



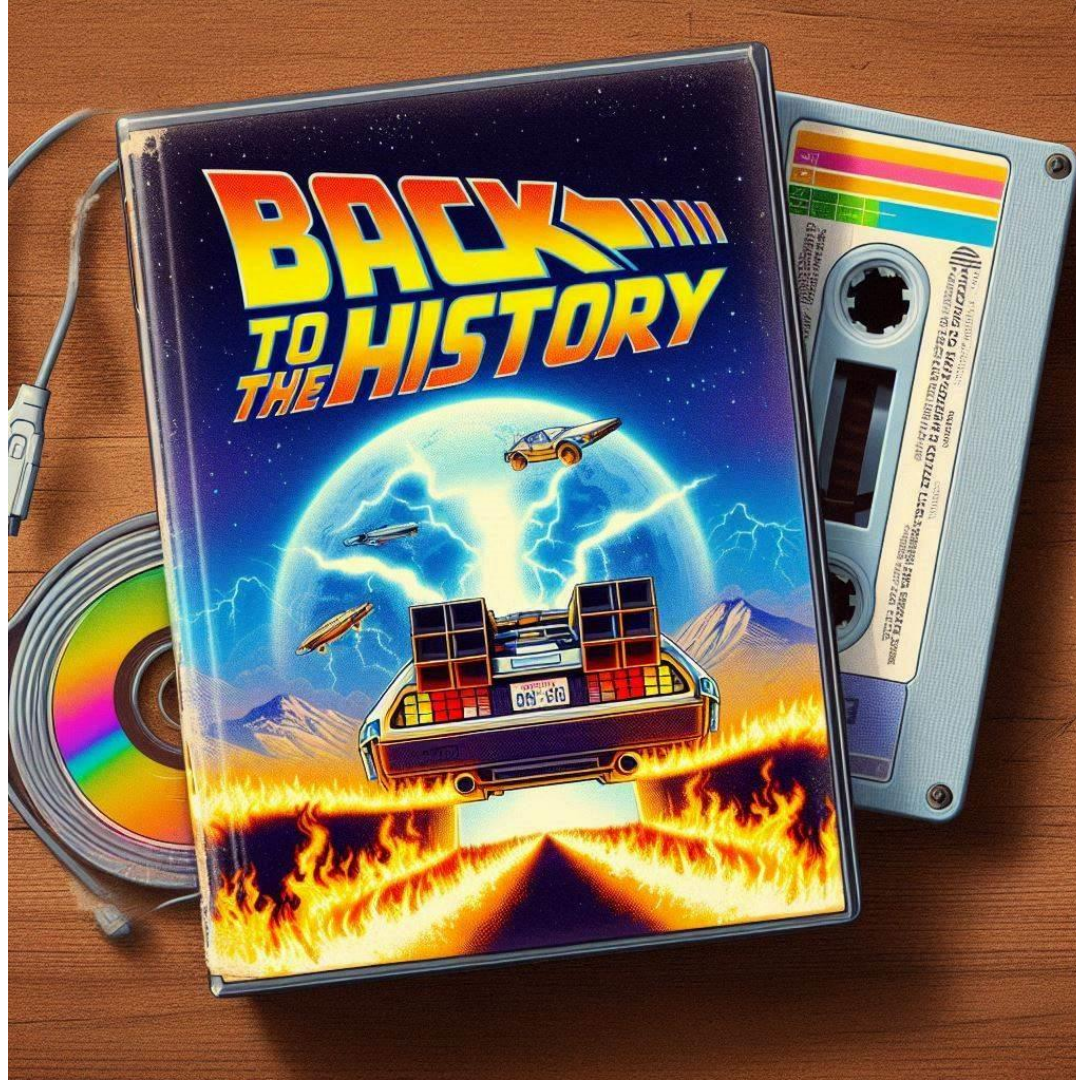
(Big) Data in Time

Alisa Benešová (Petr Filas)

16. 10. 2024

Agenda

1. Motivation
2. A Little Bit of History
3. A Little Bit of Theory
4. Conclusion



◀ PROFINIT ▶

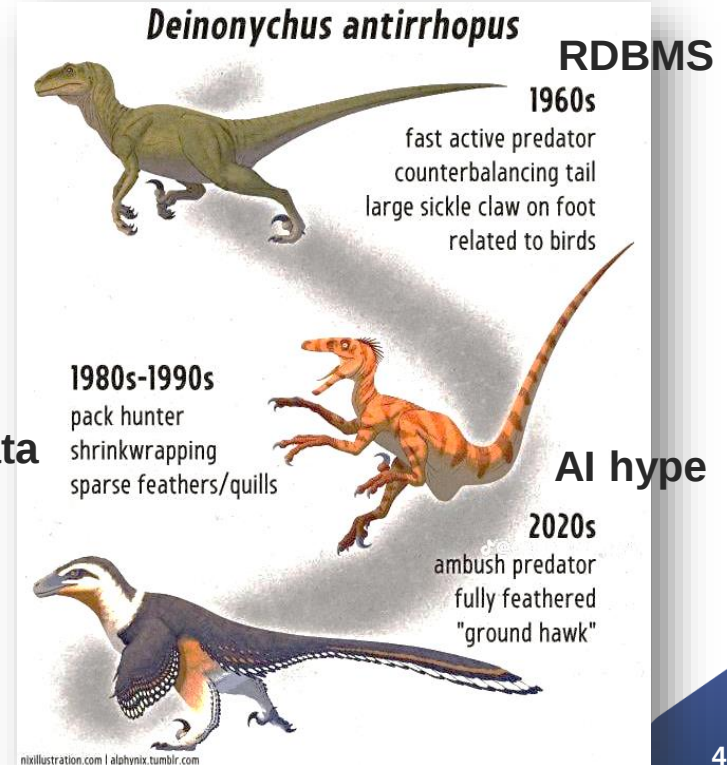
Motivation

◀ PROFINIT ▶

➤ Brief history and brief introduction to big data concepts will show

- Money are important
- Every technology has its rise, peak and decline
- Key concept remains
- Take the best and fix issues
- Let's go through data dinosaur land.

BigData



A Little Bit of History

1950s–1980s: Foundations of Data Processing

- **1950s–1960s:** Early developments in database management systems (DBMS) like hierarchical databases and IBM's Information Management System (IMS). File systems etc.
- **1970:** Edgar F. Codd invented the **relational database model**, laying the foundation for modern databases. He was working in IBM. [Codd's rules applied.](#) SQL development.
- **1980s:** Parallel database systems emerge, offering methods for scaling and distributing data workloads across multiple machines.
- **1989:** The term "**Big Data**" was first used in relation to the challenge of managing and processing massive datasets, especially in scientific computing.



1990s: WWW and Search Engines - this is where it really started



- **1990s:** The explosion of the web leads to a growing need for handling unstructured data at a larger scale. NoSQL databases appeared.
- **1994:** Companies like Yahoo! and Altavista create search engines, bringing forth the need to process massive amounts of data.
 - Altavista 1998 ~13 millions queries/day, 2000 ~80millions queries each day
- **1997:** Michael Cox and David Ellsworth publish a paper *Application-controlled demand paging for out-of-core visualization* (NASA, Intel, Nvidia Research), using the term "Big Data" to describe the challenges of visualizing **large datasets**.



1 Introduction

Visualization provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of **big data**. When data sets do not fit in main memory (*in core*), or when they do not fit even on local disk, the most common solution is to acquire more resources. This *write-a-check* algorithm

2000s: Pre-Hadoop Google Big data Era

◀ PROFINIT ▶

- **2003:** Google introduces the [Google File System \(GFS\)](#), a distributed file system designed to support large-scale data processing.
- **2004:** Google's paper on [MapReduce: Simplified Data Processing on Large Clusters](#) is published. This paradigm **revolutionized** how distributed data is processed by simplifying parallel computing.
- **2004:** Google introduces [Bigtable paper](#), a distributed storage system for managing structured data. Google is still using it.

Not Open Source

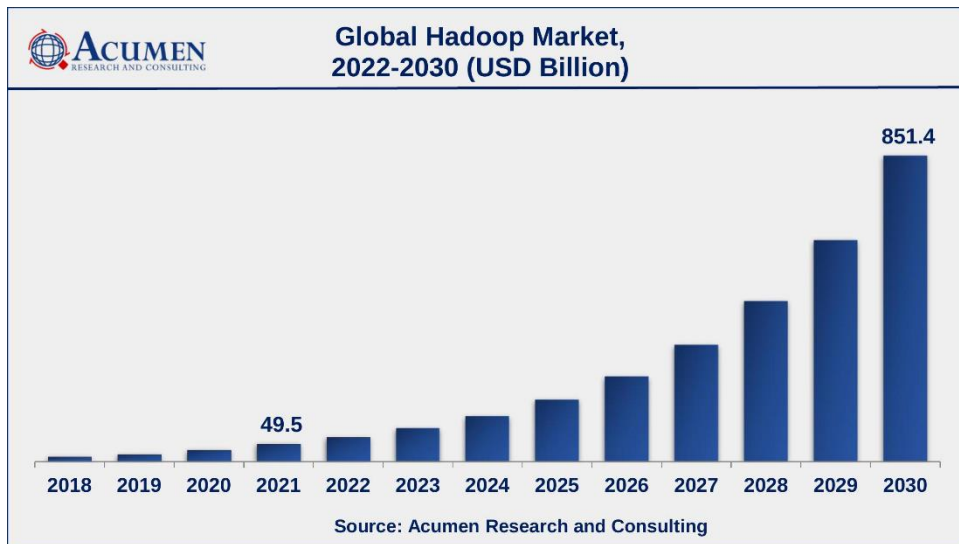
MapReduce paper

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

2006: Open-Source Hadoop Era



- **2006:** *Apache Hadoop* release as OS project.
 - First Hadoop = HDFS + MapReduce
- **2007:** Yahoo! adopts Hadoop for its web search engine, and Hadoop starts gaining significant momentum in industry.



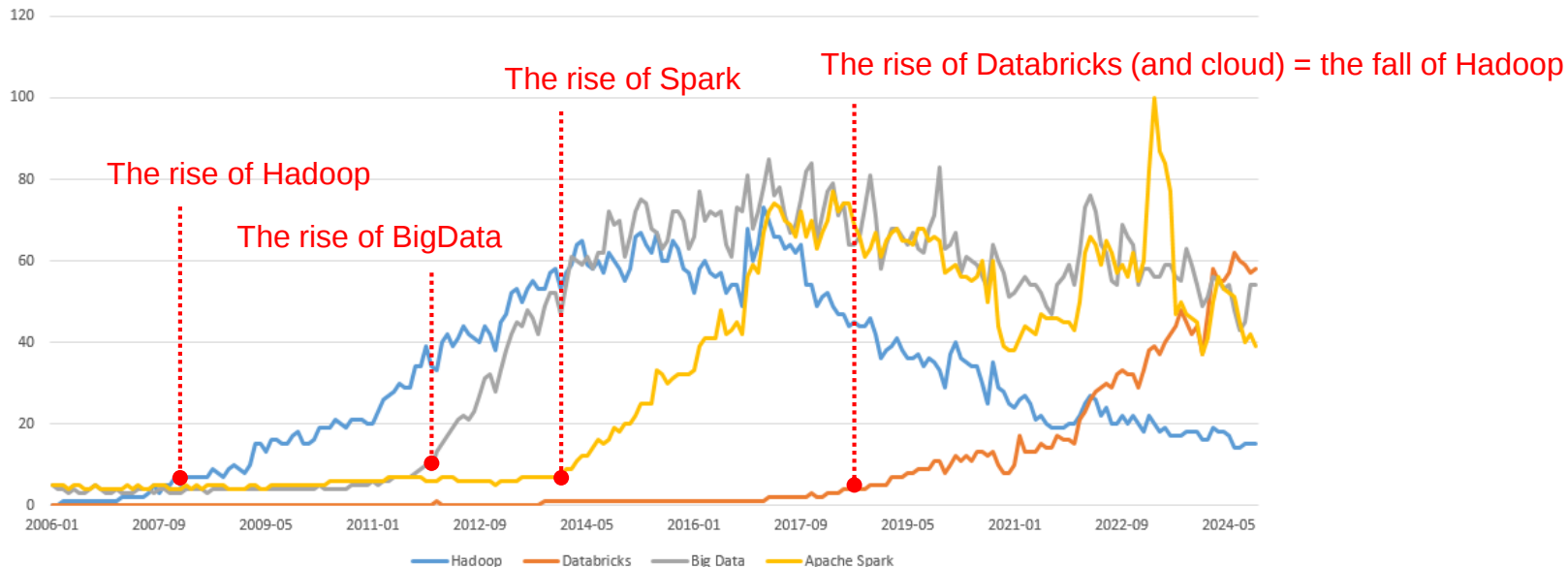
Bright future ahead predicted...

2010s: BigData = Hadoop for a while...

- **2010:** *Apache Hive* emerge, providing higher-level abstractions for querying large datasets in Hadoop "something like database"
- New SQL databases (noSQL with ACID) , cloud databases.
- **2011:** Commercial fight: Cloudera, Hortonworks, and MapR - making it accessible for enterprises.
- **2012:** Apache releases *Hadoop 2.0* with YARN (Yet Another Resource Negotiator), enabling Hadoop to support non-MapReduce applications and ushering in a more flexible resource management framework.
- **2012: Spark**, originally developed by UC Berkeley's AMPLab, is released as a **faster, in-memory alternative** to MapReduce for distributed data processing.

Late 2010s-now: Peak and decline of Hadoop, Bigdata still on

Google trends. Do you remember where it all started?

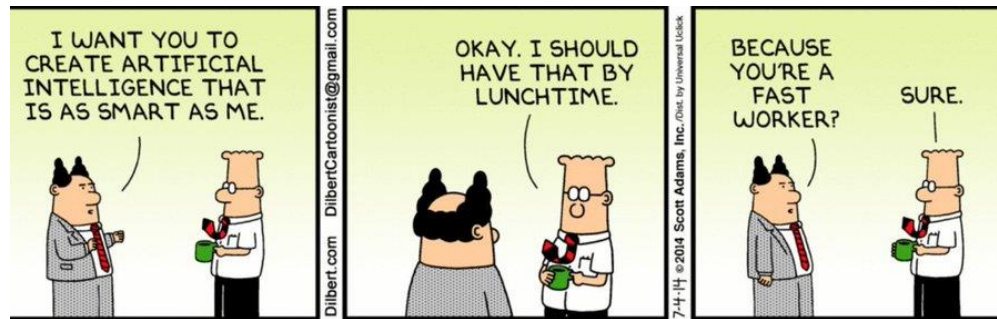


- > **2018:** Hadoop's dominance starts to decline in favor of cloud-native platforms and frameworks such as Apache Kafka, Apache Flink, and more containerized, microservice-based architectures.

2020s: Rise of AI, Streaming, and Modern Data Platforms

< PROFINIT >

- > **2020:** Modern data platforms like *Databricks*, *Snowflake*, and *Google BigQuery* gain traction due to their scalability, simplicity, and cloud-native architectures.
- > **2023-2024:** Boom started with ChatGPT (OpenAI release 2022). The AI, generative AI and machine (earlier known as Data Science 😊). Everyone is integrating big data with AI/ML pipelines, moving towards more real-time analytics and AI-powered decision-making.



- > **2024:** Hadoop is still alive but mostly for on-premise solution (e.g. Cloudera) = cloud-based data platforms won the war (new era is coming)

A Little Bit of Theory

Motivation for Paralell (Data) Processing

◀ PROFINIT ▶

➤ Solve real life problems

- **Complexity** - Problems that are difficult to solve sequentially can often be broken down into smaller, parallel tasks
- **Efficiency and Speed** - By breaking down large tasks into smaller, concurrent processes, parallel processing significantly reduces the time required to complete data-intensive tasks.
- **Optimization**: Parallel processing makes better use of available resources. By distributing tasks across multiple processors, it maximizes the use of computational power and minimizes idle time

➤ Examples

- SETI - Search for Extra-Terrestrial Intelligence
- Financial Modeling
- Recommendations (Netflix)
- Big Data in enterprise



Key Principles

> Decomposition

- Tasks being spread across multiple nodes to work in parallel

> Load Balancing

- Distributing tasks evenly across compute (workers)

> Synchronization & Communication

- Coordinating the execution of parallel tasks to ensure they work together correctly.

> Scalability

- Ensuring that the parallel processing system can handle increasing amounts of data and processing power without significant performance degradation

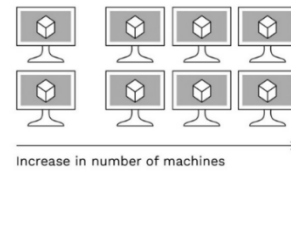
Scalability

Vertical scaling



↑
Increase in processing power

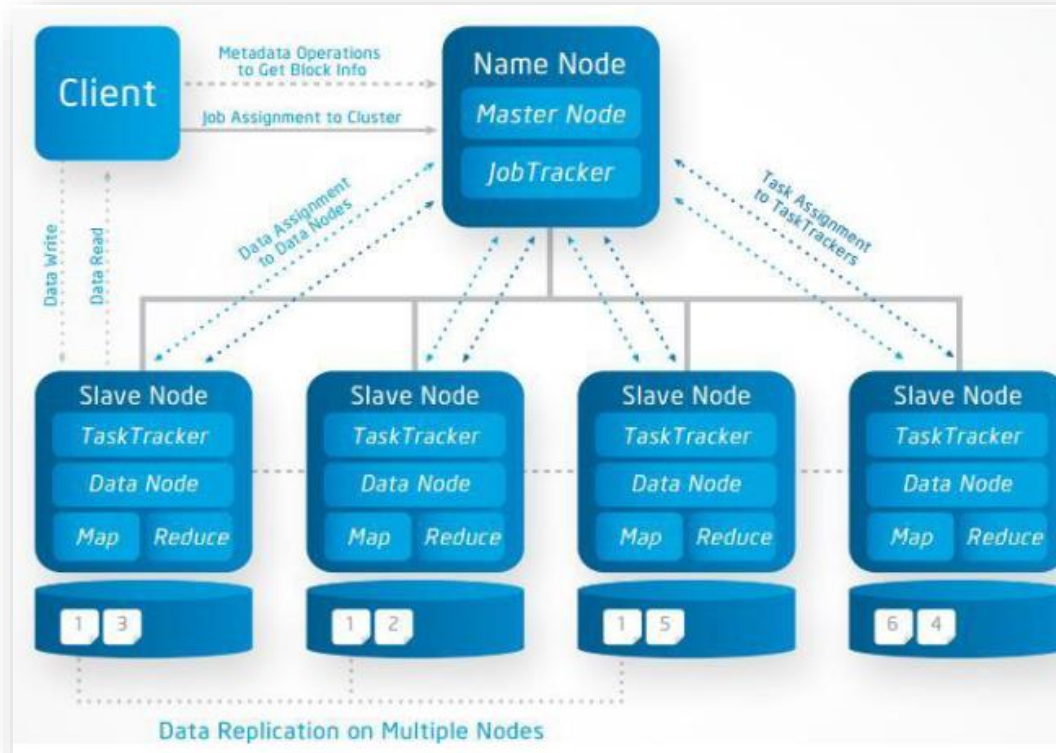
Horizontal scaling



→
Increase in number of machines

Typical Architecture

◀ PROFINIT ▶

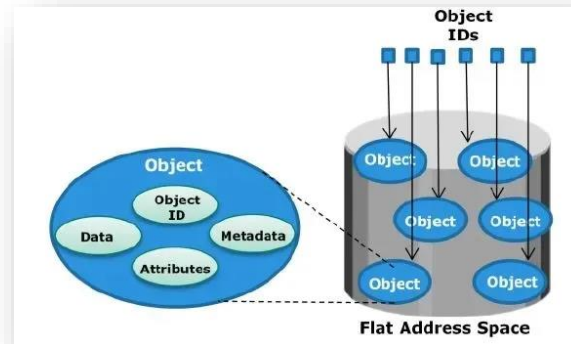


- > Decomposition of
 - Data – leads to **distributed storage** (HDFS, ADLS, S3)
 - Compute – leads to **distributed compute** (MapReduce, Spark)
- > Data and Compute separation (cloud data platforms)
 - By decoupling storage and compute, organizations can scale each component independently.
 - Storage is cheap
 - Compute is expensive
 - A lot of data $\neq$ \$\$\$, if you don't process them
 - A small amount of data $\neq$ \$, if you process them real-time (e.g. streaming)

PROFINIT

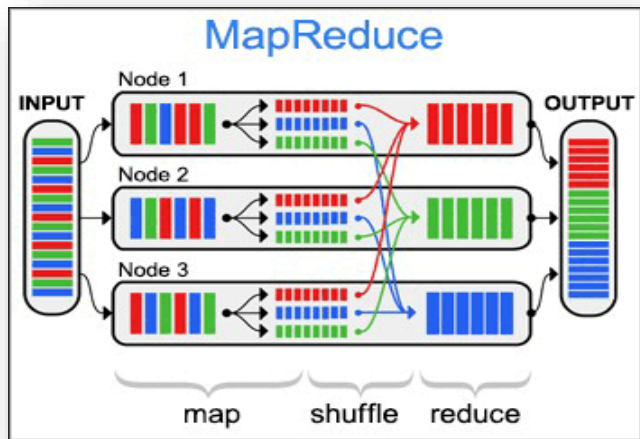
Object-Based Storage (OBS)

- ADLS (Azure Data Lake Storage)
- AWS S3 (Simple Storage Service)



- Flat address space, where each object contains the data, a unique identifier, and metadata
- No random writes within objects ☹️
- Loosely coupled with compute 😊

- MapReduce = programming paradigm



- Map - each node applies the mapping function to its data portion, filtering and sorting it according to parameters.
- Shuffle - mapped data is redistributed to other nodes on the system so that each node contains groups of key-similar data
- Reduce - Data is processed in parallel, per node, per key

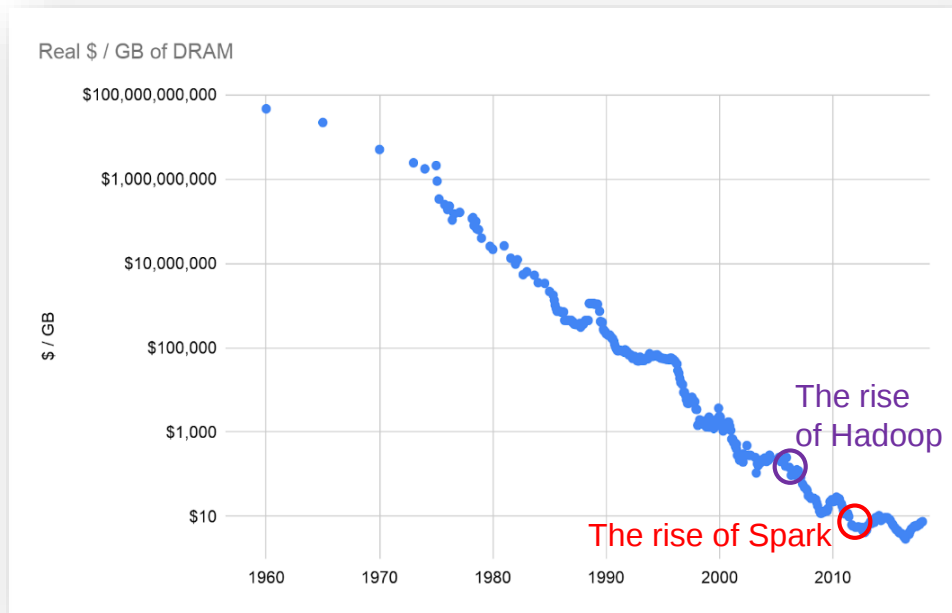
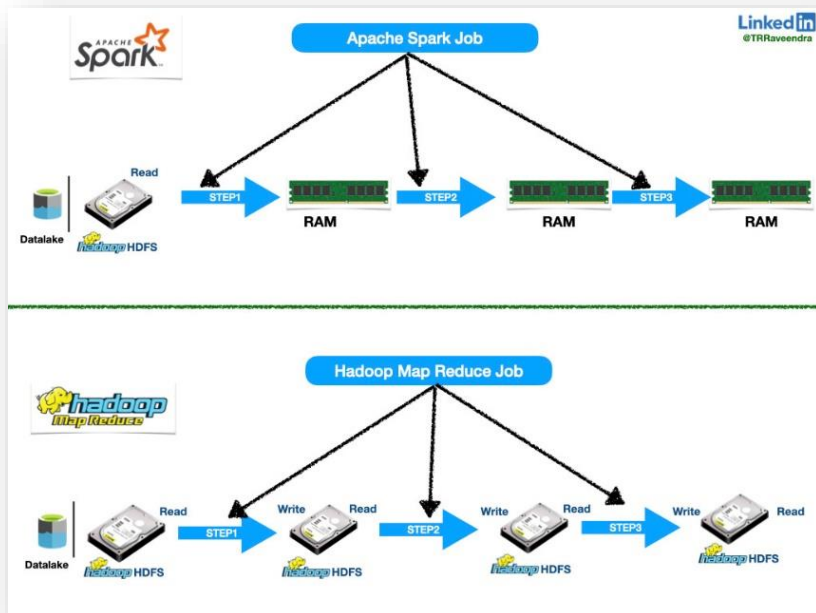
-  = implementation of MapReduce paradigm on Hadoop platform



- Does the same thing but more efficiently and conveniently
 - **Processing speed:** Apache Spark is much faster than Hadoop MapReduce (100x).
 - **Data processing paradigm:** Hadoop MapReduce is designed for batch processing, while Apache Spark is more suited for real-time data processing and iterative analytics (but handles batch as well).
 - **Ease of use:** Apache Spark has a more user-friendly programming interface and supports multiple languages, while Hadoop MapReduce requires developers to write code in Java.
 - **Fault tolerance:** Apache Spark's Resilient Distributed Datasets (RDDs) offer better fault tolerance than Hadoop MapReduce's Hadoop Distributed File System (HDFS).
 - **Integration:** Apache Spark has a more extensive ecosystem and integrates well with other big data tools, while Hadoop MapReduce is primarily designed to work with Hadoop Distributed File System (HDFS).

> So, are there any cons?

- It's all about the RAM
- RAM used to be expensive but the world has changed...



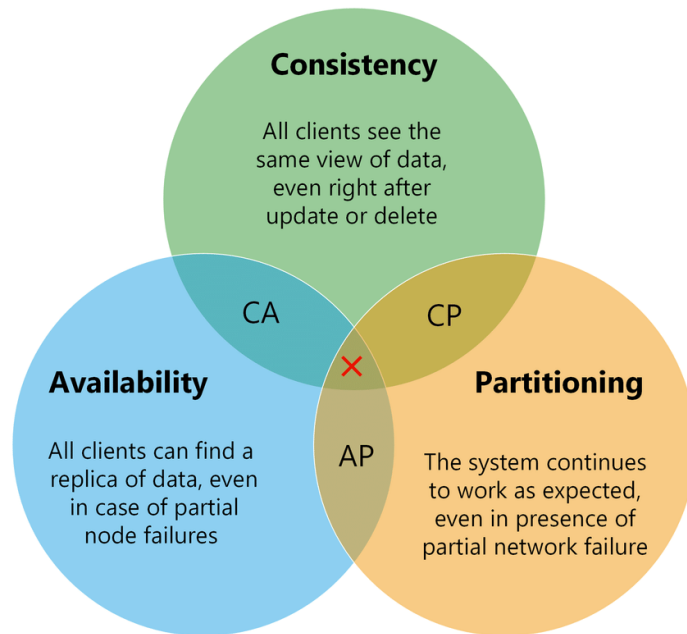
What Caused Hadoop's Decline (Some of)

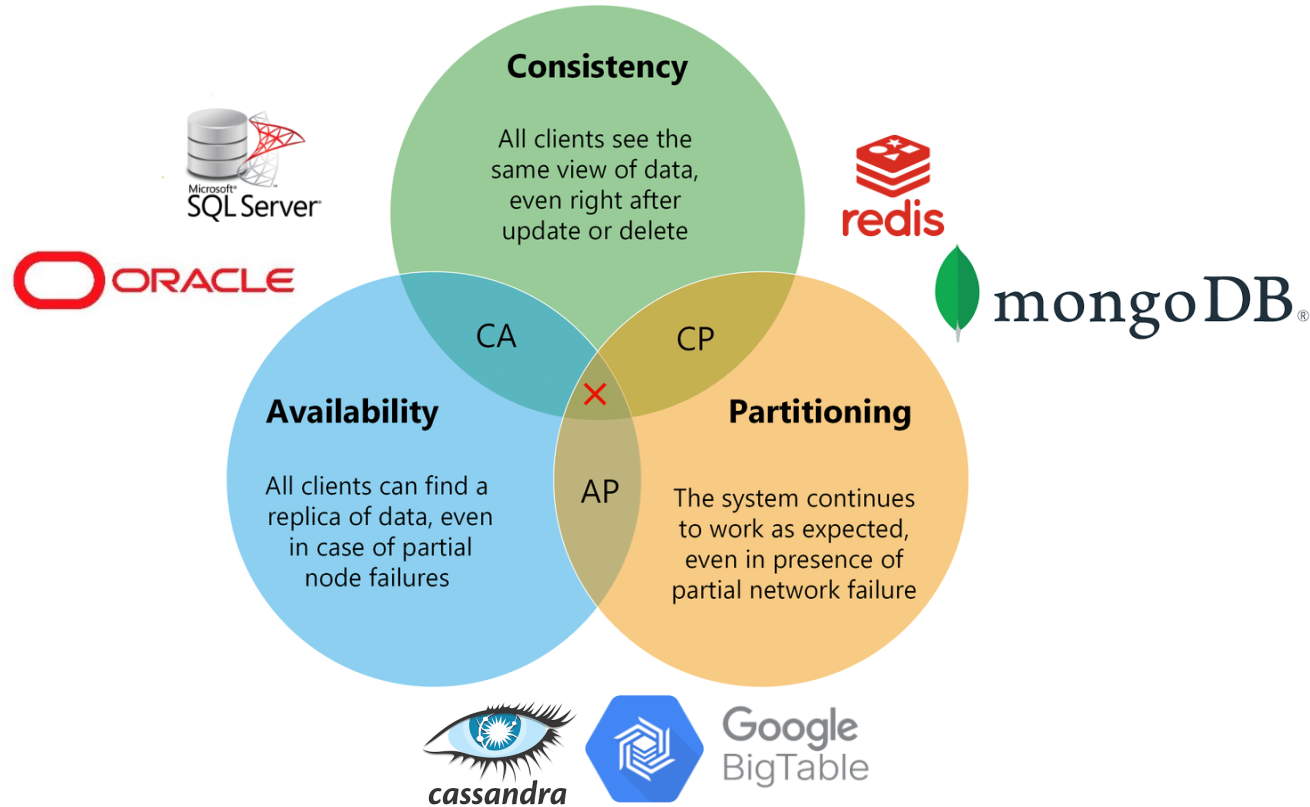
< PROFINIT >

- Complex setup, skilled administrators and developers.
- MapReduce (Java based) is slow and not sufficient for ML or RT processing → Spark
- HDFS is optimal for big data files and not flexible.
- Horizontal scalling lead to huge onprem clusters => hard and expensive to maintain.
- Numerous tools → difficult to make and maintain cohesive data pipelines.
- Weak built-in governance and security tools => hard to be compliant with regulations.
- Developers heavy, not user friendly.

CAP Theorem

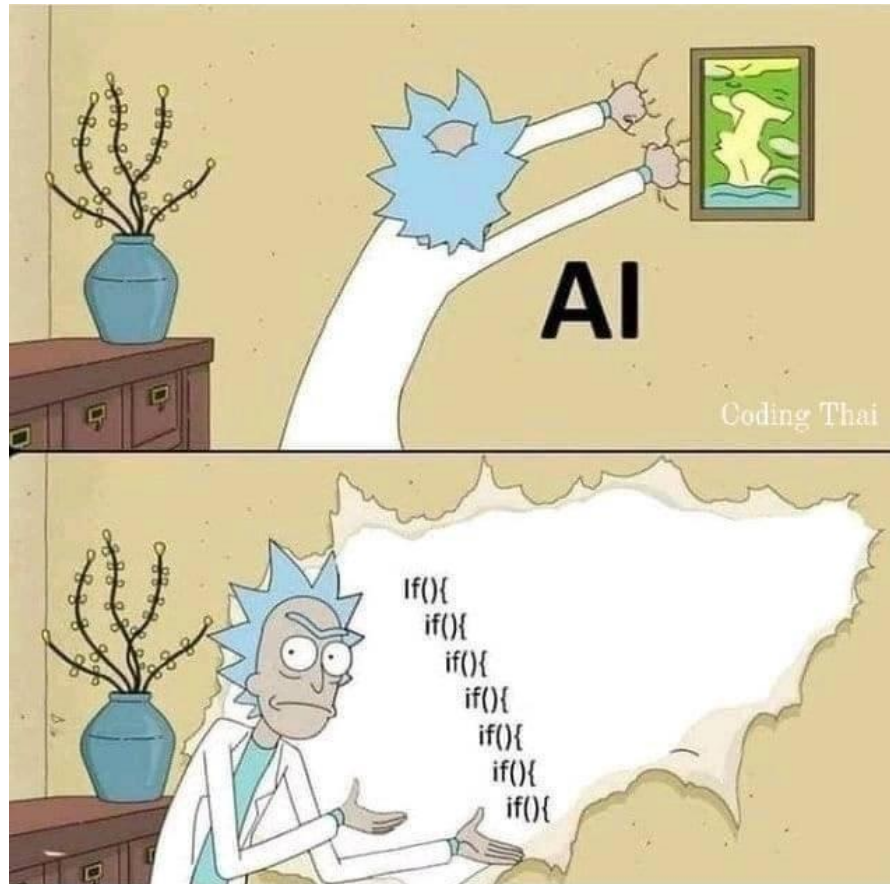
- > What of CAP theorem is applicable to Relational Databases?
 - CA
- > What of CAP theorem is applicable to BigData systems?
 - CP or AP
 - Fault or Consistency tolerant





Conclusion

- > Brief history and brief introduction to big data concepts will show
 - Money are important – GOOGLE, Yahoo, Microsoft, IBM, LinkedIn...
 - Open Source developers need to eat too ☺
 - Every technology has its rise, peak and decline ([Gartner hype cycle](#))
 - Datawarehouse, Hadoop, TV ☺, [even gen AI started to decline](#)
 - Key concept remains – distributed data, distributed processing, fault tolerancy etc.
 - **Take the best and fix issues** – modern data platforms offer security, governance, stream and batch processing, wide integration, great UI, support etc.



Děkujeme za pozornost

Profinit EU, s.r.o.
Tychonova 2, 160 00 Praha 6

Tel.: + 420 224 316 016, web: www.profinit.eu

 LINKEDIN
linkedin.com/company/profinit

 TWITTER
[@profinit_EU](https://twitter.com/profinit_EU)

 FACEBOOK
facebook.com/Profinit.EU

 YOUTUBE
Profinit EU, s.r.o.