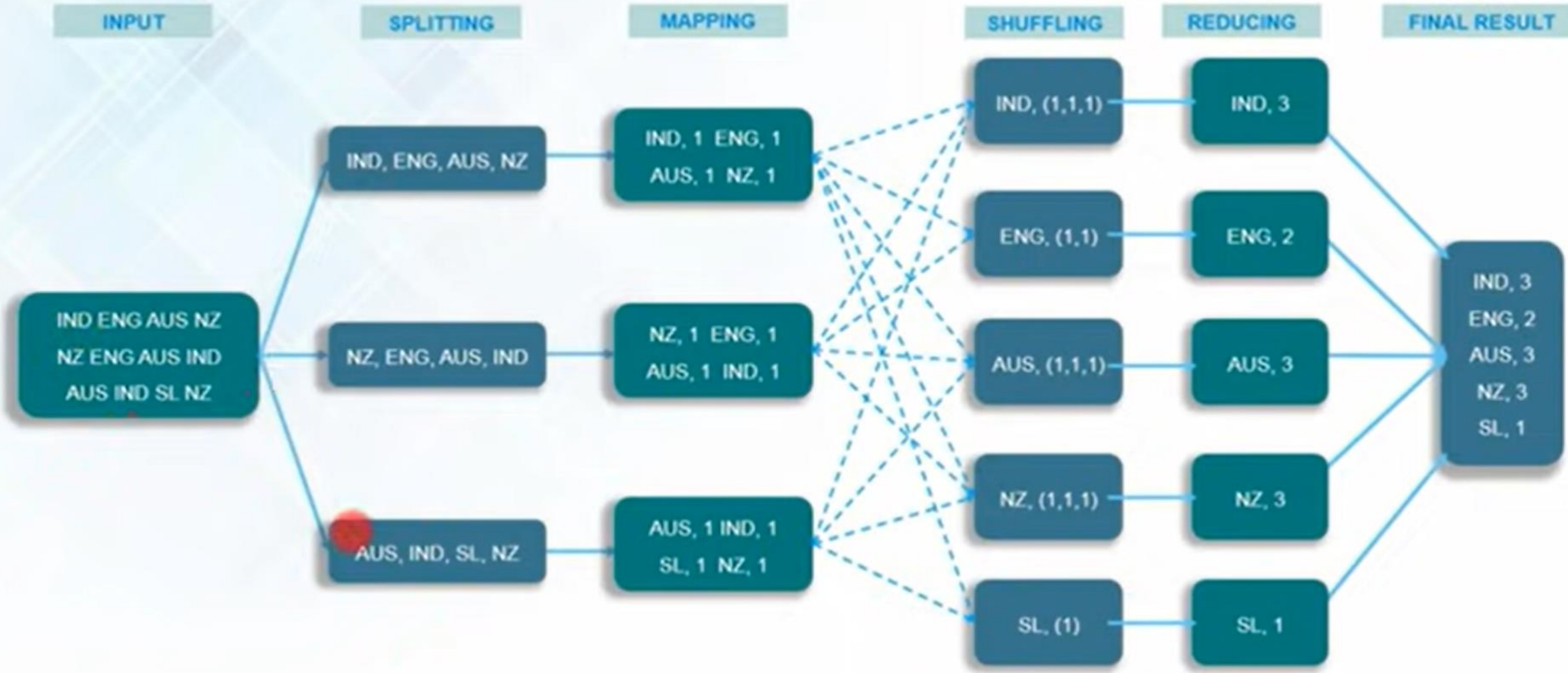


MapReduce

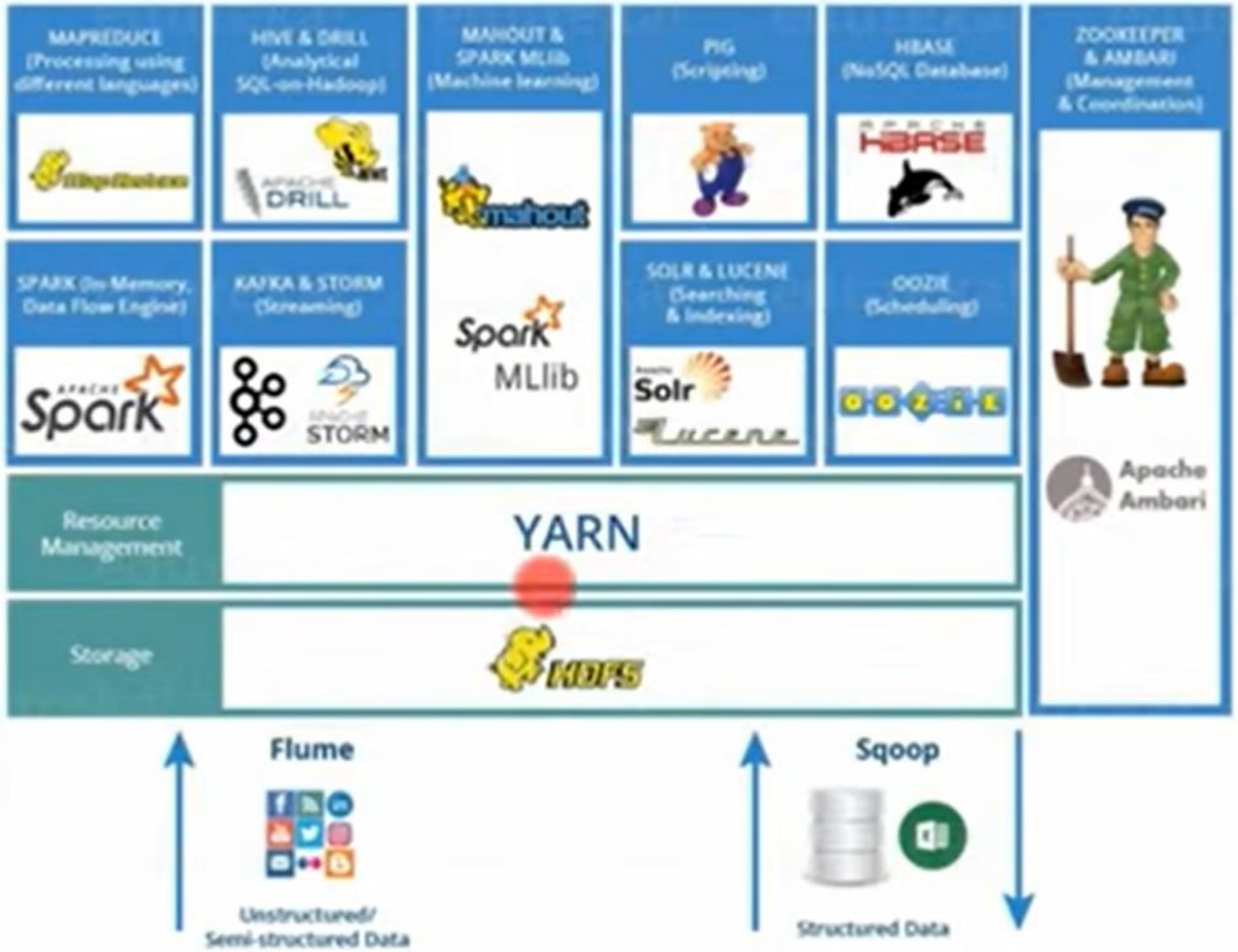
MapReduce is a software framework which helps in writing applications that processes large data sets using distributed and parallel algorithms inside Hadoop environment.



MapReduce Job Workflow



Hadoop Ecosystem



4. MapReduce

4.1 Algorithms

- **Map Phase**: Processes input data and generates intermediate key-value pairs.
- **Reduce Phase**: Aggregates intermediate data based on keys to produce final results.

4.2 Terminologies

- **Mapper**: A function or program that processes input data and generates intermediate key-value pairs.
- **Reducer**: A function or program that takes intermediate key-value pairs and aggregates them to produce final output.
- **Job**: A MapReduce program that consists of one or more map and reduce tasks.
- **Task Tracker**: Node that executes individual map or reduce tasks.

4.3 MapReduce Command and Jobs

- **Submit a Job**: Use the `hadoop jar` command to submit a MapReduce job.

```
hadoop jar myapp.jar com.example.MyDriverClass -D mapred.reduce.tasks=2 input output
```

- **Check Job Status**: Use the `hadoop job -status <job-id>` command.
- **View Job Logs**: Use the `yarn logs -applicationId <app-id>` command to check logs for debugging.

YARN in Hadoop - Key Concepts and Notes

Overview of YARN

- **YARN** (Yet Another Resource Negotiator) is the **resource management layer** of Hadoop.
- Introduced in **Hadoop 2.x** to improve scalability and efficiency.
- **Decouples** resource management and job scheduling/monitoring from the MapReduce framework.
- Enables Hadoop to run **various data-processing frameworks** (MapReduce, Spark, Tez, etc.) using the same cluster.

YARN

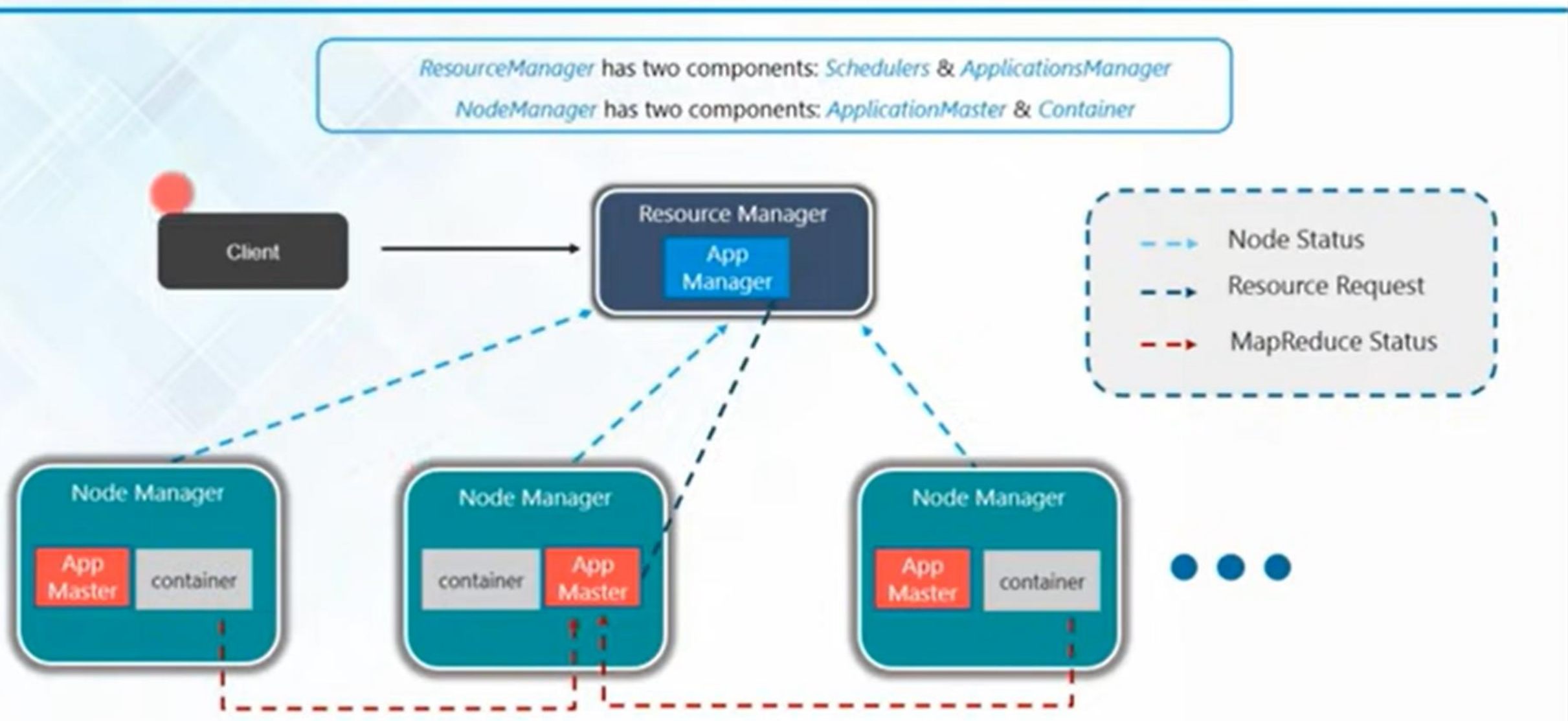
ResourceManager

- Receives the processing requests
- Passes the parts of requests to corresponding NodeManagers

NodeManagers

- Installed on every DataNode
- Responsible for execution of task on every single DataNode

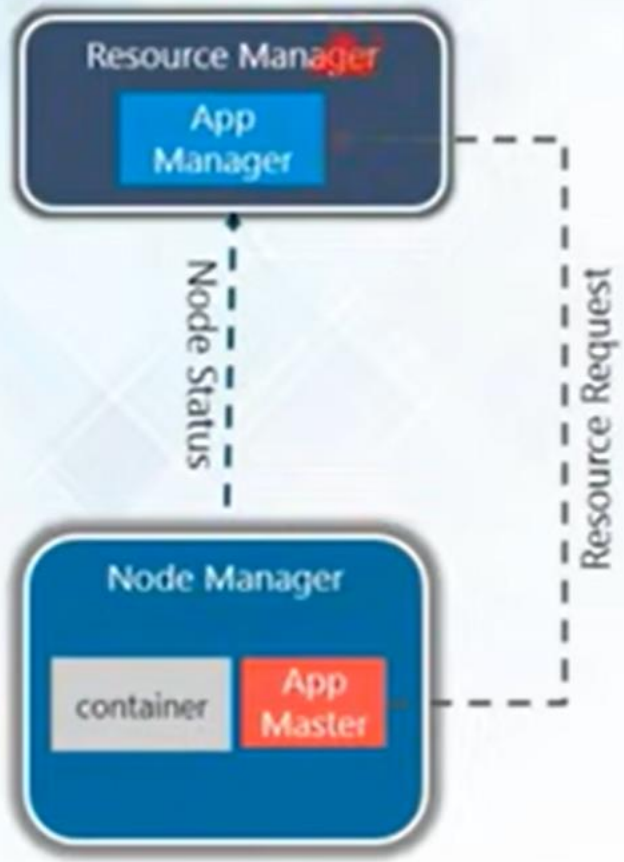
YARN Architecture



YARN Architecture

- **Resource Manager (RM)**:
  - The master node in charge of resource allocation across the cluster.
  - Consists of two components:
    - **Scheduler**: Allocates resources based on available capacity and demand (doesn't monitor tasks).
    - **ApplicationManager**: Manages submitted applications' lifecycle (negotiates resources, monitors execution).
- **Node Manager (NM)**:
  - Runs on each data node in the cluster.
  - Manages individual node's resources (memory, CPU) and container execution.
  - Reports node health and resource availability to the Resource Manager.
- **Application Master (AM)**:
  - Manages the execution of individual applications.
  - Coordinates resource requests from the Resource Manager and monitors tasks.
  - One AM is launched per application and is responsible for scheduling and monitoring containers for that application.
- **Container**:
  - A unit of resource allocation (memory, CPU) in YARN.
  - Each container runs a specific task (e.g., a Map or Reduce task in MapReduce jobs).
  - Containers are managed by the **Node Manager** on the respective node.

YARN Architecture



ApplicationsManager

- ApplicationsManager accepts the job submission
- Negotiates to containers for executing the application specific ApplicationMaster and monitoring the progress

ApplicationsMaster

- ApplicationMasters are the deamons which reside on DataNode
- Communicates to containers for execution of tasks on each DataNode

YARN Workflow

1. **Client submits a job/application** to the Resource Manager.
2. Resource Manager negotiates resources and starts the **Application Master (AM)** for the job.
3. **Application Master** requests resources from the Resource Manager for containers.
4. **Node Manager** launches containers, which execute the tasks.
5. The **Application Master** monitors the progress and completion of the job, updating the Resource Manager.

YARN Key Features

- **Multi-Tenancy**: Supports multiple applications and frameworks (MapReduce, Spark, etc.) running simultaneously.
- **Resource Utilization**: Maximizes the utilization of CPU, memory, and other resources by efficiently scheduling tasks.
- **Fault Tolerance**: Node failures are handled gracefully by reassigning tasks to other nodes.
- **Scalability**: YARN is designed to scale to thousands of nodes, making it highly suitable for large clusters.
- **Pluggable Schedulers**:
  - **FIFO Scheduler**: First In, First Out scheduling.
  - **Capacity Scheduler**: Ensures fair allocation of resources to multiple tenants based on pre-configured capacity.
  - **Fair Scheduler**: Dynamically assigns resources to ensure a fair distribution between users.

YARN Resource Management

- **Resource Allocation:**
  - YARN allocates resources in the form of **containers** (e.g., 2GB RAM, 2 vCPUs).
  - Containers are allocated dynamically based on the needs of the application.
- **Resource Request and Scheduling:**
  - Applications request resources in terms of **containers**.
  - YARN scheduler assigns resources based on availability and priority.
- **Resource Enforcement:**
  - Ensures no application uses more resources than allocated.
  - Resource Manager continuously monitors resource usage.

Advantages of YARN

- **Improved Cluster Utilization**: Resources are allocated dynamically across various frameworks, leading to better utilization.
- **Decoupling of Resource Management**: Makes Hadoop more flexible by separating resource management from processing logic.
- **Supports Non-MapReduce Applications**: Allows other processing engines (like Apache Spark, Apache Flink) to use the cluster.
- **Fault Tolerance**: Handles node failures by rerunning tasks on other available nodes.

Challenges and Considerations

- **Resource Contention**: Ensuring fair sharing of resources across applications.
- **Cluster Management Complexity**: Requires proper configuration to avoid bottlenecks in scheduling and resource allocation.
- **Application Monitoring**: Application Masters are responsible for monitoring; failures can lead to lost progress.

Streaming in Hadoop - Key Concepts and Notes

Overview of Hadoop Streaming

- **Hadoop Streaming** is a utility provided by Hadoop to allow **MapReduce** programs to be written in any language (not just Java, the default).
- It uses **standard input (stdin)** and **standard output (stdout)** to interact with the external programs (mapper and reducer).
- Particularly useful for integrating languages like Python, Ruby, Perl, or any shell scripts with Hadoop's MapReduce framework.

Workflow of Hadoop Streaming

- **Mapper**:
  - Receives input as lines of text from stdin.
  - Processes each line and outputs key-value pairs to stdout.
- **Reducer**:
  - Receives key-value pairs sorted and grouped by key from the framework.
  - Outputs the final result of the reduction to stdout.
- **Input/Output**:
  - Input data is provided via HDFS (Hadoop Distributed File System).
  - Mapper and Reducer read and write to stdin and stdout respectively.

Hadoop Streaming Command

- Basic syntax:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
  -input <input_path> \
  -output <output_path> \
  -mapper <mapper_script> \
  -reducer <reducer_script>
```

Key Parameters

- `-input` : Specifies input data location in HDFS.
- `-output` : Specifies output directory (must not exist prior).
- `-mapper` : Specifies the executable/script for the Mapper phase.
- `-reducer` : Specifies the executable/script for the Reducer phase.
- **Optional**:
  - `-file` : Used to include the script files for Mapper/Reducer.
  - `-numReduceTasks` : Set the number of reducers (can be 0 for map-only jobs).

Example - Word Count using Python

- **Mapper** (Python):

```
import sys
for line in sys.stdin:
    for word in line.strip().split():
        print(f'{word}\t1')
```

- **Reducer** (Python):

```
import sys
from collections import defaultdict

word_count = defaultdict(int)

for line in sys.stdin:
    word, count = line.strip().split('\t')
    word_count[word] += int(count)

for word, count in word_count.items():
    print(f'{word}\t{count}')
```

- **Command to Run:**

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
  -input /input/data \
  -output /output/data \
  -mapper "python mapper.py" \
  -reducer "python reducer.py" \
  -file mapper.py \
  -file reducer.py
```

Advantages of Hadoop Streaming

- Flexibility to use any language for writing MapReduce programs.
- Simplified development for non-Java programmers.
- Facilitates quick prototyping of MapReduce tasks.

Common Use Cases

- **Log Processing**: Processing large log files to extract useful insights.
- **Text Processing**: Tasks like word count, sentiment analysis, etc.
- **Data Transformation**: Converting data formats, filtering, etc.

Challenges & Considerations

- **Performance**: Non-Java programs may introduce performance overhead due to extra I/O.
- **Debugging**: Errors can be harder to track as they may originate from external scripts.
- **Efficiency**: Python or other scripting languages can be slower than Java due to interpreted nature.

Optimization Tips

- Use **combiners** for intermediate aggregation when applicable.
- Set proper **number of reducers** to optimize processing time.
- Leverage **gzip or other compression formats** to reduce I/O time.