01 Hadoop Intro

Apache Hadoop, often just called Hadoop, is a powerful open-source framework built to process and store massive datasets by distributing them across clusters of affordable, commodity hardware. Its strength lies in scalability and flexibility, enabling it to work with both structured and unstructured data.

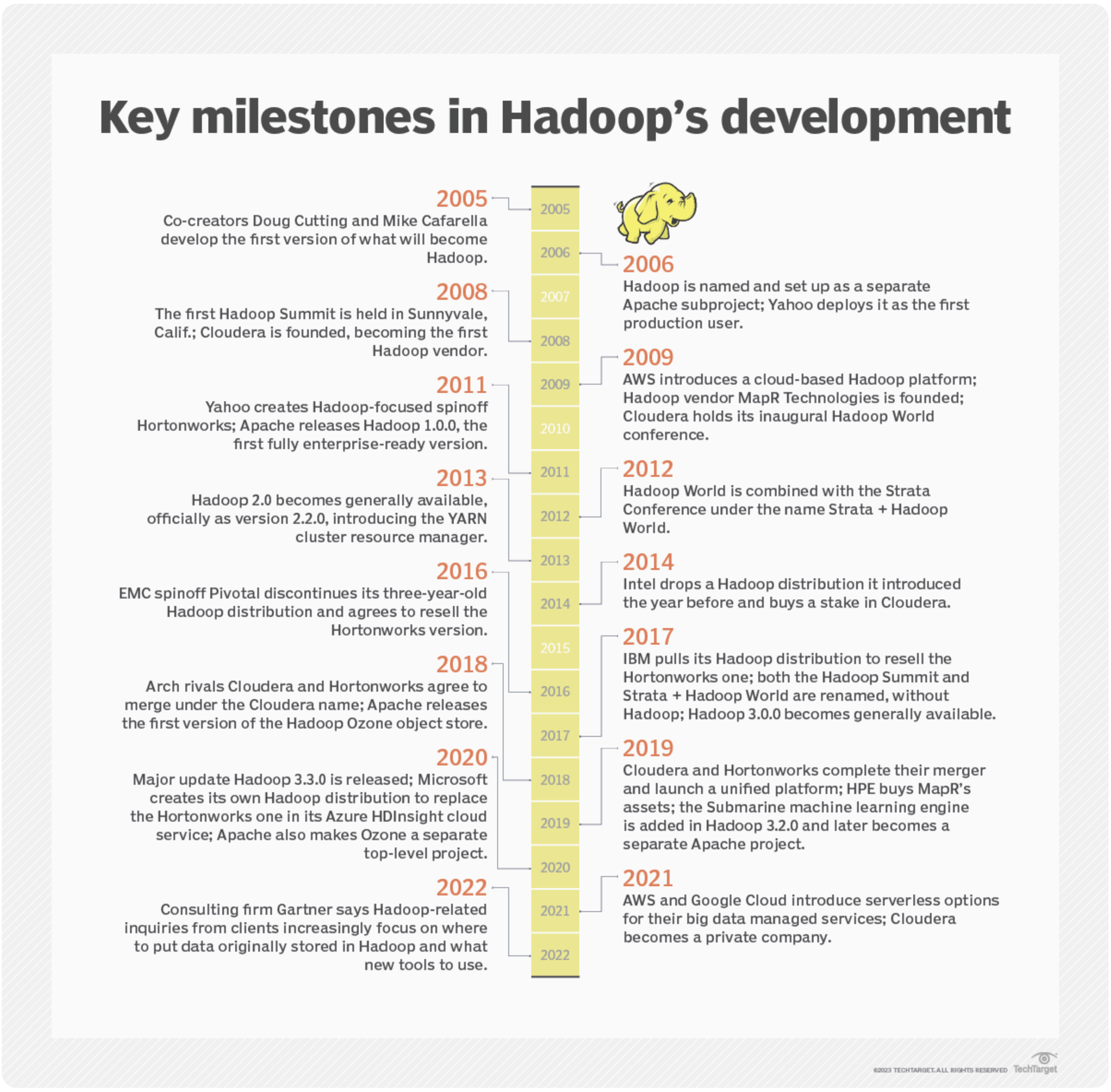
> Hadoop is an open-source distributed computing framework that enables the processing of large datasets across clusters of computers using simple programming models.

Hadoop is an opensource project comprising a broad set of tools to store and process big data in a distributed environment across a cluster of commodity servers using a simple programming model.
It is designed to scale out from single node to thousands of nodes, each offering its processing power and storage capacity.
The challenge is not about storing massive data, but it is about the processing speed and developing distributed algorithms.
Hadoop is highly distributed,
- horizontally scalable,
- fault-tolerant,
- high throughput,
- flexible, and
- cost-effective software solution for big data processing.

History of Hadoop:

For many years, users stored data in a database and processed via SQL queries for analysis.
The DWH was used to store structured historical data and process them using

OLAP for knowledge extraction.
Data in DWH (before Hadoop) is pre-processed/sampled before the analysis.
Google faced problems in processing huge data.
Google wanted to download the whole Internet and index to support search queries with MySQL database.
Google indexed 1 million pages in 1998, one billion in 2008, and over a trillion pages every day after 2010.
So, Google implemented Google File System (GFS), Google Map Reduce (GMR), and Big Table in C++ for large-scale index data processing.
Later, **Doug Cutting** and **Mike Cafarella** implemented HDFS and MapReduce in java based on GFS and GMR.
2003 – GFS white paper was released.
2004 – Nutch distributed file system was designed based on GFS.
2004 – GMR white paper was released.
2005 – Nutch MapReduce was designed based on GMR.
2006 – Nutch distributed file system + Nutch MapReduce together renamed as Hadoop (as a subproject of Nutch) and separated from the Nutch project.
2008 – Hadoop was handed over to Apache software foundation, renamed as Apache Hadoop and opensourced.
2009 – Apache Spark
2011 – Apache Storm
2012 – Yet Another Resource Manager (YARN)



Key milestones in Hadoop's development

2005 Co-creators Doug Cutting and Mike Cafarella develop the first version of what will become Hadoop.

2006 Hadoop is named and set up as a separate Apache subproject; Yahoo deploys it as the first production user.

2008 The first Hadoop Summit is held in Sunnyvale, Calif.; Cloudera is founded, becoming the first Hadoop vendor.

2009 AWS introduces a cloud-based Hadoop platform; Hadoop vendor MapR Technologies is founded; Cloudera holds its inaugural Hadoop World conference.

2011 Yahoo creates Hadoop-focused spinoff Hortonworks; Apache releases Hadoop 1.0.0, the first fully enterprise-ready version.

2012 Hadoop World is combined with the Strata Conference under the name Strata + Hadoop World.

2013 Hadoop 2.0 becomes generally available, officially as version 2.2.0, introducing the YARN cluster resource manager.

2014 Intel drops a Hadoop distribution it introduced the year before and buys a stake in Cloudera.

2016 EMC spinoff Pivotal discontinues its three-year-old Hadoop distribution and agrees to resell the Hortonworks version.

2017 IBM pulls its Hadoop distribution to resell the Hortonworks one; both the Hadoop Summit and Strata + Hadoop World are renamed, without Hadoop; Hadoop 3.0.0 becomes generally available.

2018 Arch rivals Cloudera and Hortonworks agree to merge under the Cloudera name; Apache releases the first version of the Hadoop Ozone object store.

2019 Cloudera and Hortonworks complete their merger and launch a unified platform; HPE buys MapR's assets; the Submarine machine learning engine is added in Hadoop 3.2.0 and later becomes a separate Apache project.

2020 Major update Hadoop 3.3.0 is released; Microsoft creates its own Hadoop distribution to replace the Hortonworks one in its Azure HDInsight cloud service; Apache also makes Ozone a separate top-level project.

2021 AWS and Google Cloud introduce serverless options for their big data managed services; Cloudera becomes a private company.

2022 Consulting firm Gartner says Hadoop-related inquiries from clients increasingly focus on where to put data originally stored in Hadoop and what new tools to use.

HADOOP FEATURES

Hadoop transforms a cluster of commodity servers into a service that stores and processes PBs of data reliably in a cost-effective way.
Hadoop is capable of handling volume and various problems in big data. It is also offered as a cloud service (HDInsight from Microsoft Azure, Elastic MapReduce (EMR) from Amazon, Dataproc from Google Cloud, etc.).

1. **Highly distributed**
   Hadoop works on a distributed file system and distributed computing. So, data and program can be moved around in a compute cluster.
2. **Horizontally scalable**
   Hadoop framework supports horizontal scalability for compute cluster and facilitates to write scalable algorithms using MapReduce. Hadoop supports AP in CAP theorem for a distributed system. Losing consistency in CAP theorem, shared nothing architecture, moving computation to data, no blocking and

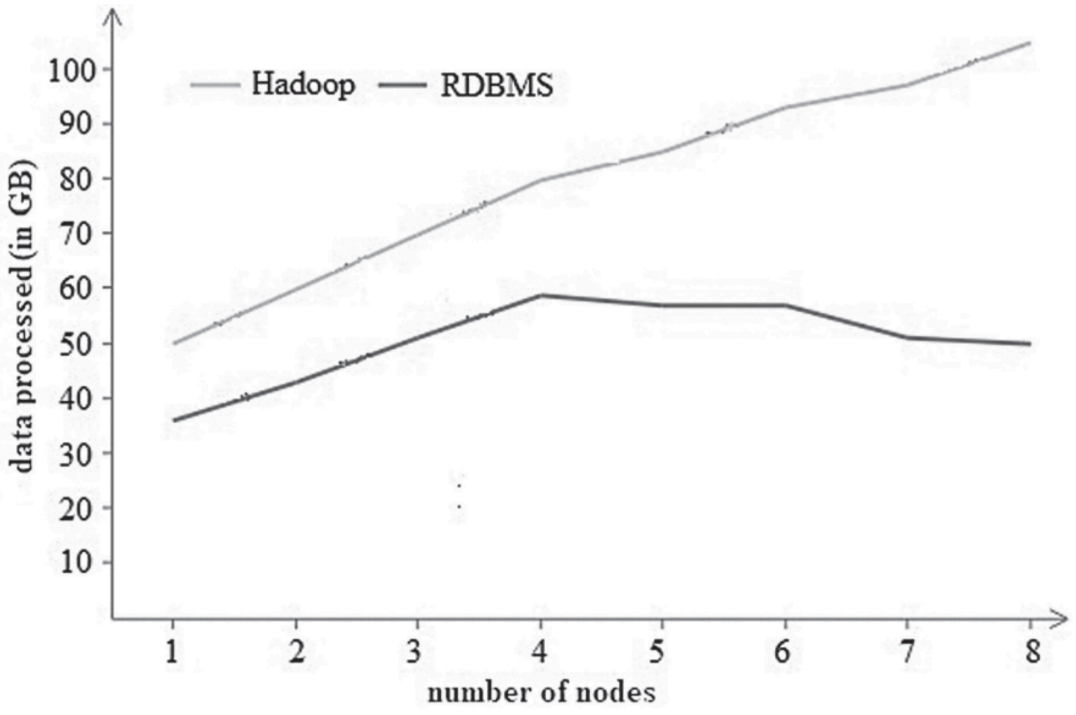
synchronization among tasks allow Hadoop more horizontally scalable. Theoretically, there is no maximum limit in scaling out. Moreover, the application program need not be rewritten according to scaling.

3. **Fault-tolerance** Cheap nodes fail, especially if you have many in the cluster. Meantime between failures for a server is three years. Meantime between failures for 1000 servers is about one day. So, the failure of servers in a cluster is more likely. Hence, data and computation loss are apparent.

> Fault-tolerance is the ability of a system to recover from failure automatically and remain functional. Hadoop self-corrects (autonomic computing) when data/process gets lost.

Hadoop has been designed to detect and handle failures.

4. **High throughput**
   Throughput is the amount of data processed per unit of time. As you increase the number of nodes in the Hadoop cluster, throughput increases linearly (which is not true in RDBMS).



5. **Flexible software**
   When a Hadoop cluster is up and running, you can dynamically add/remove nodes on the fly without disturbing/shutting down the cluster.
6. **Commodity hardware**
   Hadoop is designed to run on commodity servers. This means that you are not tied to expensive, proprietary hardware from a single vendor. You can use standardized, commonly available hardware from any vendor to build your Hadoop cluster. The commodity does not mean our laptop/desktop machines,

which are cheap and has a higher failure rate. Nonetheless, Hadoop can be deployed in our personal computers/laptop too, but running them 24/7 is not possible.

7. **Rack-aware/topology-aware**
   Hadoop is fed the network topology (arrangement of nodes in a cluster) via a configuration file, using which Hadoop can decide where to place data to minimize network flow and improve fault-tolerance.
8. **Shared nothing cluster** One server in a cluster cannot access the resources (memory, storage) of another server. Therefore, each server in Hadoop cluster autonomously functions. Finally, Hadoop application developers need not worry about networking, file IO, distributing program and data, parallelization and load balancing, task/ data/node failure (fault-tolerance). Hadoop takes care of these complexities and lets programers concentrate on writing algorithms to process the data. Hadoop completely hides system-level complexities from programmers.

Hadoop Framework

Terms

File system – is a software that controls how data is stored and retrieved from disks and other storage devices. It manages storage disks with different data access pattern like file-based (NTFS, ext4), object-based (S3), block-based (cinder, database)

Distributed system – A group of networked heterogeneous/homogeneous computers that work together as a single unit to accomplish a task is called a distributed system.

Distributed computing – is managing a set of processes across a cluster of machines/ processors, and communicate/coordinate each other by exchanging messages to finish a task.

Distributed storage – A collection of storage devices from different computers in a cluster providing single disk view is called distributed storage.

Distributed File System (DFS) – Every file system software provides a namespace (path) to files in a tree structure. Local file system gives namespace only for the disks attached to it. DFS provides a unique global (unified) namespace on distributed storage. Example: if a user executes a command at node5 to display the value of "x" stored in node10, the result is displayed in node5 itself. User does not know where "x" is stored.

There are three primary tools in Hadoop, upon which all other tools are installed.

- Hadoop Distributed File System – storage part
- MapReduce – processing part
- YARN: Resource Manager

Hadoop Distributed File System (HDFS)

HDFS is a software that allows us to store big data in a distributed environment across a cluster of low-cost, unreliable, commodity machines with streaming data access pattern. “Streaming data access pattern” means that the program reads and writes data from/to standard input and standard output.

> HDFS does not support update operation, so data in HDFS is Written Once and Read Many (WORM) times.

> A commodity hardware is low-specifications industry-grade hardware

Once the file is written to HDFS it is immutable

- HDFS has three sub-components:
    - Name Node (NN),
    - Secondary Name Node (SNN),
    - Data Node (DN).

HDFS is one of the file systems in Hadoop. There are other file systems: Local, WebHDFS, HAR, View, S3, Azure, Swift. URI scheme for these file systems is different as given below.

```
    - HDFS - hdfs://IP:port#/path
    - Local file system - fs:///path
    - Hadoop archive file - har:///path
```

MapReduce (MR)

MR is a distributed, scalable, fault-tolerant, dataparallel programming model for processing big data on a cluster of low cost, unreliable commodity machines. MR allows writing distributed, scalable jobs with little effort, unlike programming with MPI. MR jobs can be launched in different file systems: HDFS, HBase, S3, local file system, etc.

> MR is a programming model and processing engine specifically used for parallel processing of large data sets.

Hadoop 1's MR had two sub-components:

- **Job Tracker (JT)**: Managing and coordinating MR Jobs. Functions:
    1. **Job Scheduling**: It scheduled MapReduce jobs submitted by users to the cluster, assigning resources (map and reduce tasks) based on availability.
    2. **Task Assignment**: It assigned map and reduce tasks to TaskTrackers based on data locality (i.e., proximity to data) and resource availability.
    3. **Monitoring and Fault Tolerance**: It monitored the progress of map and reduce tasks, detecting failures and reassigning tasks when necessary to

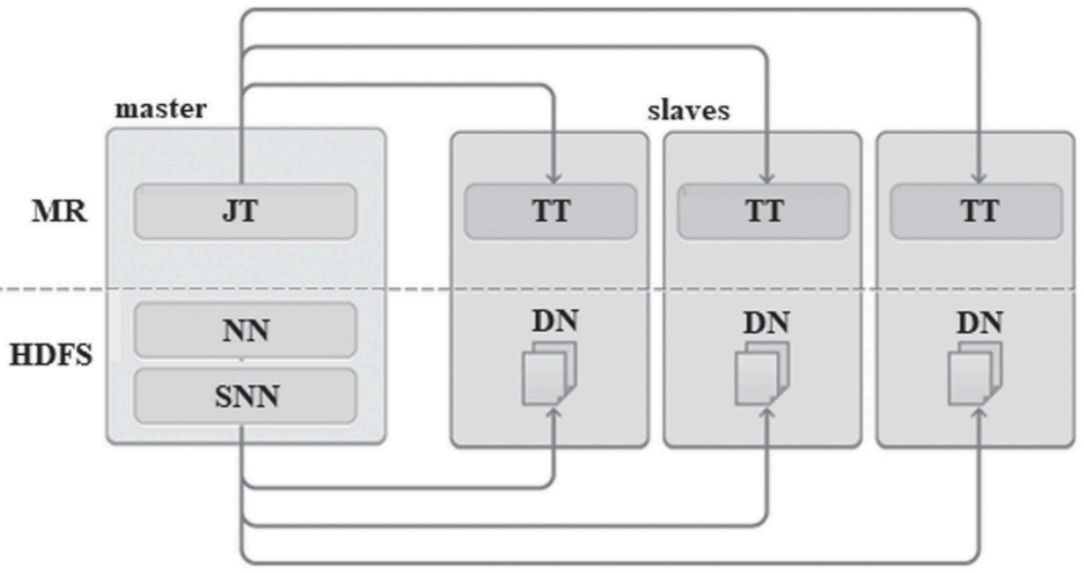
ensure fault tolerance.

- **TaskTracker (TT)**: Executing individual map and reduce tasks assigned to them by the JobTracker. Functions:
  1. **Task Execution**: They executed map and reduce tasks on the nodes where data was located, minimizing data transfer over the network and improving performance.
  2. **Heartbeat**: They periodically sent heartbeats to the JobTracker to report their status (e.g., available resources, task execution progress).
  3. **Task Retry**: In case of task failure, TaskTrackers attempted to rerun the failed task on the same node or on another node in the cluster.

MR is commonly preferred for the following applications.

- Searching, sorting, grouping.
- Simple statistics: counting, ranking.
- Complex statistics: PCA, covariance.
- Pre-processing huge data to apply machine learning algorithms.
- Classification: naïve Bayes, random forest, regression.
- Clustering: k means, hierarchical, density, bi-clustering.
- Text processing, index building, graph creation and analysis, pattern recognition, collaborative filtering, sentiment analysis.

MR and HDFS Components



YARN was introduced with Hadoop 2.x

The architecture has become more flexible and scalable. YARN decouples the resource management and job scheduling functions, allowing for multiple types of processing frameworks (beyond MapReduce) to run on the same cluster.

In the present scenario with YARN:

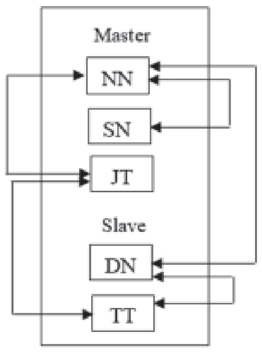
- The ResourceManager (RM) replaces the JobTracker and is responsible for

managing cluster resources and scheduling applications.
- NodeManagers (NMs) replace TaskTrackers and are responsible for managing resources (CPU, memory) on individual nodes and executing application containers.
- ApplicationMaster (AM) is responsible for negotiating resources with the ResourceManager, launching and monitoring application containers, and handling application-specific logic.

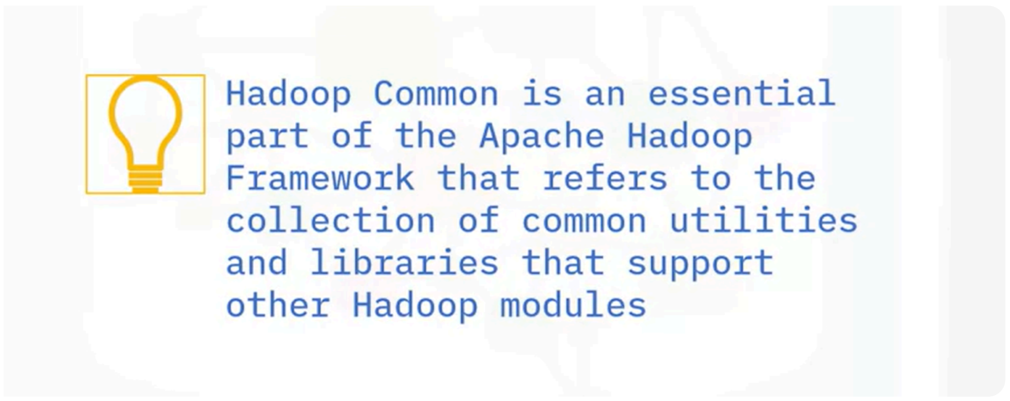
This architecture allows for better resource utilization, support for various processing frameworks (e.g., MapReduce, Spark, Tez), and improved scalability compared to the traditional MapReduce framework.
In summary, while the JobTracker and TaskTracker were key components of the original Hadoop MapReduce framework, the architecture has evolved with YARN to provide a more flexible and efficient resource management system for Big Data processing.

Hadoop Cluster

A data-center contains a set of racks stacked with a set of servers (nodes) forming a cluster with a high-speed local network. As Hadoop is based on shared-nothing architecture, each node has its storage HDDs.
HDFS and MR components communication.



Hadoop Common



Hadoop Common is an essential part of the Apache Hadoop Framework that refers to the collection of common utilities and libraries that support other Hadoop modules

HDFS

HDFS manages big data across a cluster of machines with streaming data access pattern.

It employs distributed storage to provide a single disk view and DFS to provide unique global namespace on distributed storage. It is a specially designed file system with functionalities such as distribution, fault-tolerance, replication and deployed on low-cost, unreliable machines compared to other DFSs.

> HDFS splits the file into blocks

HDFS provides file abstraction, which means that a file beyond a storage disk size is partitioned into chunks and stored across a cluster of nodes. For a regular user, the huge file is logically shown as a single file, but in reality, parts of this file are stored in different nodes in the cluster.

NAME NODE (NN)

NN is a centralized service/daemon that acts as a cluster storage manager.
Responsibilities

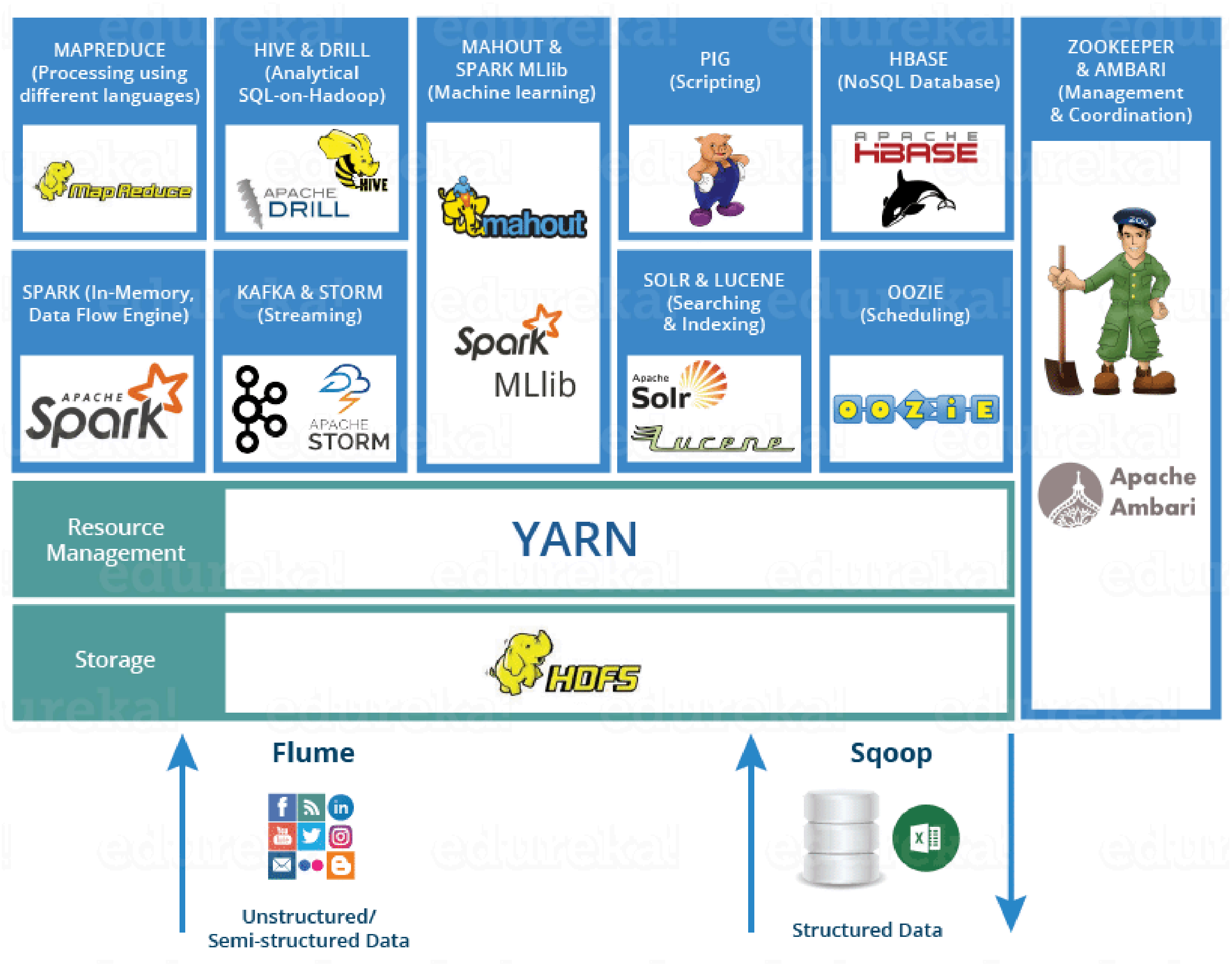
- maintaining meta-data of files and directories.
- controlling and coordinating DNs for file system operations such as create, read, write, etc.
  Clients communicate with NN in order to perform everyday file system operations. In turn, NN gives clients the location of DNs in the cluster (where the data block is available) to carry out operations. NN does authentication if it is configured.

Drawbacks of Hadoop:

1. Hadoop is not suitable for processing transactions due to its lack of random access capabilities.
2. Hadoop is ineffective when tasks cannot be done in parallel or when there are dependencies within the data. Dependencies occur when one record must be processed before another.
3. Hadoop is not good for low latency data access, which is crucial for real-time applications like trading, online gaming, and Voice over IP.
4. Hadoop struggles with processing a large number of small files

---

Hadoop Ecosystem


MAPREDUCE (Processing using different languages)
HIVE & DRILL (Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib (Machine learning)
PIG (Scripting)
HBASE (NoSQL Database)
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM (Streaming)
SOLR & LUCENE (Searching & Indexing)
OOZIE (Scheduling)
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data
Apache Ambari