MITS6005

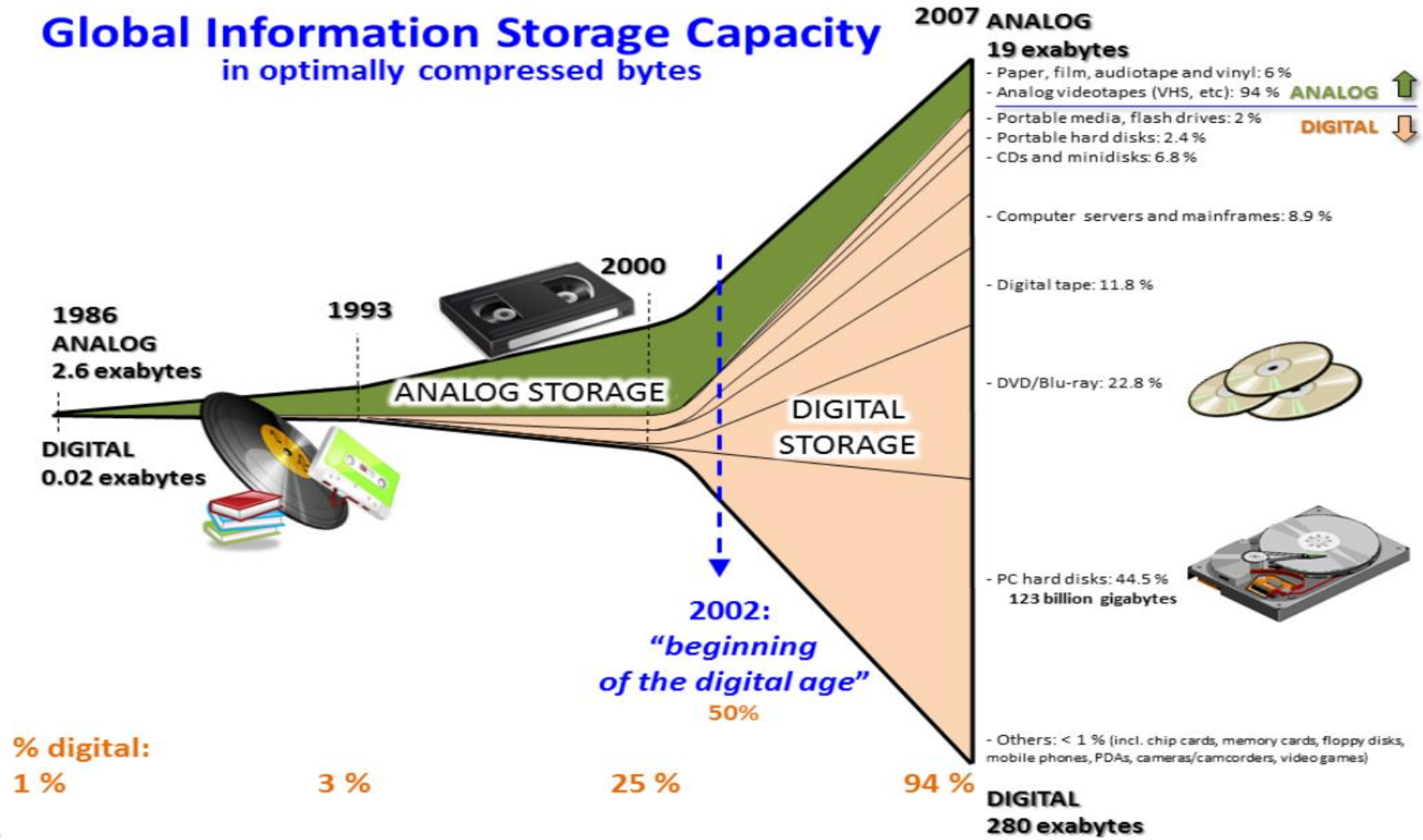
Big Data

Session 4

Hadoop

Growth of global information-storage

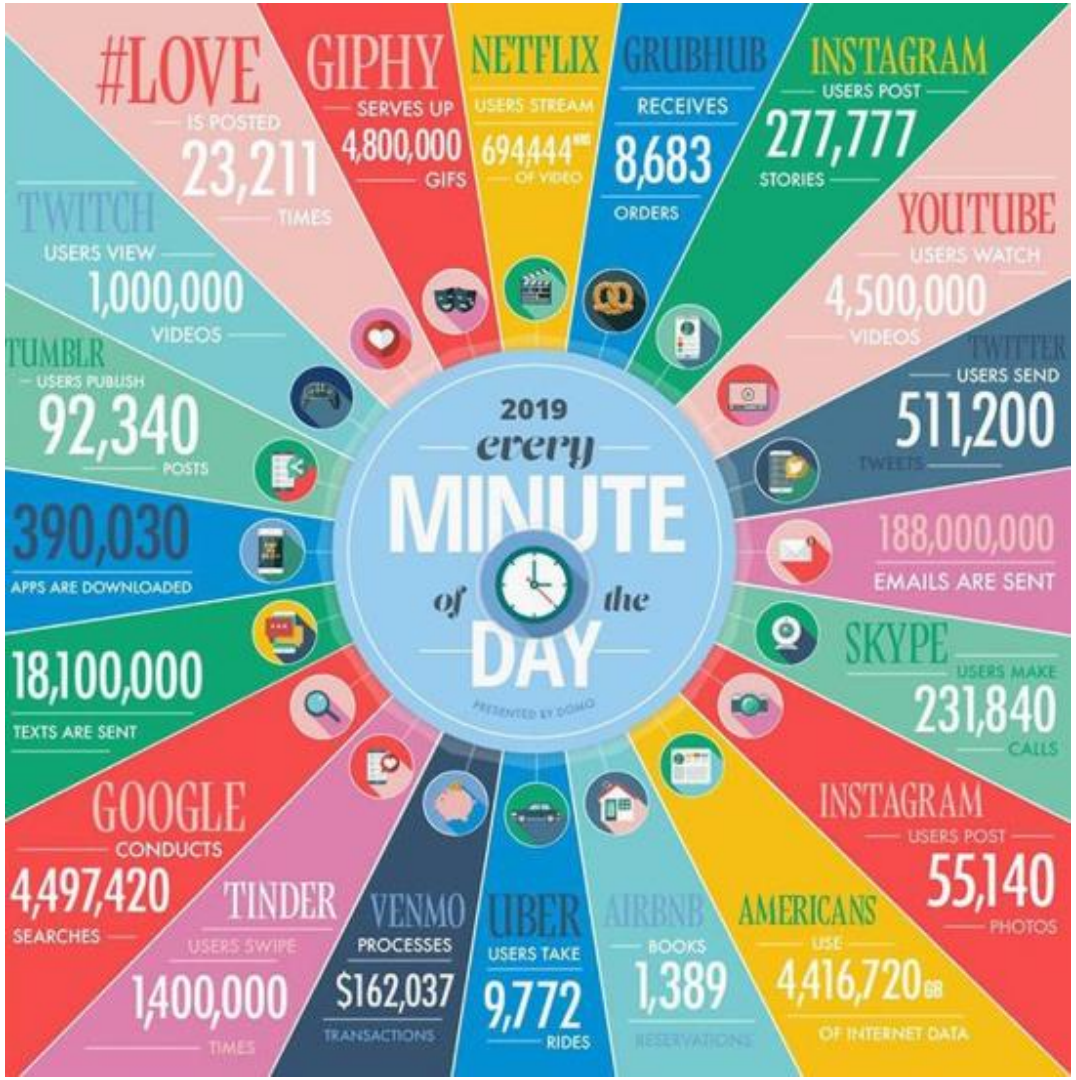
Hilbert, Martin; López, Priscila (2011). "The World's Technological Capacity to Store, Communicate, and Compute Information". *Science*. **332** (6025): 60–65.



How much data in 2019?

- Netflix users stream 694,444 hours of video
- Youtube users watch 4,500,000 videos
- Airbnb books 1,389 reservations
- Uber users take 9,772 rides
- Venmo users transfer 162,037 payments
- By 2025, it's estimated that 463 exabytes of data will be

  created each day globally;
  equivalent of 212,765,957 DVDs per day!



https://www.visualcapitalist.com/how-much-data-is-generated-each-day/
https://www.forbes.com/sites/nicolemartin1/2019/08/07/how-much-data-is-collected-every-minute-of-the-day/#6287da1b3d66

Single-Core Limitation

- Power consumption limits the speed increase we get from transistor density

Distributed Systems

- Allows developers to use multiple machines for a single task

- Data Storage:
  - Typically divided into Data Nodes and Compute Nodes
  - At compute time, data is copied to the Compute Nodes
  - Fine for relatively small amounts of data
  - Modern systems deal with far more data than was gathering in the past

Distributed System: Characteristics/Advantages

- **Scalability and Modular Growth:**

Distributed systems are inherently scalable as they work across different machines and scale horizontally. This means a user can add another machine to handle the increasing workload instead of having to update a single system over and over again.

- **Fault Tolerance and Redundancy**:

A business running a cluster of 8 machines across two data centers means its apps would work even if one data center goes offline

- **Low Latency:**

Since users can have a node in multiple geographical locations, distributed systems allow the traffic to hit a node that's closest, resulting in low latency and better performance.

- **Cost Effectiveness:**

Distributed systems are much more cost effective compared to very large centralized systems. Their initial cost is higher than standalone systems, but only up to a certain point after which they are more about economies of scale.

- **Efficiency:**

Distributed systems allow breaking complex problems/data into smaller pieces and have multiple computers work on them in parallel, which can help cut down on the time needed to solve/compute those problems.

Distributed System: Problems/Disadvantages

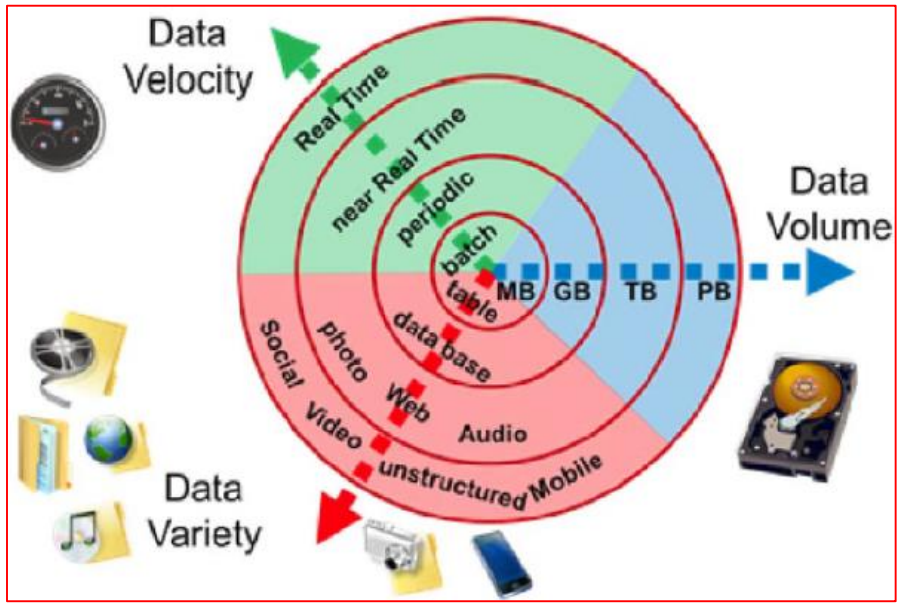
- Programming on a distributed system is much more complex
  - Synchronizing data exchanges
  - Managing a finite bandwidth
  - Controlling computation timing is complicated

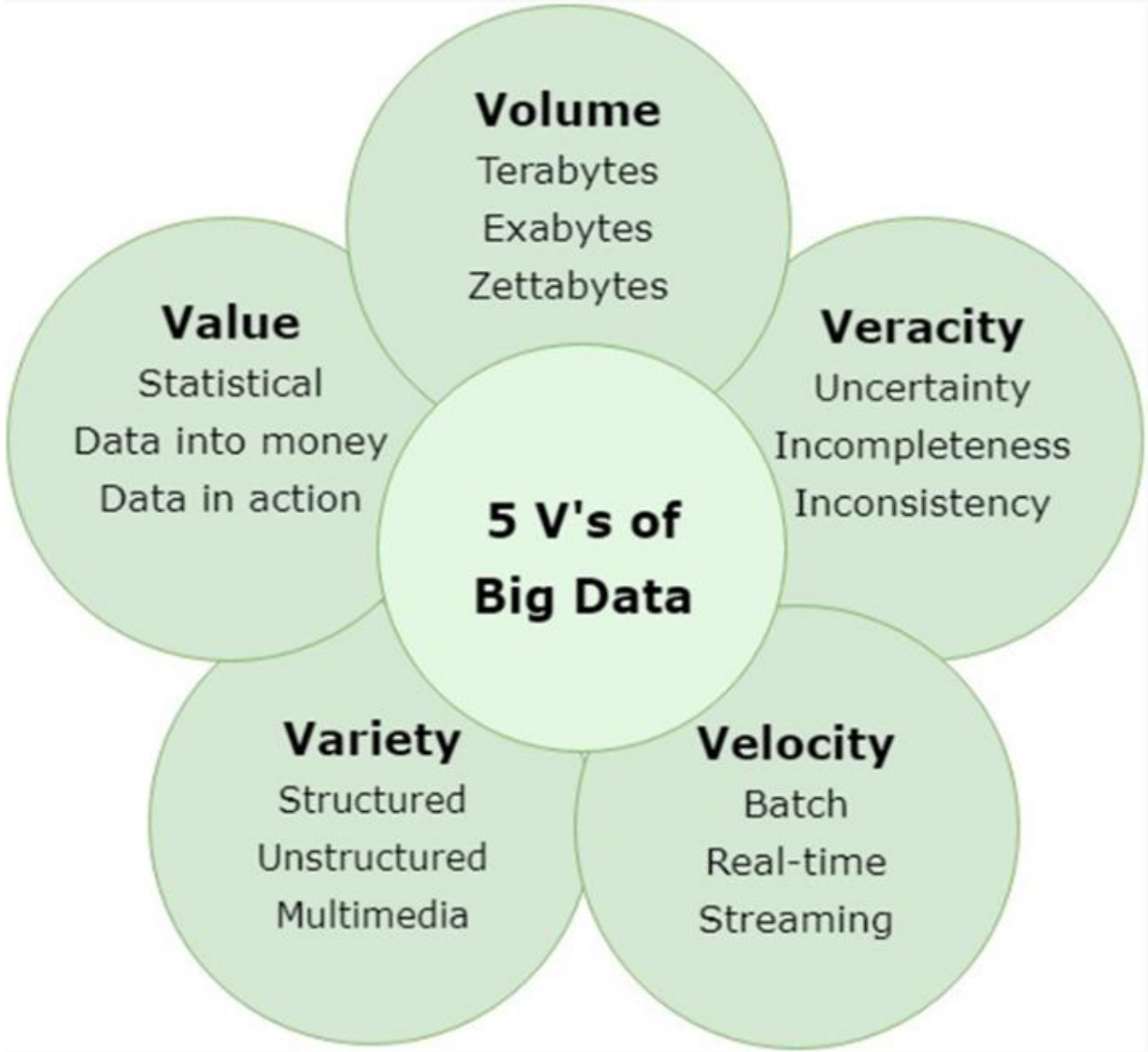
- Distributed systems must be designed with the expectation of failure

- Some of disadvantages of distributed computing:
  - Complexity: difficult to deploy, maintain and troubleshoot/debug than their centralized counterparts.
  - Higher Initial Cost: The deployment cost of a distribution is higher than a single system.
  - Security Concerns: Data access can be controlled fairly easily in a centralized computing system.

Characteristics of Big Data

Big Data comprises of five V's or characteristics:
Volume
Velocity
Variety
Veracity
Value





Hariri, R.H., Fredericks, E.M. & Bowers, K.M. J Big Data (2019) 6: 44. https://doi.org/10.1186/s40537-019-0206-3

What is Apache Hadoop?

- Apache Hadoop is the most used tool in big data industry with its enormous capability of storing and processing big data.
- It is 100% open source framework and runs on inexpensive commodity servers as clusters.
- It can run on a cloud infrastructure.
- Created by Doug Cutting and Mike Carafella in 2005.

Uses for Hadoop

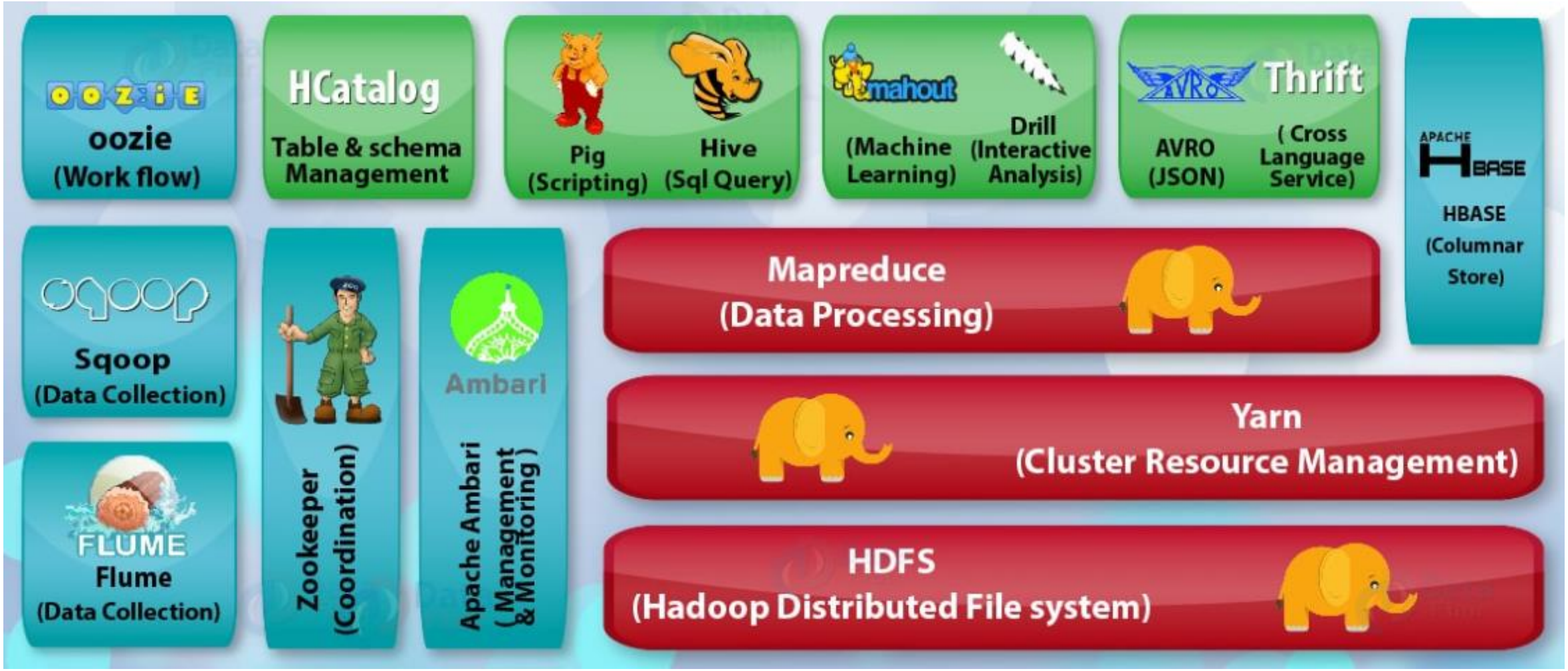
A list of some important use cases for Hadoop:

- Security and Law Enforcement: The national security agency of the USA uses Hadoop to prevent terrorist attacks. It is used to detect and prevent cyber-attacks.
- Customer's Requirements Understanding: Many companies like financial, telecom use this technology to find out the customer's requirement by analysis big amount of data. Social media also uses this technology, and keep posting an advertisement on various social media sites to the target customer whenever the user open their social media on browser.
- Cities and Countries Improvement: Used to development of the country, state, cities by analyzing of data, example traffic jams can be controlled by uses of Hadoop, it used in the development of a smart city, It used to improve the transport of city. It gives proper guidelines for the buses, train, and another way of transportation.
- Financial Trading and Forecasting: Hadoop is used in the trading field. It has a complex algorithm that scan markets with predefined condition and criteria to find out trading opportunities. It is designed in a way that it can work without human interaction in case nobody present to monitor the things according to end users need this works.
- Understanding and Optimizing Business Processes: Retailer can customize their stocks by prediction came from various source like social media google search, various other platforms. A company can take best decision to improve their business and maximized their profits.
- Improving Healthcare and Public Health: Hadoop is used in the medical field to improve the public health. Many health-related applications are based on Hadoop only, where they monitor day to day activities.

More examples and reference: https://www.educba.com/uses-of-hadoop/

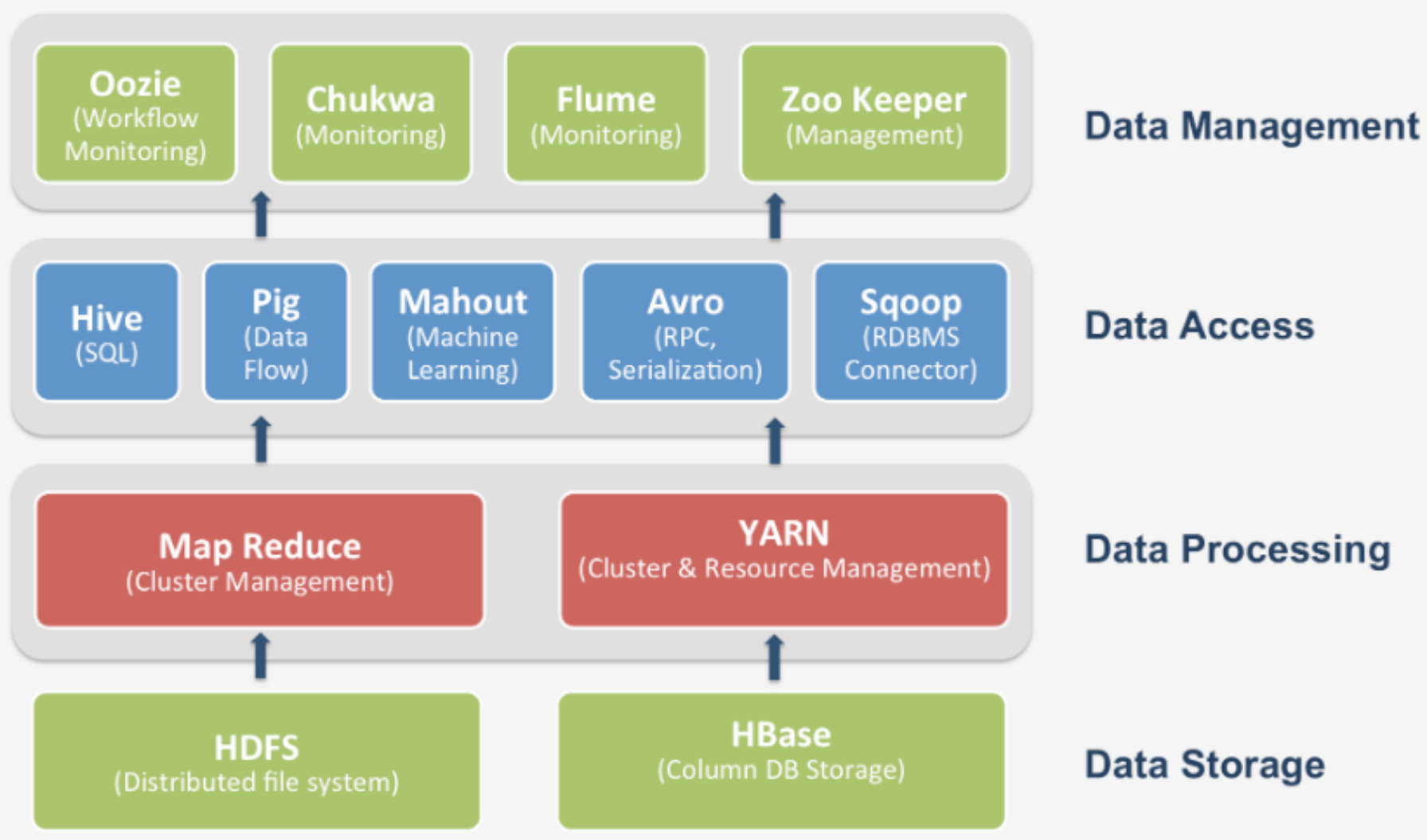
Who Uses Hadoop?

The Hadoop Ecosystem



Hadoop ecosystem explained in the above figure

The Hadoop Ecosystem and use cases

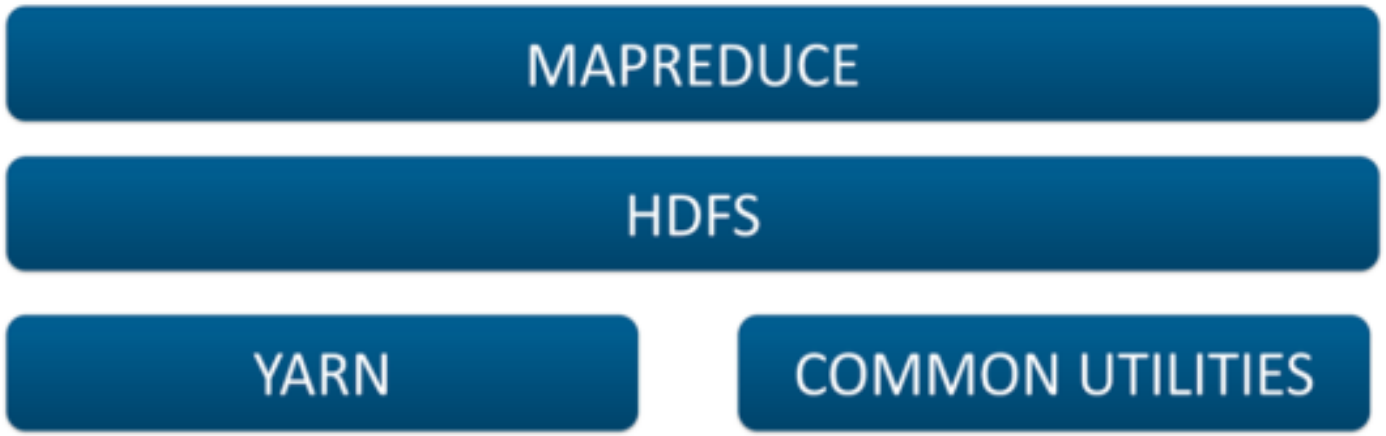


Hadoop ecosystem and use case of each

Hadoop Core Components

The Core Components of Hadoop are as follows:

- **MapReduce**
- **HDFS**
- **YARN**
- **Common Utilities**



Motivations for Hadoop

- The existing data storage and processing tools appeared to be inadequate to handle large amounts of data coming from different sources of the internet.

- Several reasons why organizations use Hadoop, such as:
  - Low cost;
  - High computing power;
  - Scalability;
  - Storage flexibility;
  - Data protection.

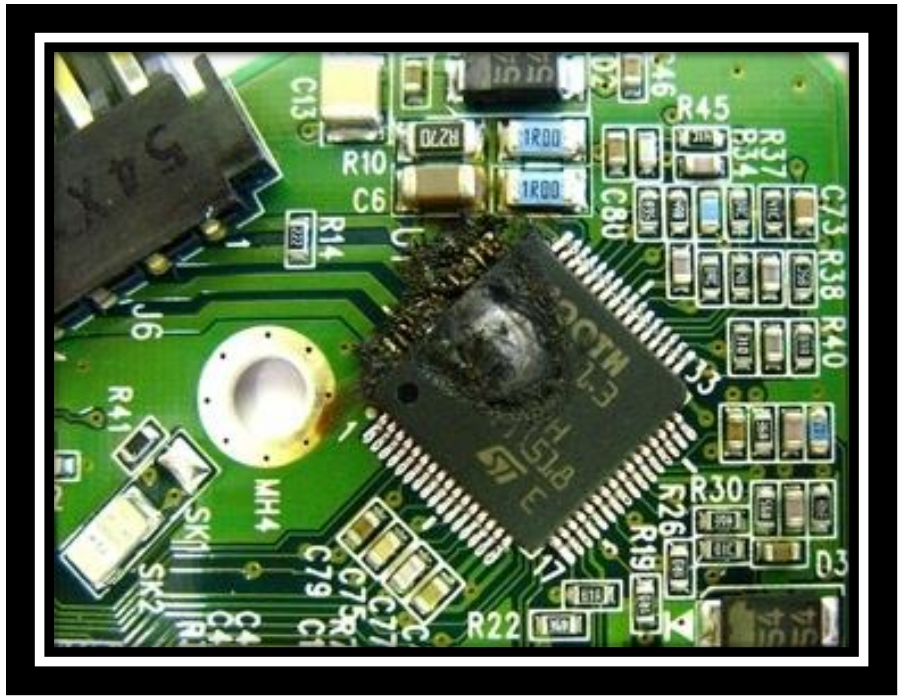
Requirements for Hadoop

- Must support partial failure
- Must be scalable

Partial Failures

- Failure of a single component must not cause the failure of the entire system only a degradation of the application performance

- Failure should not result in the loss of any data



Component Recovery

- If a component fails, it should be able to recover without restarting the entire system
- Component failure or recovery during a job must not affect the final output

Scalability

- Increasing resources should increase load capacity

- Increasing the load on the system should result in a graceful decline in performance for all jobs
  - Not system failure

Hadoop

- Based on work done by Google in the early 2000s
  - “The Google File System” in 2003
  - “MapReduce: Simplified Data Processing on Large Clusters” in 2004
- The core idea was to distribute the data as it is initially stored
  - Each node can then perform computation on the data it stores without moving the data for the initial processing

Core Hadoop Concepts

- Applications are written in a high-level programming language
  - No network programming or temporal dependency
- Nodes should communicate as little as possible
  - A “shared nothing” architecture
- Data is spread among the machines in advance
  - Perform computation where the data is already stored as often as possible

High-Level Overview

- When data is loaded onto the system it is divided into blocks
  - Typically 64MB or 128MB
- Tasks are divided into two phases
  - Map tasks which are done on small portions of data where the data is stored
  - Reduce tasks which combine data to produce the final output
- A master program allocates work to individual nodes

Fault Tolerance

- Failures are detected by the master program which reassigns the work to a different node
- Restarting a task does not affect the nodes working on other portions of the data
- If a failed node restarts, it is added back to the system and assigned new tasks
- The master can redundantly execute the same task to avoid slow running nodes

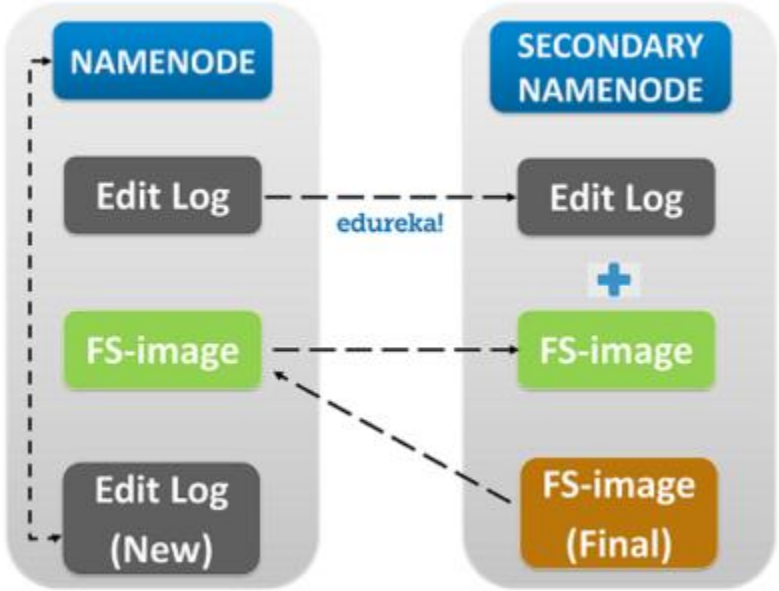
Hadoop Distributed File System (HDFS)

- HDFS is a file system written in Java based on the Google's GFS
- Provides redundant storage for massive amounts of data
- Responsible for storing data on the cluster
- Data files are split into blocks and distributed across the nodes in the cluster
- Each block is replicated multiple times

HDFS Basic Concepts

The HDFS comprises the following components.

- ✓ **NameNode:**
- ✓ **DataNode:**
- ✓ **Secondary NameNode:**



HDFS Basic Concepts

- **NameNode:** The centralized piece of the HDFS, known as the *Master* and designed to store the *Meta Data.* Name Node is responsible for monitoring the *Health Status* of the *Slave Nodes* and to assign *Tasks* to the Data Nodes.

  Tasks of HDFS NameNode:

  – Manage file system namespace.

  - Regulates client's access to files.
  - Executes file system execution such as naming, closing, opening files and directories.

- **DataNode:** The actual unit which stores the data, known as the *Slave* and responds to the Name Node about its Health Status and the task status in the form of a *Heartbeat.* If the Data Node fails to respond to the Name Node, then the Name Node considers the Slave Node to be Dead and reassigns the task to the Next available Data Node.

  Tasks of HDFS DataNode:

  – DataNode performs operations like block replica creation, deletion, and replication according to the instruction of NameNode.

  – DataNode manages data storage of the system.

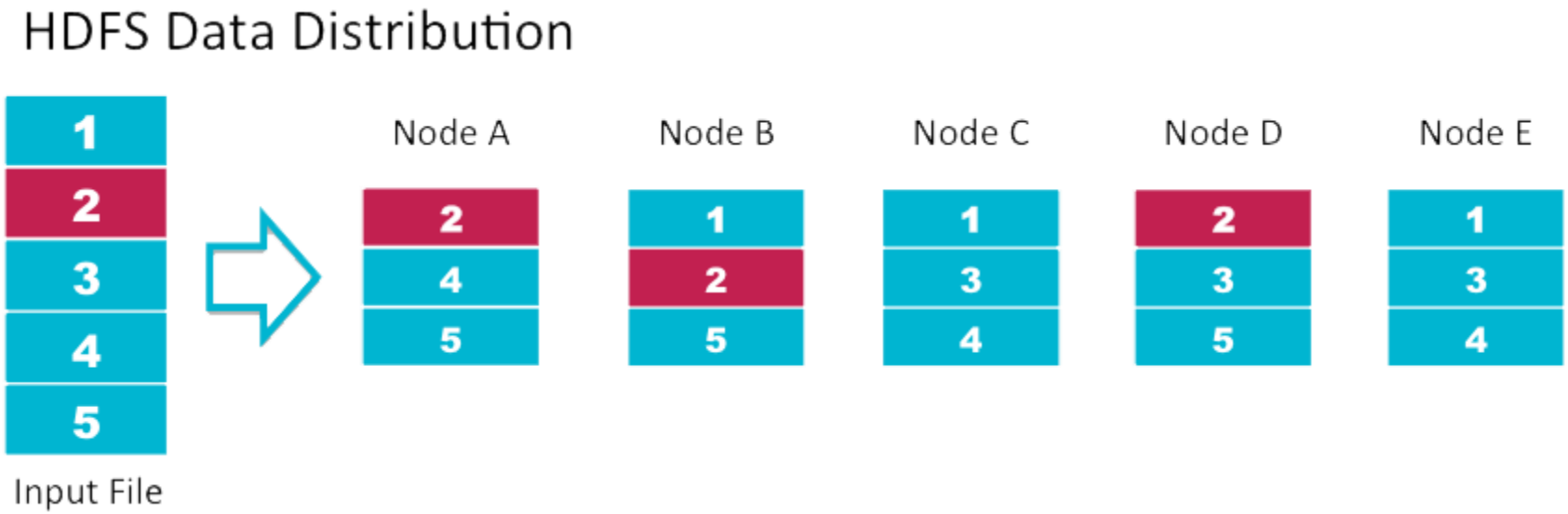
- **Secondary NameNode:** The Secondary Name Node is not a backup of the name node. It acts as a *Buffer* to the Name Node. It stores the intermediate updates the *FS-image* of the Name Node in the *Edit-log* and updates the information to the *Final FS-image* when the name node is inactive.

HDFS Basic Concepts

- HDFS works best with a smaller number of large files
  - Millions as opposed to billions of files
  - Typically 100MB or more per file
- Files in HDFS are write once
- Optimized for streaming reads of large files and not random reads
- Files are split into blocks
- Blocks are split across many machines at load time
  - Different blocks from the same file will be stored on different machines
- Blocks are replicated across multiple machines
- The NameNode keeps track of which blocks make up a file and where they are stored

Data Replication

- Default replication is 3-fold



Data Retrieval

- When a client wants to retrieve data
  - Communicates with the NameNode to determine which blocks make up a file and on which data nodes those blocks are stored
  - Then communicated directly with the data nodes to read the data

MapReduce

- A method for distributing computation across multiple nodes
- Each node processes the data that is stored at that node
- **MapReduce:** It is a Software Data Processing model, designed in Java Programming Language.
  - **Map:** It takes data and set then divides it into chunks such that they are converted into a new format which would be in the form of a key-value pair.
  - **Reduce:** It is the second part where the **Key/Value** pairs are reduced to tuples.
- The **MapReduce** process enables us to perform various operations over the big data such as **Filtering** and **Sorting** and many such similar ones.

MapReduce Features

- Automatic parallelization and distribution
- Fault-Tolerance
- Provides a clean abstraction for programmers to use

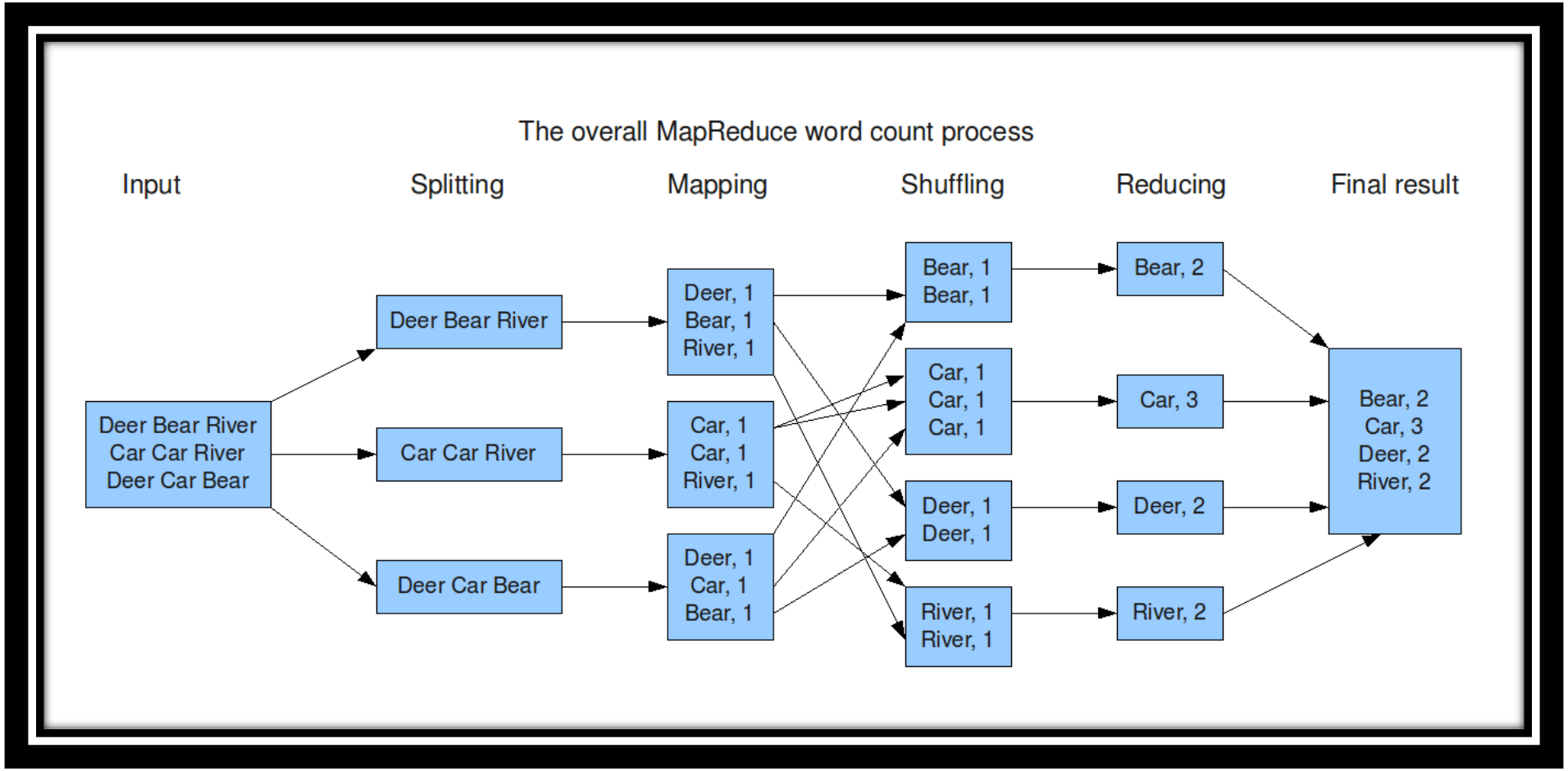
How MapReduce works

- The Mapper:
  - Reads data as key/value pairs
    - The key is often discarded
  - Outputs zero or more key/value pairs

- Shuffle and Sort:
  - Output from the mapper is sorted by key
  - All values with the same key are guaranteed to go to the same machine

- The Reducer
  - Called once for each unique key
  - Gets a list of all values associated with a key as input
  - The reducer outputs zero or more final key/value pairs
  - Usually just one output per input key

MapReduce: Word Count Example



Hadoop Cluster

- NameNode
  - Holds the metadata for the HDFS
- Secondary NameNode
  - Performs housekeeping functions for the NameNode
- DataNode
  - Stores the actual HDFS data blocks
- JobTracker
  - Manages MapReduce jobs
- TaskTracker
  - Monitors individual Map and Reduce tasks

The NameNode

- Stores the HDFS file system information in a fsimage
- Updates to the file system (add/remove blocks) do not change the fsimage file
  - They are instead written to a log file
- When starting the NameNode loads the fsimage file and then applies the changes in the log file

The Secondary NameNode

- **NOT** a backup for the NameNode
- Periodically reads the log file and applies the changes to the fsimage file bringing it up to date
- Allows the NameNode to restart faster when required

JobTracker and TaskTracker

- JobTracker
  - Determines the execution plan for the job
  - Assigns individual tasks

- TaskTracker
  - Keeps track of the performance of an individual mapper or reducer

Why do these tools exist?

- MapReduce is very powerful, but can be awkward to master

- These tools allow programmers who are familiar with other programming styles to take advantage of the power of MapReduce

Other Tools

- Hive
  - Hadoop processing with SQL
- Pig
  - Hadoop processing with scripting
- Cascading
  - Pipe and Filter processing model
- HBase
  - Database model built on top of Hadoop
- Flume
  - Designed for large scale data movement