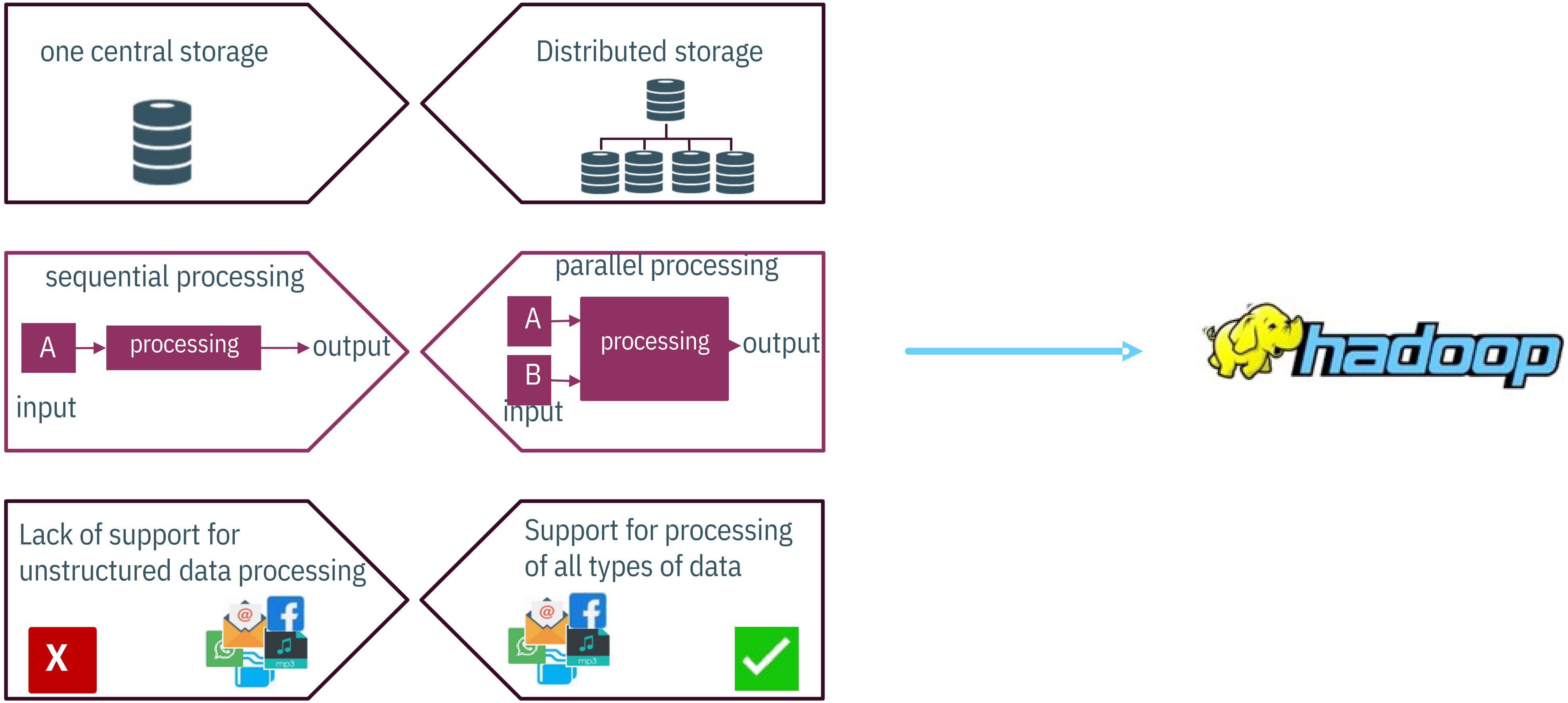
DATA TECHNOLOGIES AND SERVICES

LECTURER:

Martina Šestak, Ph.D.

martina.sestak@um.si
Office: G2-1N.10
Office hours: Tuesday, 9AM-10AM
(send email beforehand)

BIG DATA –CHALLENGES AND SOLUTIONS



one central storage

Distributed storage

sequential processing

A → processing → output

input

parallel processing

A, B → processing → output

input

Lack of support for unstructured data processing

✗

Support for processing of all types of data

✓

HADOOP

- Need for:
  - Several processors, that will process requests
  - Ability to process all types of data (structured, semi-structured, unstructured)
  - Distributed storage to store and access data

HADOOP

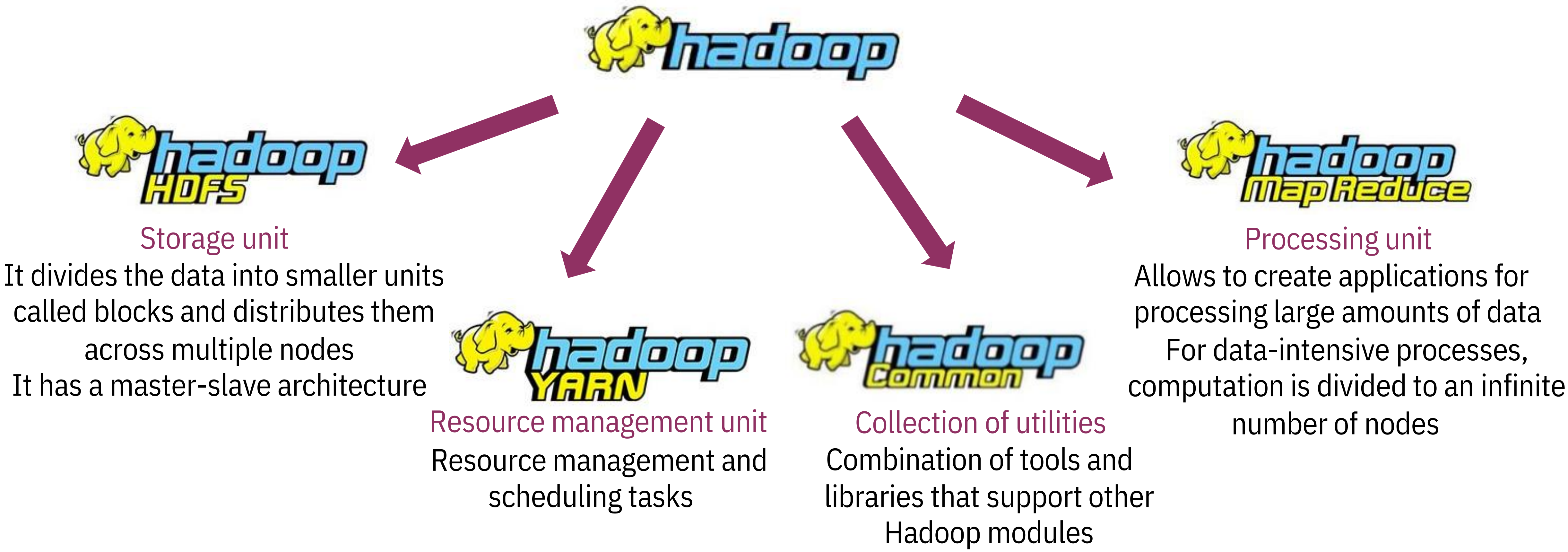
ENABLES DISTRIBUTED STORAGE OF BIG DATA AND THEIR PARALLEL PROCESSING

- An open source framework for distributed data processing based on **Java**!
- It enables reliable, scalable, distributed data processing and storage
- It enables massively parallel computing
- Allows the use of commodity hardware

WHY HADOOP?

- **Scalability:** Handle massive datasets efficiently
- **Cost-effectiveness:** Leverages commodity hardware
- **Fault-tolerance:** Built-in redundancy for reliability
- **Flexibility:** Support various data formats and processing models

HADOOP CORE COMPONENTS



Storage unit

It divides the data into smaller units called blocks and distributes them across multiple nodes

It has a master-slave architecture

Resource management unit

Resource management and scheduling tasks

Collection of utilities

Combination of tools and libraries that support other Hadoop modules

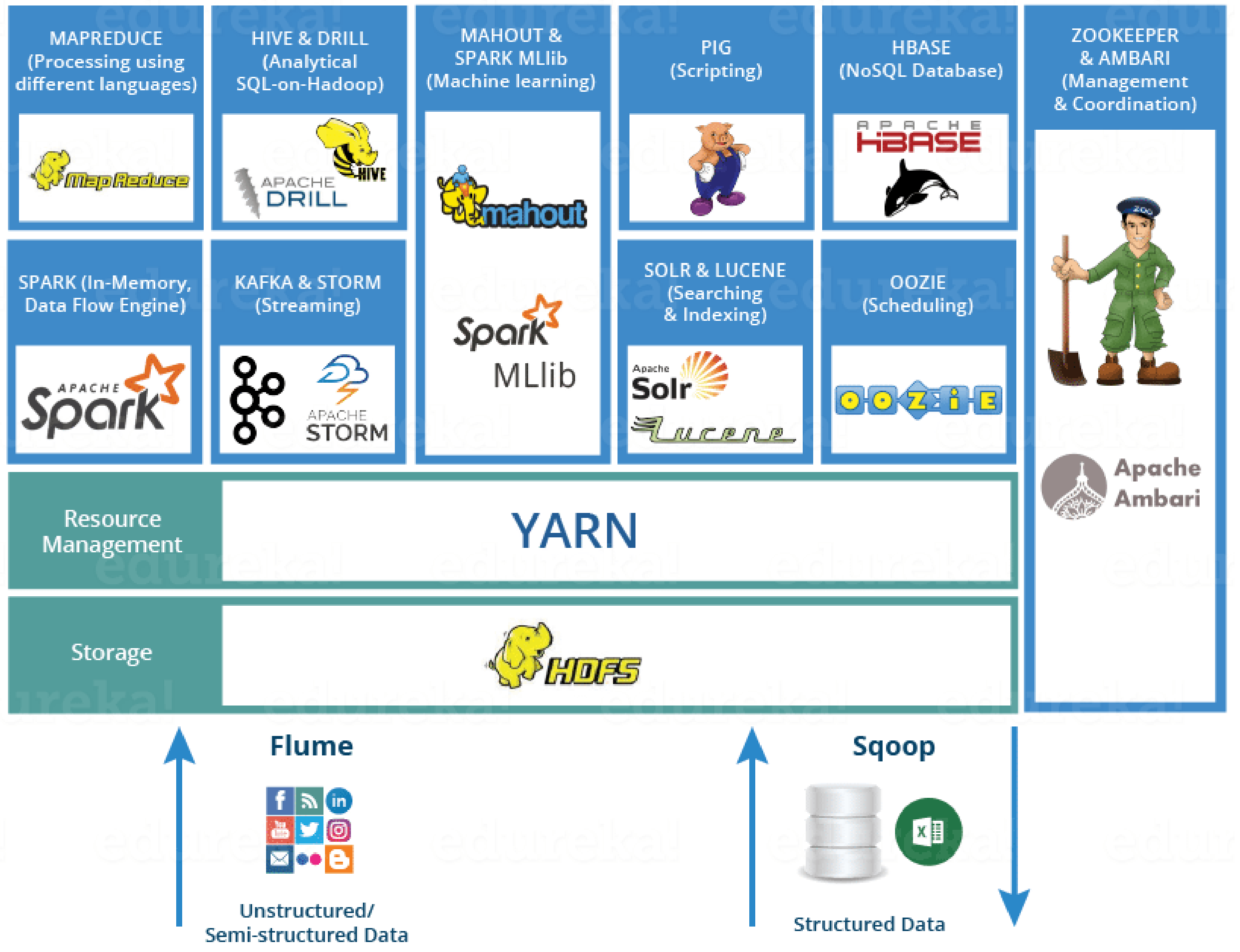
Processing unit

Allows to create applications for processing large amounts of data

For data-intensive processes, computation is divided to an infinite number of nodes

HADOOP ECOSYSTEM

- HDFS–Hadoop Distributed File System
- YARN–Yet Another Resource Negotiator



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

Source: https://www.edureka.co/blog/wp-content/uploads/2016/10/HADOOP-ECOSYSTEM-Edureka.png (accessed on October 28, 2024)

HADOOP COMPONENTS IN DETAIL

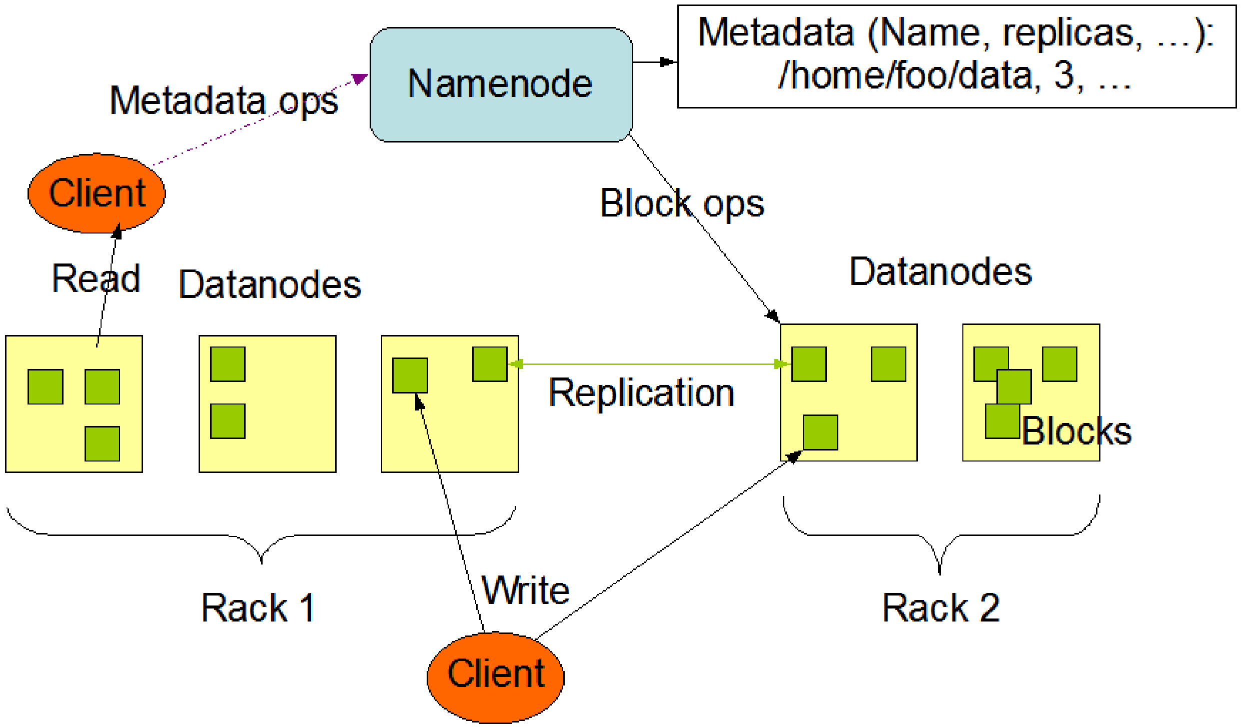
- A **distributed** file system designed to store very large files and run on commodity hardware (cheap and interchangeable)
- Each device in the cluster stores a subset of data that build the entire file system
- When needed to store more data–add more devices with multiple disks
- **Master-slave architecture**

Key features:

- Scalability – petabytes of data and more
- Fault tolerance – data replication across multiple nodes
- High throughput – optimized for large data transfers
- Simple design – easy to understand and manage

HADOOP COMPONENTS IN DETAIL

HDFS Architecture



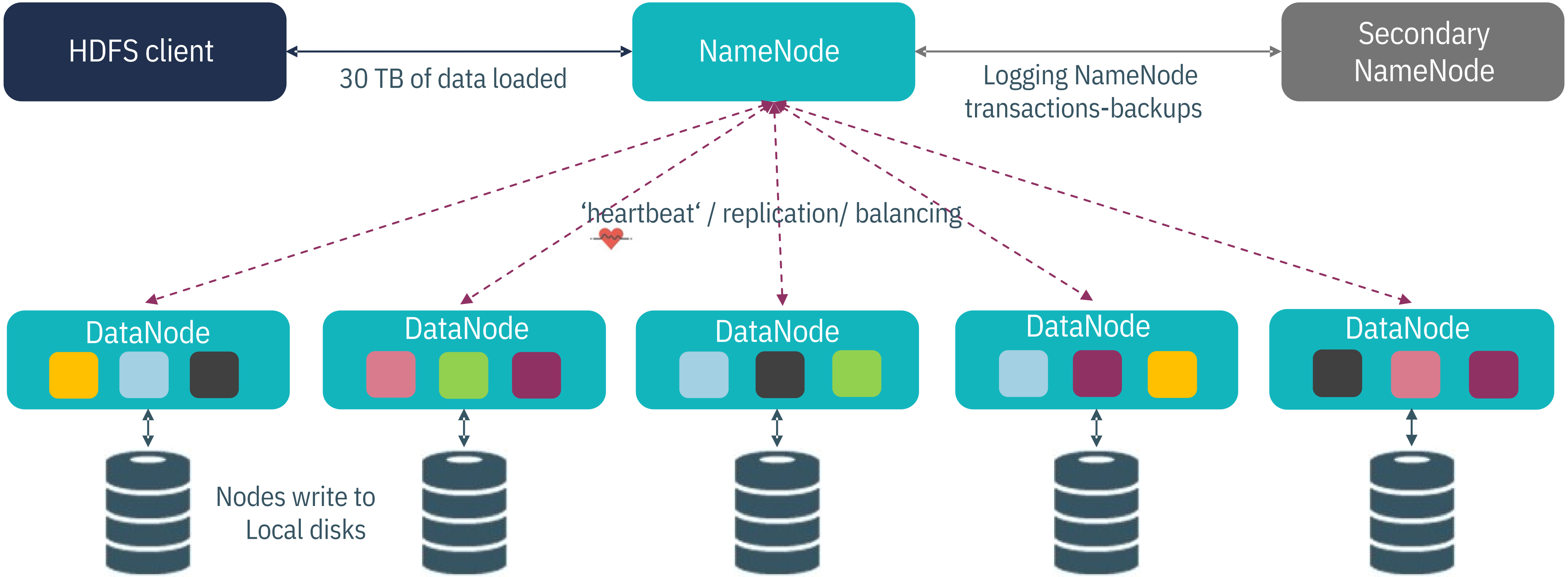
Source: https://hadoop.apache.org/docs/r1.2.1/images/hdfsarchitecture.gif (accessed on October 28, 2024)

HADOOP COMPONENTS IN DETAIL

- HDFS cluster:
  - Single NameNode (**master**), managing the FS namespace and controlling access by clients
  - More DataNodes (**slaves**), managing storage for nodes they run on, serving read and write requests from clients, and perform block creation/deletion and replication on instructions from the NN
- **Hierarchical file organization**
- Data stored in **files** organized into **directories**
  - Data operations: open, close, rename file…
- Each file stored as a sequence of **blocks** of same size (except the last block), replicated across DataNodes

HADOOP COMPONENTS IN DETAIL

Stores the metadata
Stores locations of replicated blocks
Maintains and manages data nodes



HDFS client

30 TB of data loaded

NameNode

Logging NameNode
transactions-backups

Secondary
NameNode

‘heartbeat‘ / replication/ balancing

DataNode

DataNode

DataNode

DataNode

DataNode

Nodes write to
Local disks

Stores content of the
files divided into blocks

Each block replicated at
least3x
128 MB blocks

Responsible for reads
and writes

HADOOP COMPONENTS IN DETAIL

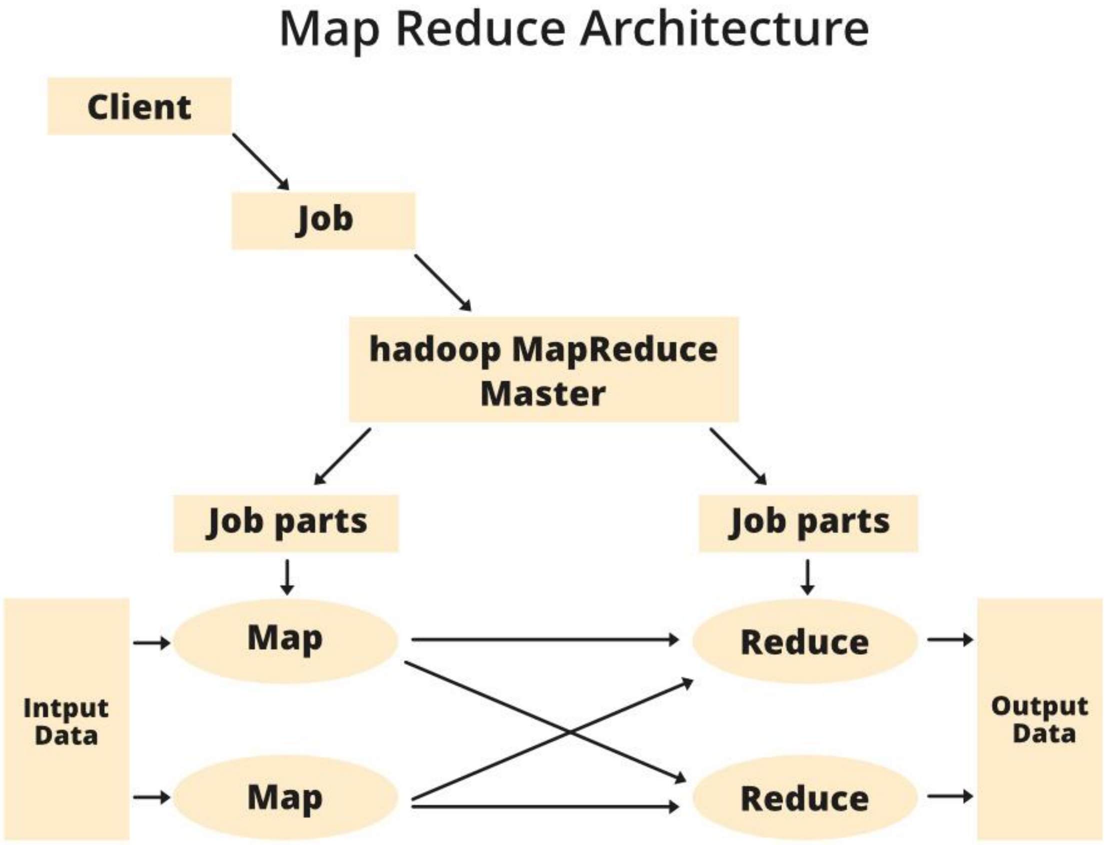
It uses parallel and distributed principles of big data processing.

- Programming model (software framework) designed to develop large-scale parallel data processing applications
- An implementation environment that supports the programming model and coordinates its implementation
- Introduced by Google
- Large amounts of data are processed using parallel and distributed principles

MapReduce job: basic unit of processing

1. Divides the data into smaller chunks
2. Sends chunks of data to ***map tasks*** run on multiple computers (clients)
   - Processing is performed on client nodes (slave node)
3. The processing result is sent to the main node (master node)
4. The results are combined by the ***reduce tasks*** and the final result is stored in the file system (e.g. HDFS)

HADOOP COMPONENTS IN DETAIL



Source: https://media.geeksforgeeks.org/wp-content/cdn-uploads/20200908123810/MapReduce-Architecture.jpg (accessed on October 28, 2024)

HADOOP COMPONENTS IN DETAIL

Breaks input data into fragments

Programmer defines map and reduce functions

- Map takes key-value pair as input and returns a set of intermediate key-value pairs as output

Combines all intermediate values with the same intermediate key into a group and passes them to the reduce function

- Reduce takes an intermediate key and combines its set of values into a smaller set consisting of zero or more elements

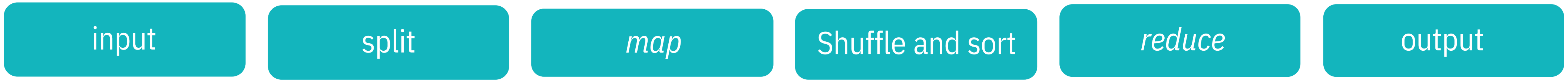
Phase **map** is performed on all input data.
Phase **reduce** uses the reduce function on all intermediate keys and associated value sets.

HADOOP COMPONENTS IN DETAIL

Breaks input data into fragments

Programmer defines map and reduce functions

- Map takes key-value pair as input and returns a set of intermediate key-value pairs as output
- Reduce takes an intermediate key and combines its set of values into a smaller set consisting of zero or more elements



input

split

map

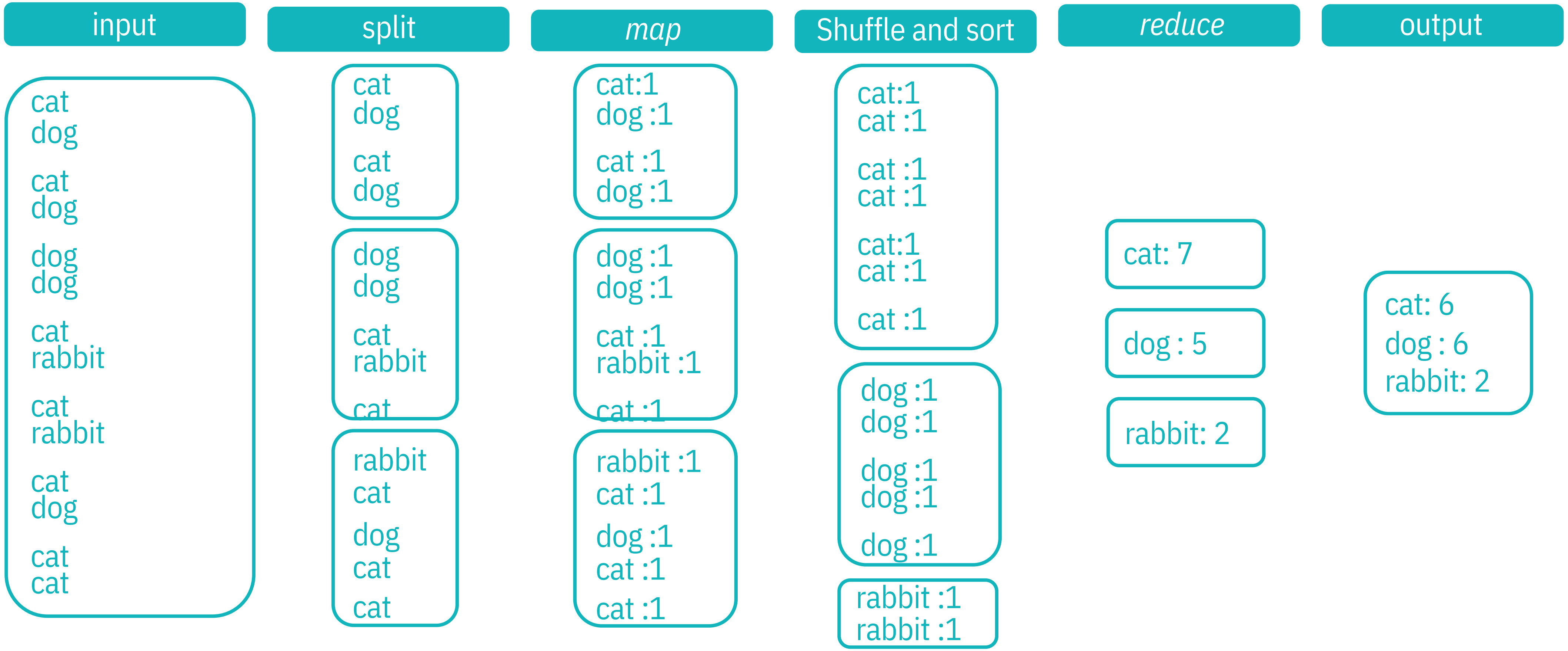
Shuffle and sort

reduce

output

HADOOP COMPONENTS IN DETAIL

Task: counting the occurrences of each animal in a given input data



input

cat
dog
cat
dog
dog
dog
cat
rabbit
cat
rabbit
cat
dog
cat
cat

split

cat
dog
cat
dog

dog
dog
cat
rabbit
cat

rabbit
cat
dog
cat
cat

map

cat:1
dog :1
cat :1
dog :1

dog :1
dog :1
cat :1
rabbit :1
cat :1

rabbit :1
cat :1
dog :1
cat :1
cat :1

Shuffle and sort

cat:1
cat :1
cat :1
cat :1
cat:1
cat :1
cat :1

dog :1
dog :1
dog :1
dog :1
dog :1

rabbit :1
rabbit :1

reduce

cat: 7

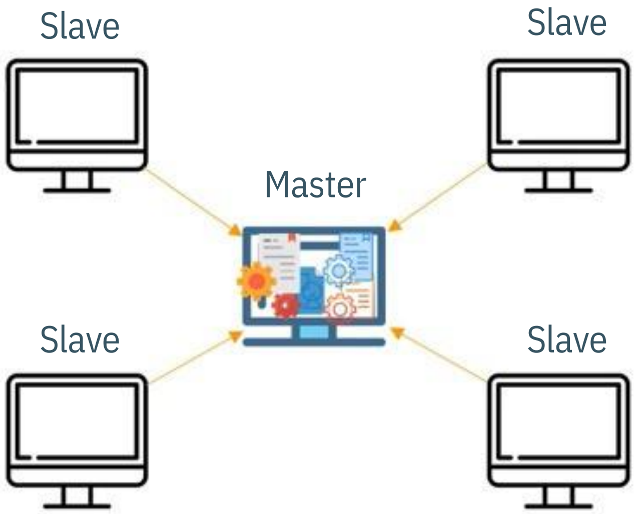
dog : 5

rabbit: 2

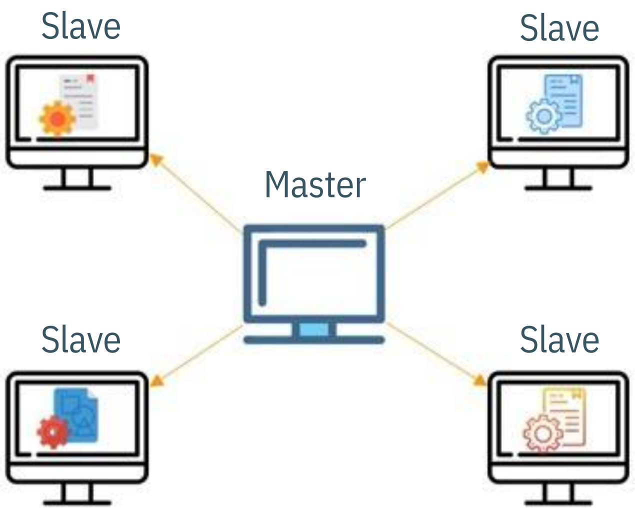
output

cat: 6
dog : 6
rabbit: 2

HADOOP COMPONENTS IN DETAIL

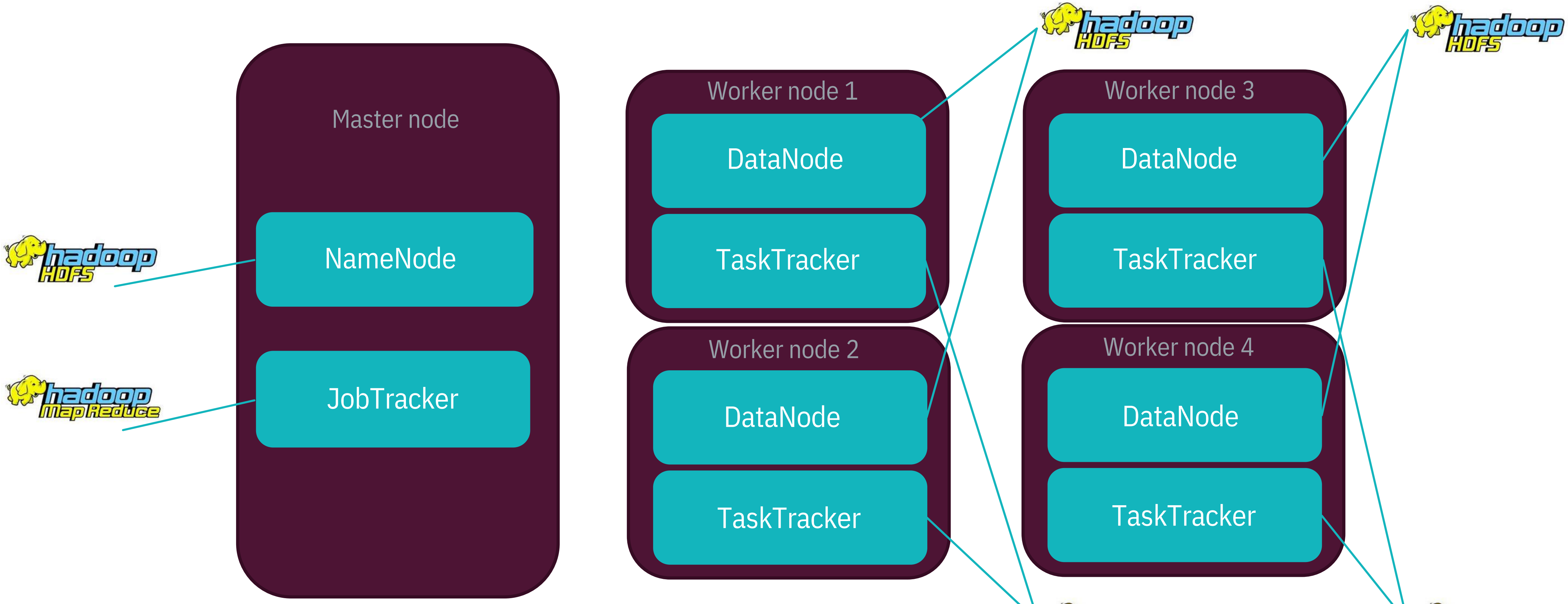


TRADITIONAL APPROACH
Data is processed at the Master node



MAPREDUCE APPROACH
Data is processed at the Slave nodes

MAPREDUCE PROCESSING ENVIRONMENT

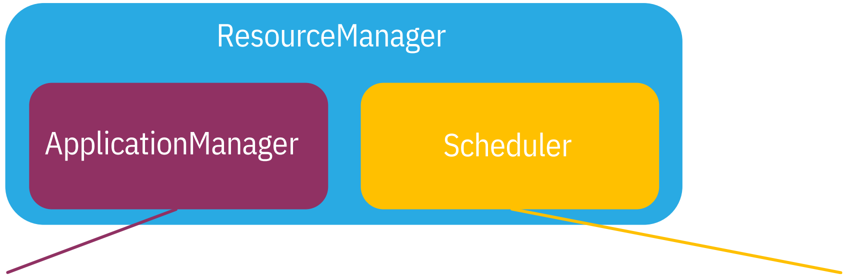


JOB TRACKER –schedules and manages MR job
TASK TRACKER –executes individual *map* and *reduce* tasks on each cluster node

HADOOP COMPONENTS IN DETAIL

YARN–Yet Another Resource Negotiator

- Hadoop version 2.0 –ResourceManager unit
- Responsible for resource allocation and management



Monitoring the application progress, application status tracking, negotiation of resources with ResourceManager

Schedules the tasks based on resource availability and application allocation

NodeManager

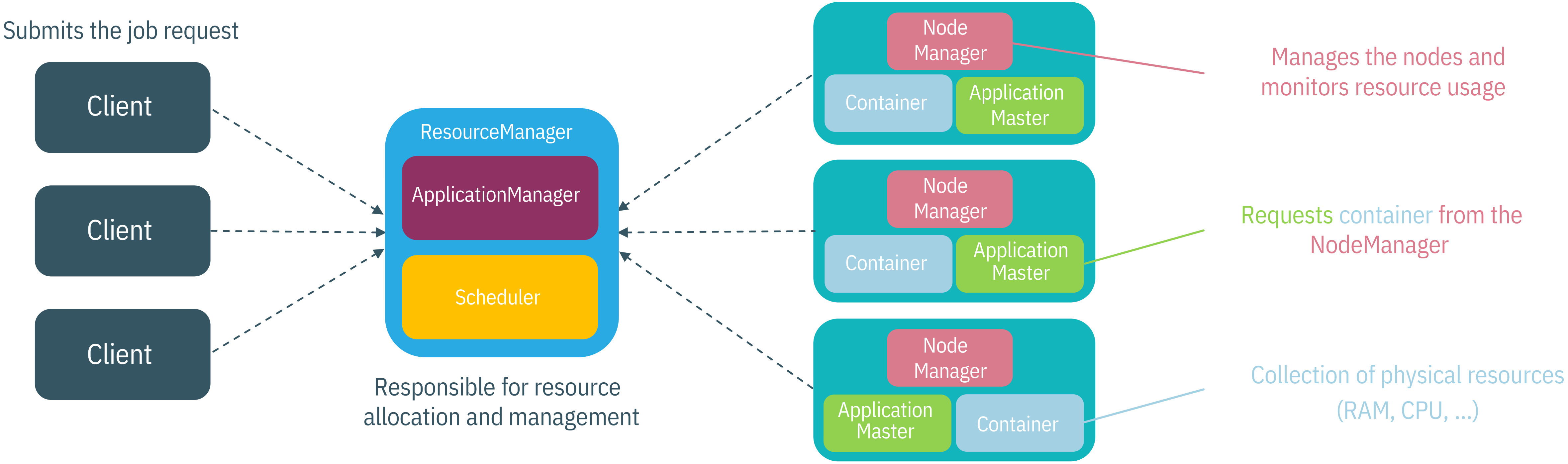
Container

ApplicationMaster

HADOOP COMPONENTS IN DETAIL

YARN–Yet Another Resource Negotiator

- Hadoop version2.0 –ResourceManager unit
- Responsible for resource allocation and management

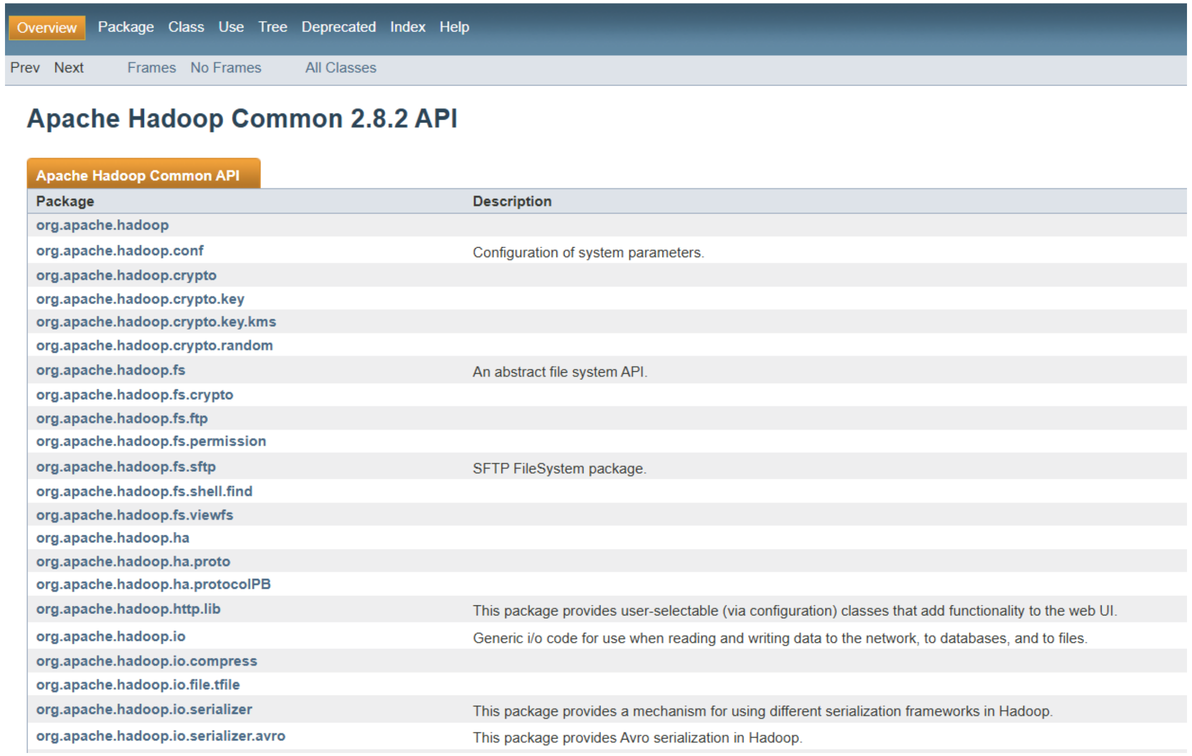


HADOOP COMPONENTS IN DETAIL

Collection of utilities

- Combination of tools and libraries that support setting up and working with Hadoop

- Open-source



Overview Package Class Use Tree Deprecated Index Help
Prev Next Frames No Frames All Classes

Apache Hadoop Common 2.8.2 API

Apache Hadoop Common API

Package	Description
org.apache.hadoop	
org.apache.hadoop.conf	Configuration of system parameters.
org.apache.hadoop.crypto	
org.apache.hadoop.crypto.key	
org.apache.hadoop.crypto.key.kms	
org.apache.hadoop.crypto.random	
org.apache.hadoop.fs	An abstract file system API.
org.apache.hadoop.fs.crypto	
org.apache.hadoop.fs.ftp	
org.apache.hadoop.fs.permission	
org.apache.hadoop.fs.sftp	SFTP FileSystem package.
org.apache.hadoop.fs.shell.find	
org.apache.hadoop.fs.viewfs	
org.apache.hadoop.ha	
org.apache.hadoop.ha.proto	
org.apache.hadoop.ha.protocolPB	
org.apache.hadoop.http.lib	This package provides user-selectable (via configuration) classes that add functionality to the web UI.
org.apache.hadoop.io	Generic i/o code for use when reading and writing data to the network, to databases, and to files.
org.apache.hadoop.io.compress	
org.apache.hadoop.io.file.tfile	
org.apache.hadoop.io.serializer	This package provides a mechanism for using different serialization frameworks in Hadoop.
org.apache.hadoop.io.serializer.avro	This package provides Avro serialization in Hadoop.

OTHER HADOOP TOOLS AND SOLUTIONS

PIG | HIVE | Presto

- Tools that ease the complexity of writing complex Java MapReduce programs

Platform for development and execution of high-level language (Pig Latin) scripts for complex ETL and data analysis jobs on Hadoop datasets

Read-only relational database that enables SQL based querying of Hadoop datasets

Framework that supports usage of distributed SQL queries for Hadoop, NoSQL and cloud storages.

NoSQL storage that provides real-time random read/write access to datasets in Hadoop.

OTHER BIG DATA TOOLS BUILT ON TOP OF HADOOP

sqoop

Extensible application for bulk transfer of data between Hadoop and structured data stores and relational databases

Spark

In-memory data processing engine which can run either over Hadoop or standalone as an alternative (successor to Hadoop itself)

Apache Zeppelin

Web-based environment that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala, … Provides built-in Spark integration

kafka

Stream processing and message brokering system

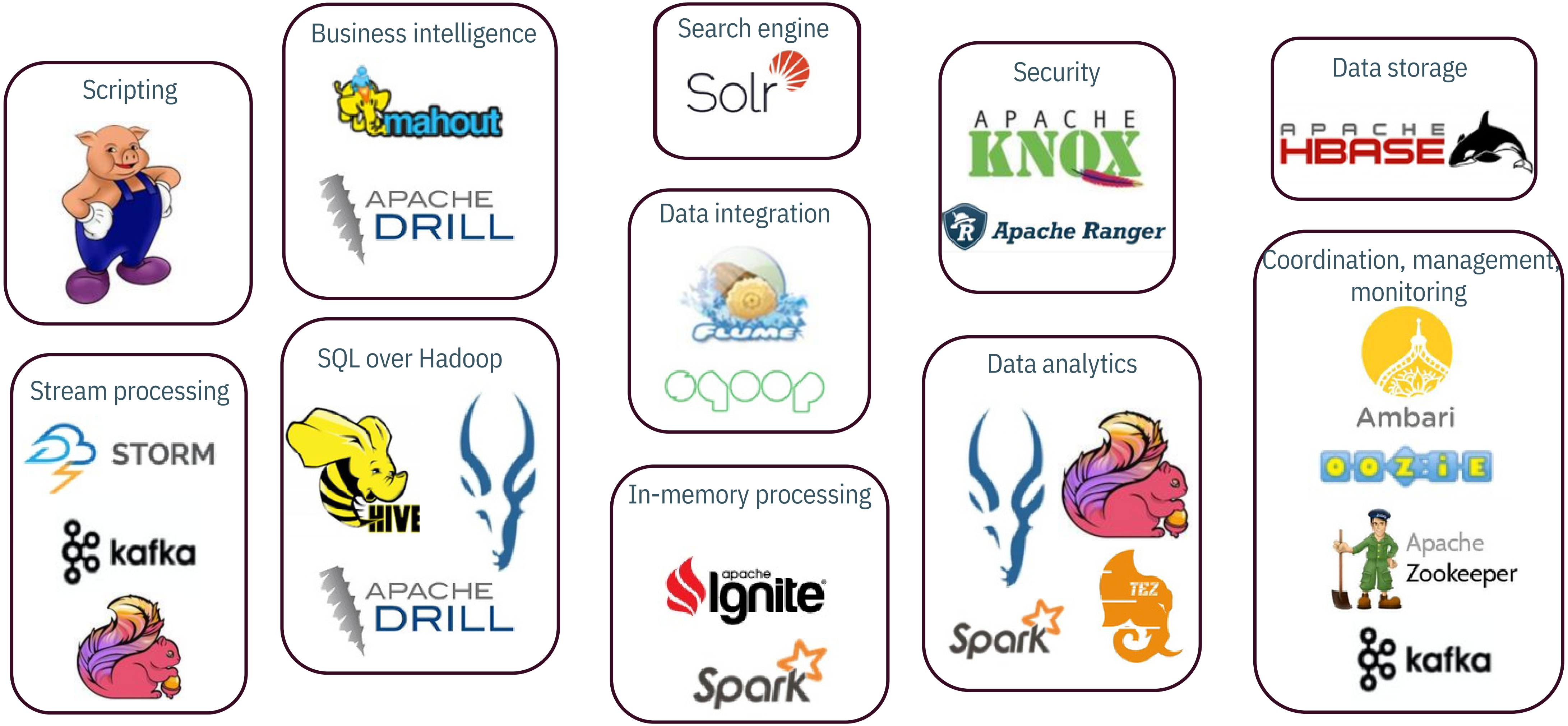
presto

Open source distributed SQL query engine for running interactive analytic queries against data sources of all sizes

apache Ignite

Horizontally scalable, fault-tolerant distributed in-memory computing platform for building real-time applications that can process large amounts of data with in-memory speed

HADOOP



Scripting

Stream processing

Business intelligence

SQL over Hadoop

Search engine

Data integration

In-memory processing

Security

Data analytics

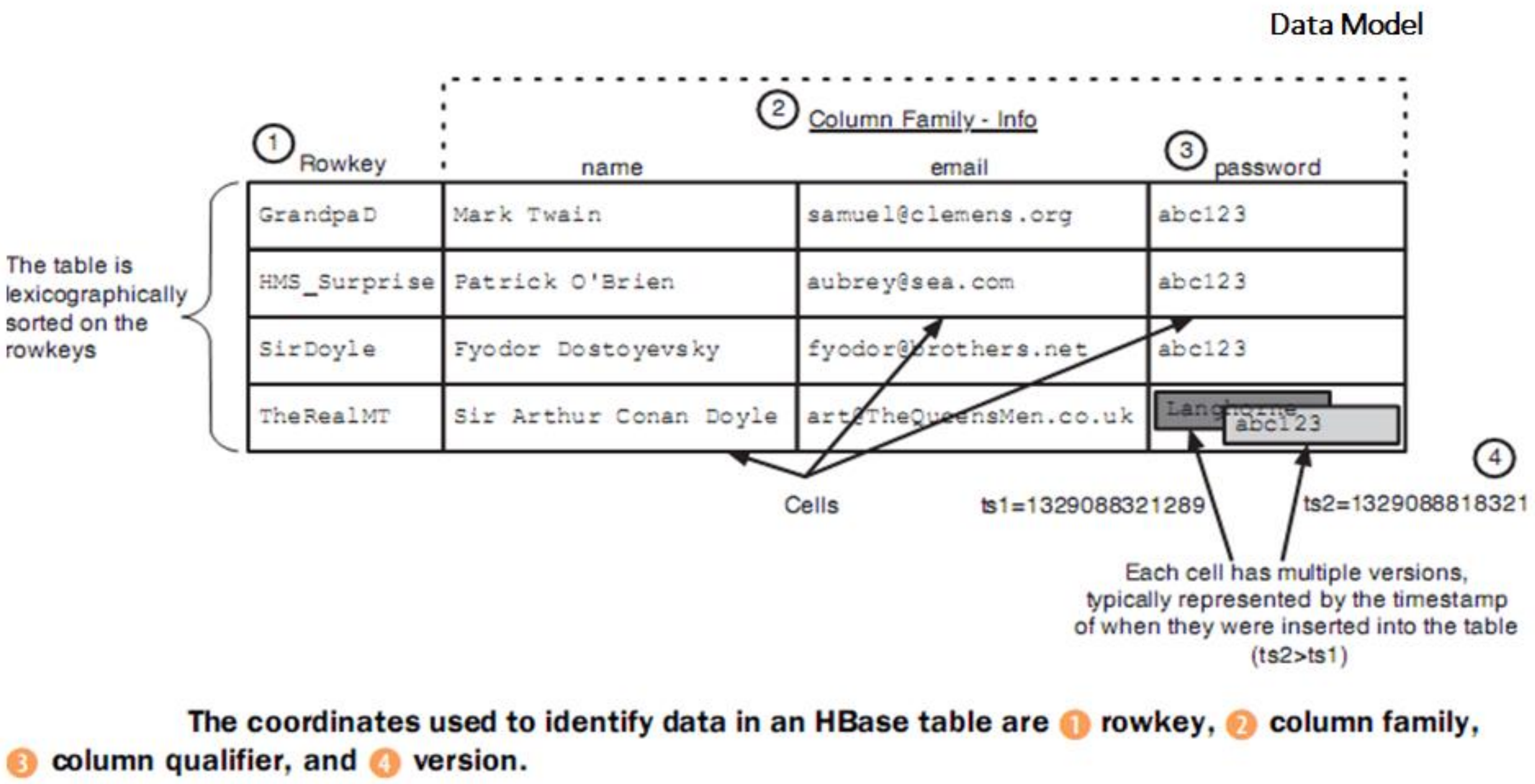
Data storage

Coordination, management, monitoring

APACHE HBASE

- A distributed column-oriented database
- Built on top of the Hadoop file system(HDFS)
- An open-source project
- Horizontally scalable
- Data model is similar to Google’s BigTable
- Designed to provide quick random access to huge amounts of structured data
- Leverages the fault tolerance provided by HDFS
- Is a part of the Hadoop ecosystem (random real-time read/write access to data in the HDFS)
- Data can be stored in HDFS either directly or through HBase

HBASE DATA MODEL



Data Model

Rowkey	name	email	password
GrandpaD	Mark Twain	samuel@clemens.org	abc123
HMS_Surprise	Patrick O'Brien	aubrey@sea.com	abc123
SirDoyle	Fyodor Dostoyevsky	fyodor@brothers.net	abc123
TheRealMT	Sir Arthur Conan Doyle	art@TheQueensMen.co.uk	Langhorne abc123

The table is lexicographically sorted on the rowkeys

Each cell has multiple versions, typically represented by the timestamp of when they were inserted into the table (ts2>ts1)

The coordinates used to identify data in an HBase table are 1 rowkey, 2 column family, 3 column qualifier, and 4 version.

Source: https://images.javatpoint.com/hadooppages/images/HBase_Data_Model.PNG (accessed on October 28. 2024)

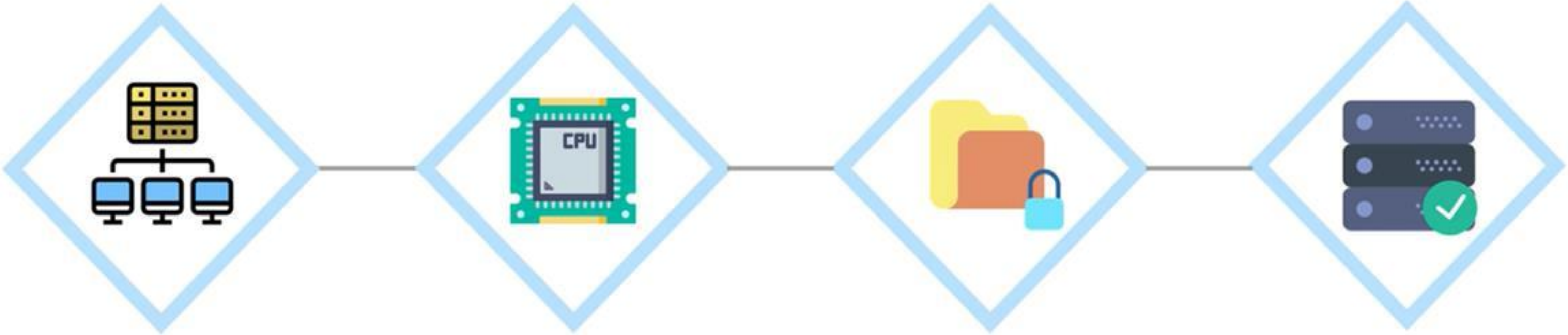
HBASE vs HDFS

HDFS	HBase
Distributed file system suitable for storing large files	Database built on top of HDFS
Does not support fast record lookups	Provides fast lookups for larger tables
High latency batch processing	Low latency access to single rows (random access)
Provides only sequential access to data	Internally uses hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups

Relational DB vs HBase

Relational DBMS	**HBase**
Governed by its schema, which describes the entire structure of tables	Is schema-less,it doesn‘t have the concept of fixed Columns schema, defines only column families
Built for small tables–hard to scale	Built for wide tables–horizontally scalable
Transactional	No transactions
Normalized data	De-normalized data
Good for structureddata	Good for semi-structuredas wellas structureddata
OLTP	OLAP

HADOOP



Provides distributed storage

Implemented on commodity hardware

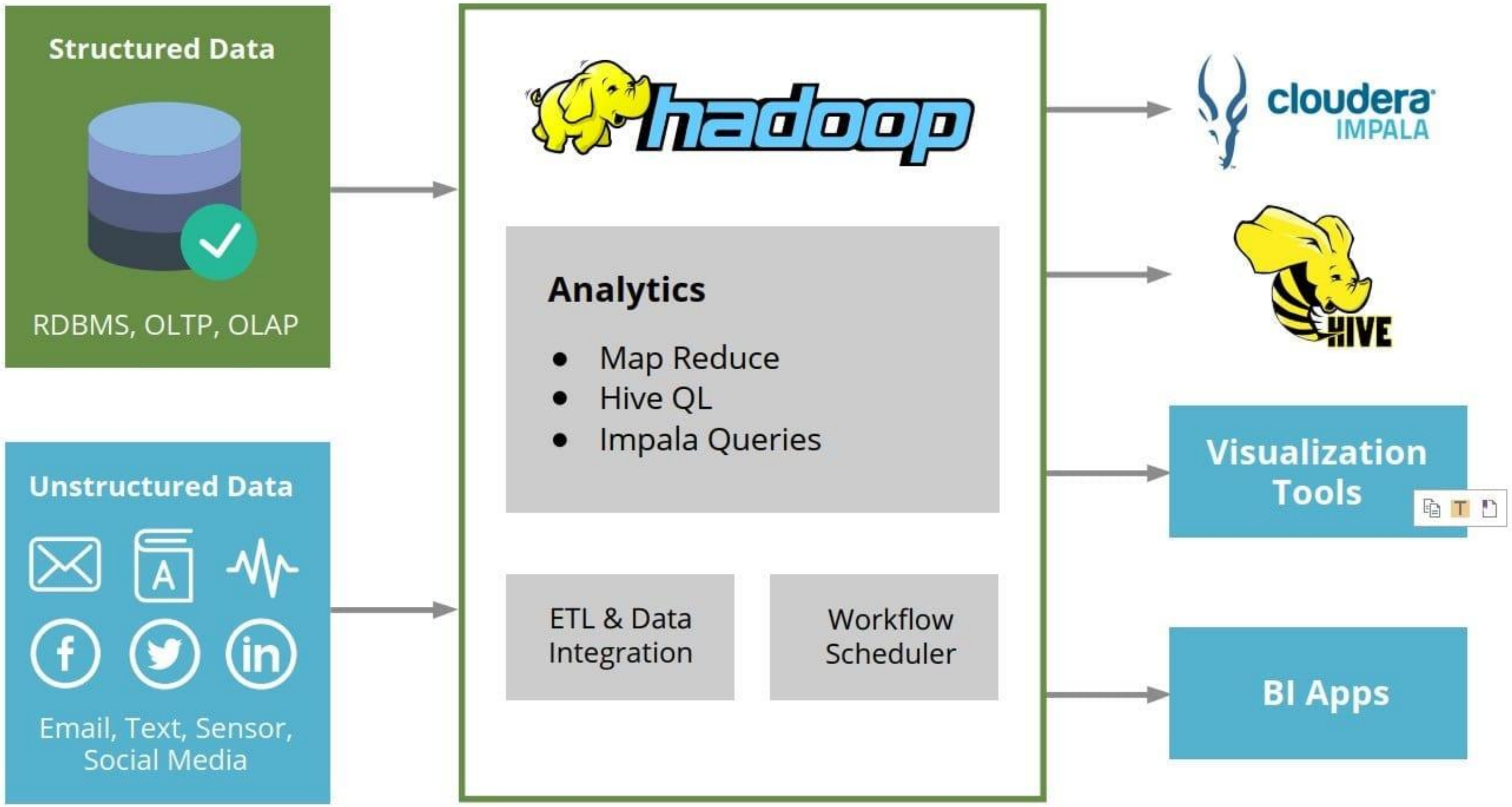
Provides data security

Highly fault tolerant

HADOOP USE CASES

- **Data warehousing**: storing and managing large amounts of structured and unstructured data
- **Data mining**: discovering patterns and trends in large datasets
- **Machine learning**: training and deploying ML models on large datasets
- **Real-time analytics:** processing data in real-time
- **Log analysis**: analyzing large amounts of log data to identify anomalies and security threats

BASIC HADOOP DATA PIPELINE



Structured Data

RDBMS, OLTP, OLAP

Unstructured Data

Email, Text, Sensor, Social Media

hadoop

Analytics

- Map Reduce
- Hive QL
- Impala Queries

ETL & Data Integration

Workflow Scheduler

cloudera IMPALA

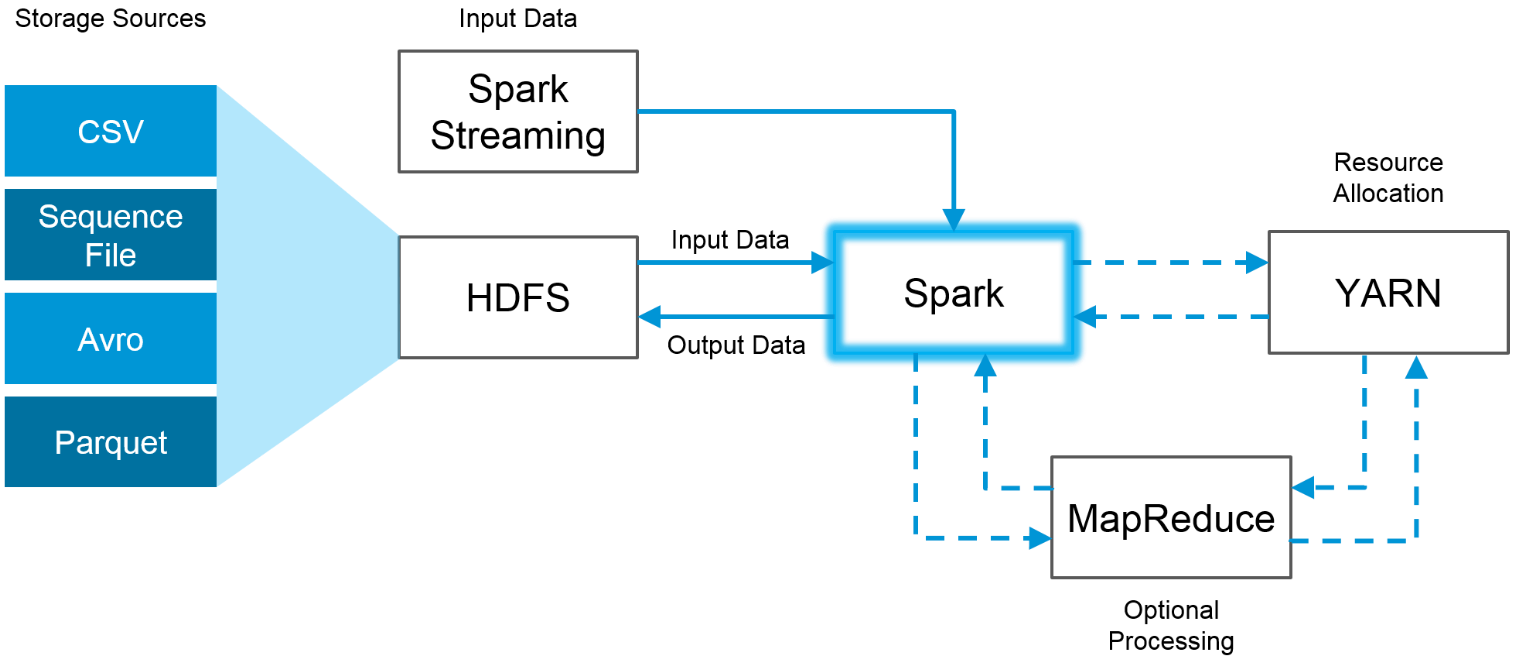
HIVE

Visualization Tools

BI Apps

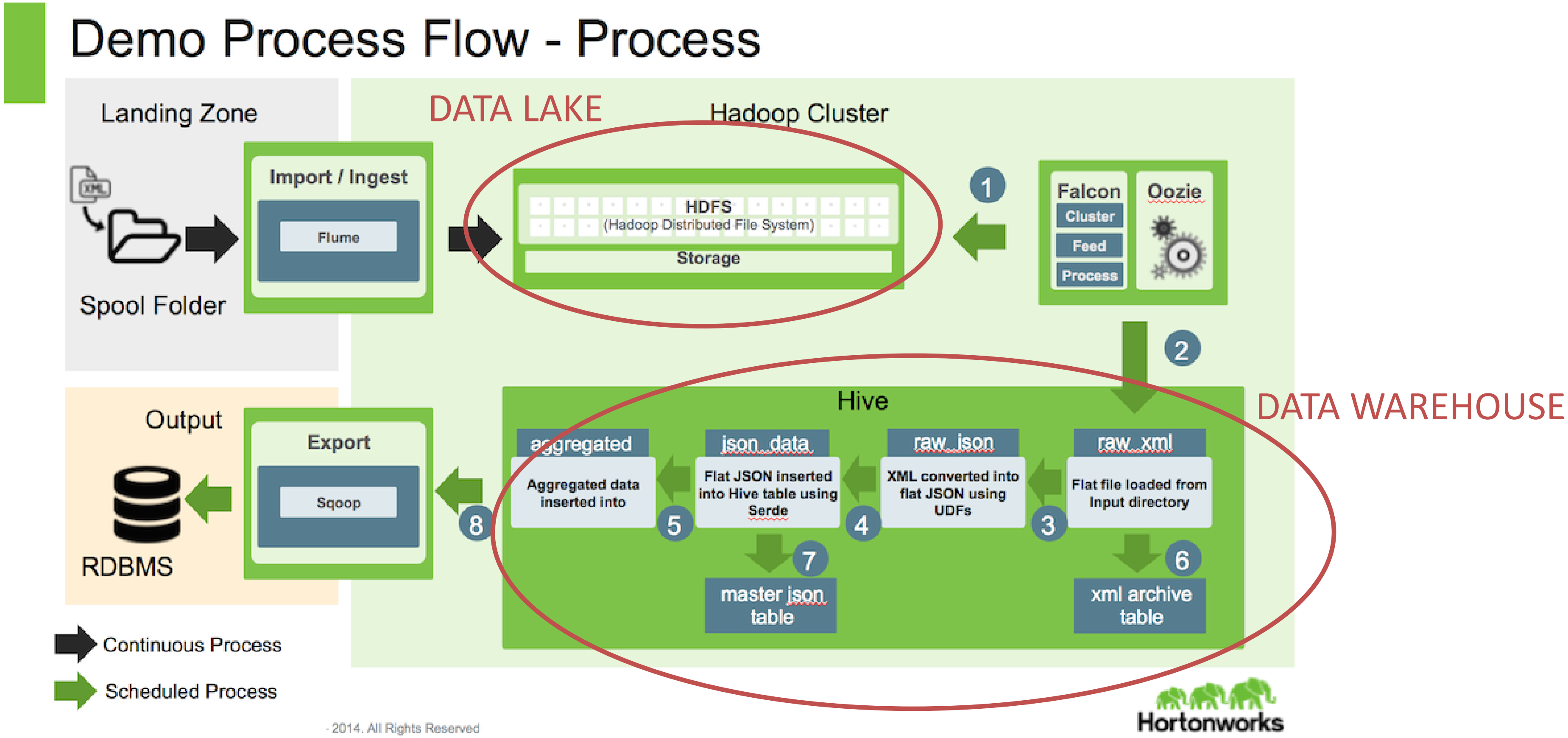
Source: https://cdn-ikpmokn.nitrocdn.com/wzsiHUTKOGMusXyLcAuGjsKvQklTiGbG/assets/images/optimized/rev-61bc12e/thirdeyedata.ai/wp-content/uploads/2018/06/hadoop-graphics.jpg (accessed on October 28, 2024)

EXAMPLE



Source: https://adinasarapu.github.io/images/hadoop.png (accessed on October 28, 2024)

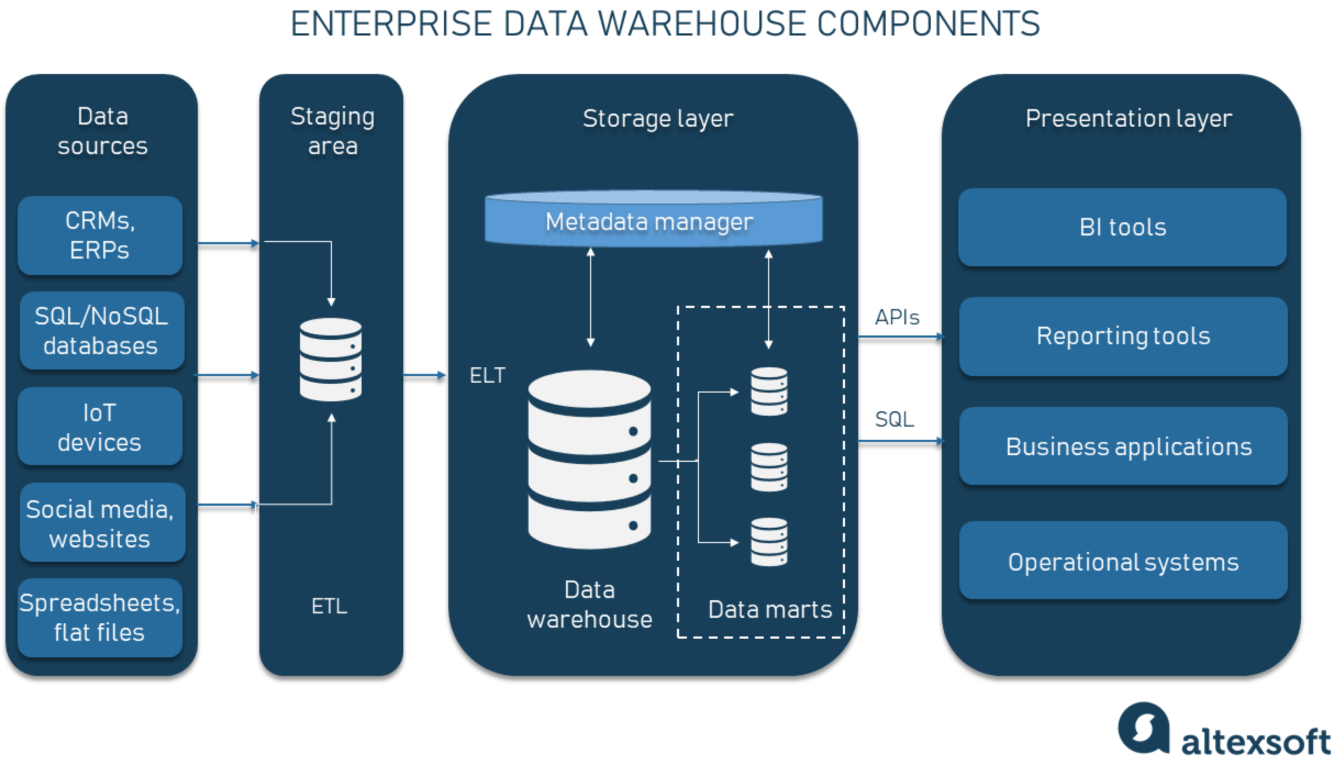
EXAMPLE



Source: https://raw.githubusercontent.com/sainib/hadoop-data-pipeline/master/READMEs/imgs/architecture.png (accessed on October 28, 2024)

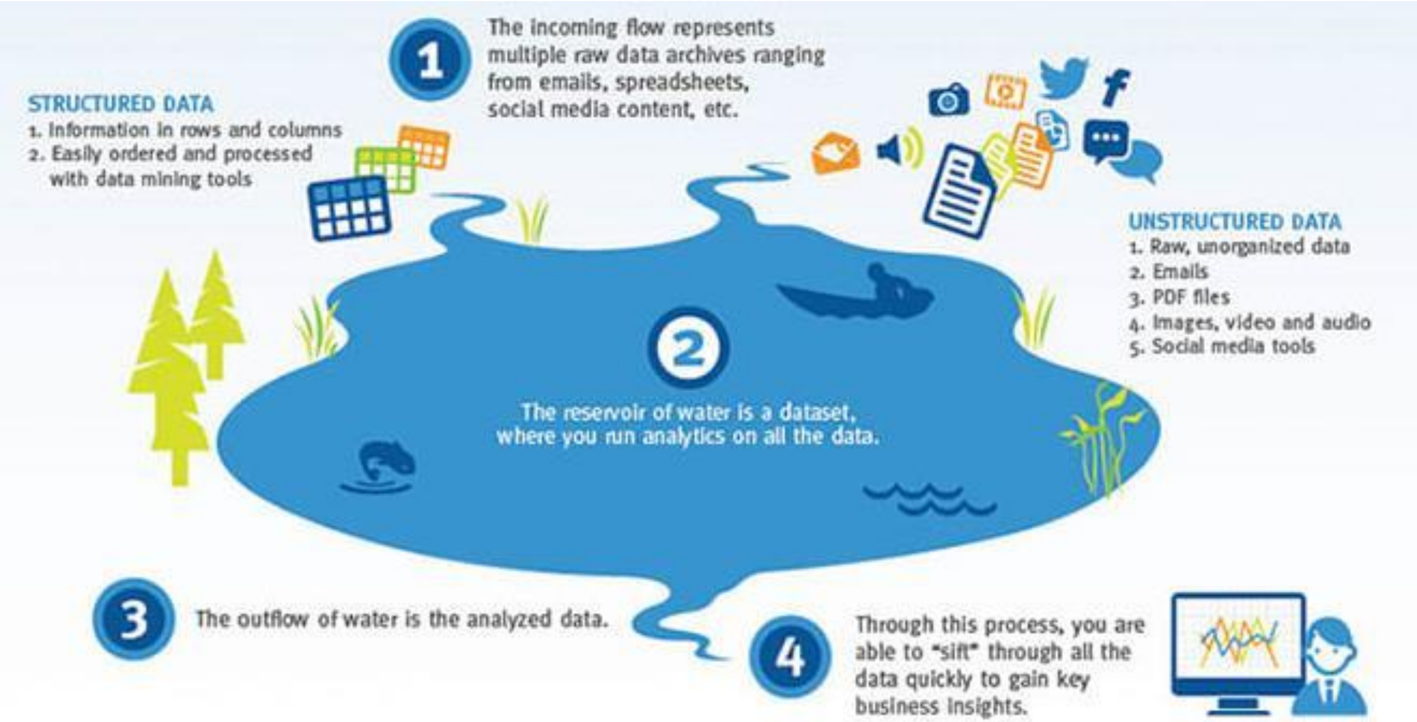
DATA WAREHOUSES

- A storage intended for storing data obtained from transactional systems, operational data warehouses, and external sources
- Optimized for analyzing relational data coming from transactional systems and various business applications
- The data structure and schema are defined in advance
- The data is cleaned, enriched, and transformed to provide a single source of truth.



DATA LAKES

- Highly scalable storage that allows for the storage of large amounts of raw data in its original format
- A processing system that can ingest streaming data without compromising data structure
- Fast access to data without traditional data modelling



STRUCTURED DATA

1. Information in rows and columns
2. Easily ordered and processed with data mining tools

1 The incoming flow represents multiple raw data archives ranging from emails, spreadsheets, social media content, etc.

UNSTRUCTURED DATA

1. Raw, unorganized data
2. Emails
3. PDF files
4. Images, video and audio
5. Social media tools

2 The reservoir of water is a dataset, where you run analytics on all the data.

3 The outflow of water is the analyzed data.

4 Through this process, you are able to "sift" through all the data quickly to gain key business insights.

DATA WAREHOUSES vs DATA LAKES

Feature	Data Warehouse	Data Lake
Purpose	Optimized for business intelligence and analytics	Optimized for storing large amounts of raw data
Data Structure	Highly structured and organized	Diverse, including structured, semi-structured, and unstructured data
Data Processing	Data is cleaned, transformed, and integrated before storage	Data is stored in its raw format and processed on demand
Schema	Schema-on-write	Schema-on-read
Access	Primarily used for reporting and analysis	Used for exploratory data analysis, machine learning, and data science
Scalability	Can be scaled to accommodate growing data volumes	Highly scalable to store massive amounts of data
Cost	Higher upfront costs for data preparation and integration	Lower upfront costs, but higher ongoing costs for data processing and analysis
Complexity	More complex to implement and manage	Simpler to implement, but requires more advanced data engineering skills