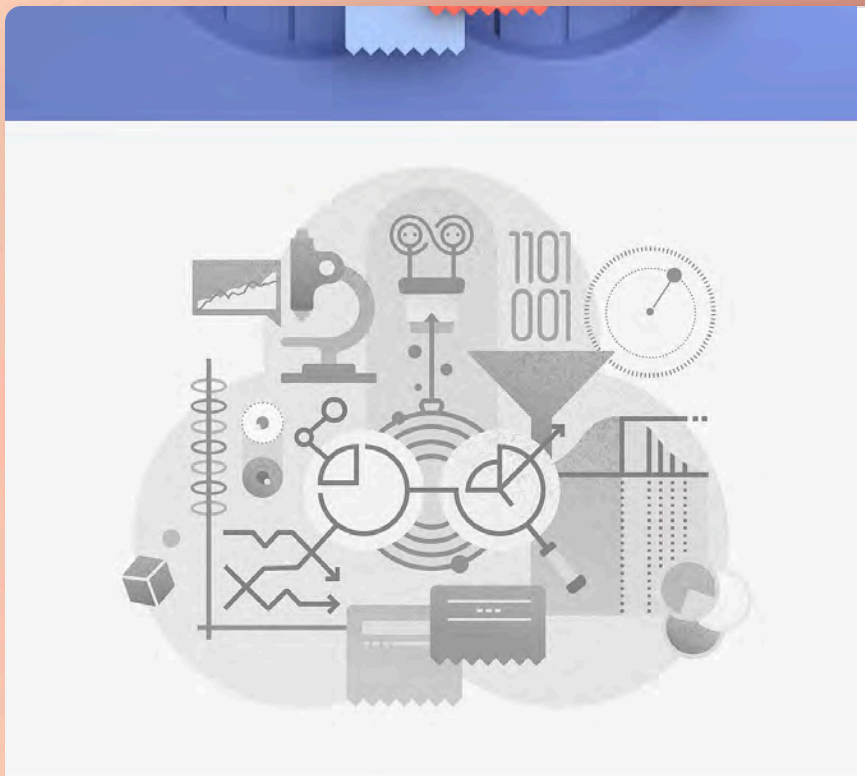
Big Data

An exploration of storage, processing, and the evolution of distributed computing systems

Assoc. Prof. Dorien Herremans

dorienherremans.com



INTRODUCTION

What Is Big Data?

The Data Explosion

- 90% of all data created in last two years
- Incremental growth across industries
- Fundamental shift in computing
- ~180+ zettabytes generated annually in 2025

Common Sources

- Phone tower logs, website analytics
- Medical imaging (X-rays, MRI scans, wearables)
- Stock market transactions
- E-commerce order systems
- Social network behavior patterns
- AI generated content, prompt logs

Challenge: We must develop efficient methods to store and process this unprecedented volume of information.

The Birth of "Big Data"

1

1997

NASA scientists Cox and Ellsworth first coined the term in a research paper

2

The Problem

Visualization datasets exceeded main memory and local disk capacity

3

The Solution

"Acquire more resources" – distributed computing emerges

> "We call this the problem of big data. When data sets do not fit in main memory, or even on local disk, the most common solution is to acquire more resources."
>
> *— Cox & Ellsworth, NASA, 1997*

Modern Definitions of Big Data

Oxford Dictionary

"Data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges."

Business Perspective

"A new attitude by businesses, non-profits, government agencies, and individuals that combining data from multiple sources could lead to better decisions."

Technical View

"The belief that the more data you have, the more insights and answers will rise automatically from the pool of ones and zeros."

Popularized around **2008**, the term evolved from a technical challenge to a strategic business philosophy.

Source: Forbes, "12 Big Data Definitions: What's Yours?"

Characteristics of Big Data

Core Attributes

- **Scale:** Too large for single-machine processing
- **Structure:** Structured, unstructured, or semi-structured
- **Analytics:** Enables predictive insights (more importance recently)

Key Challenges

1. Most data is worthless?
2. Data is created fast?
3. Multiple formats from diverse sources?

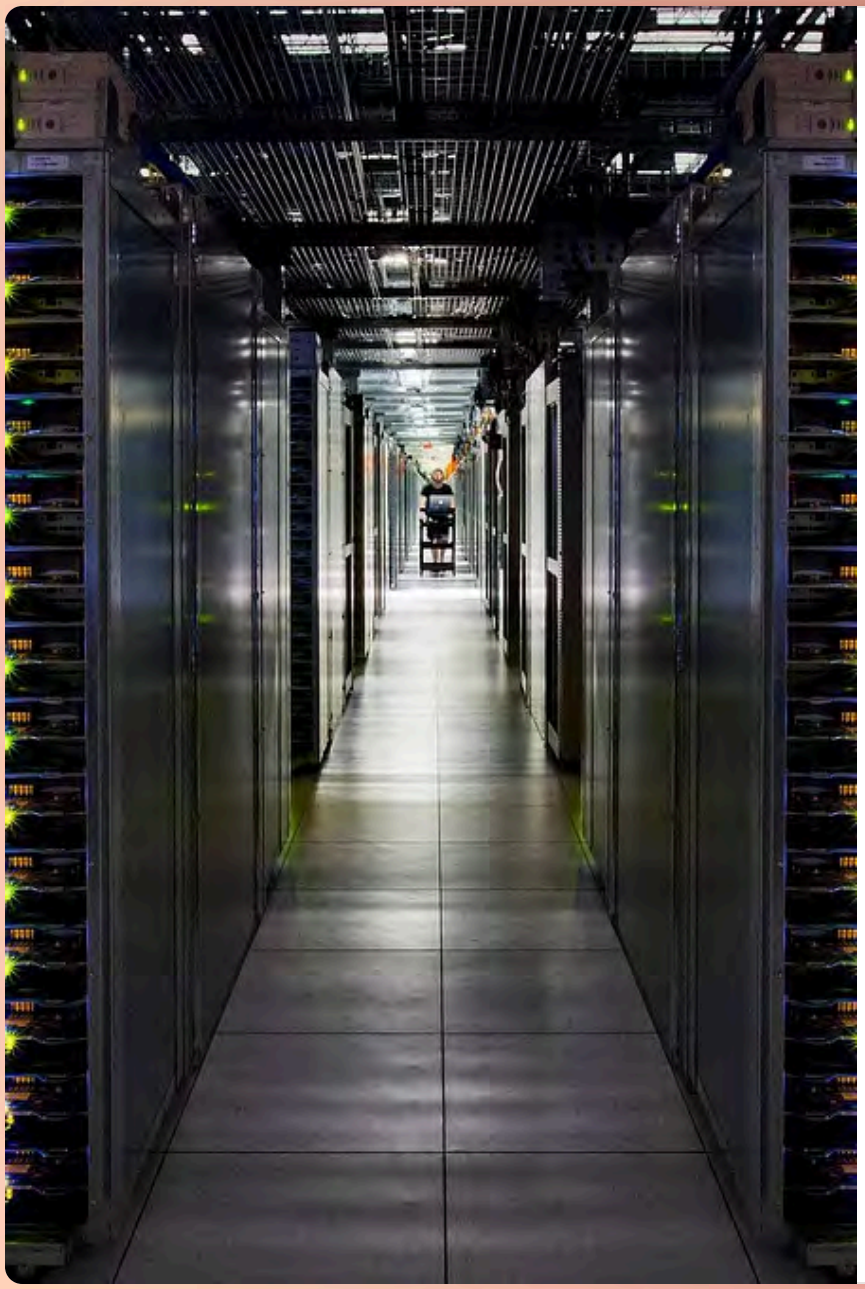
Characteristics of Big Data

Core Attributes

- **Scale:** Too large for single-machine processing
- **Structure:** Structured, unstructured, or semi-structured
- **Analytics:** Enables predictive insights (more importance recently)

Key Challenges

1. Most data is worthless? ❌
2. Data is created fast? ✅
3. Multiple formats from diverse sources? ✅



Understanding the Challenges

Value Extraction

Identifying meaningful insights within massive, noisy datasets, meaningful in future?

Speed Requirements

Processing data at the rate it's generated—often in real-time

Format Diversity

Integrating heterogeneous data from countless sources and systems

APPLICATIONS

Industries Transformed by Big Data

Healthcare & Research

Patient records, genomics, drug discovery

Education

Learning analytics, personalized curricula

Telecommunications

Network optimization, customer behavior

Media

Content recommendations, audience insights

IoT

Sensor networks, smart cities

Finance

Fraud detection, risk assessment

Retail

Inventory optimization, personalization

Manufacturing

Predictive maintenance, supply chain

Energy & Utilities

Grid management, consumption forecasting

All these systems face similar challenges, often described as the **3 V's** of big data.

The 3 V's of Big Data

Volume

The sheer size and scale of data—from megabytes to petabytes and beyond

Variety

Diverse data types and formats from multiple sources and systems

Velocity

The speed at which data is generated, collected, and must be processed

First defined by **Doug Laney** in 2001, these dimensions capture the fundamental challenges of big data systems.

A fourth V—**Veracity** (trustworthiness)—has since been added to account for data quality and reliability issues.

VOLUME

1. Volume: The Scale Challenge

180+

Zettabytes in 2025

Equivalent to 11.25 trillion fully-loaded 16GB iPads

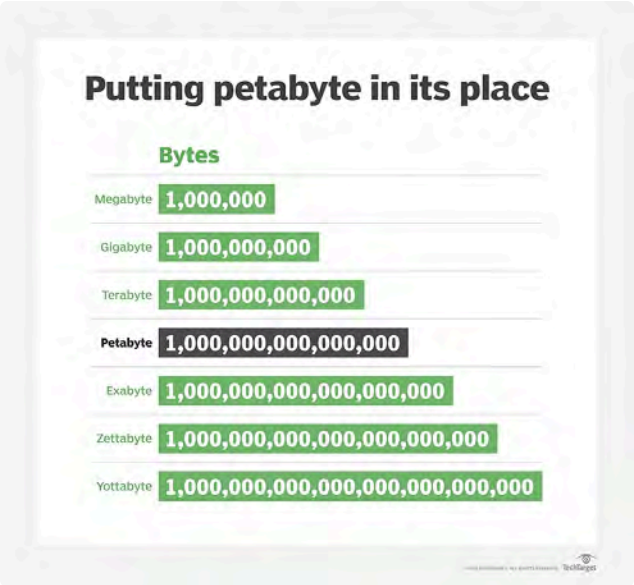
100K x

Cost Reduction

1980: 1GB = $100,000
2017: 1GB < $0.008

Data Growth Trajectory

- From megabytes to petabytes to yottabytes
- Per-capita storage capacity doubles every 40 months
- Internet of Things: cameras, sensors, RFID readers



Storage Economics & Challenges

1

Affordable Storage

Commodity hardware makes storing massive datasets economically feasible

2

Reliability Costs

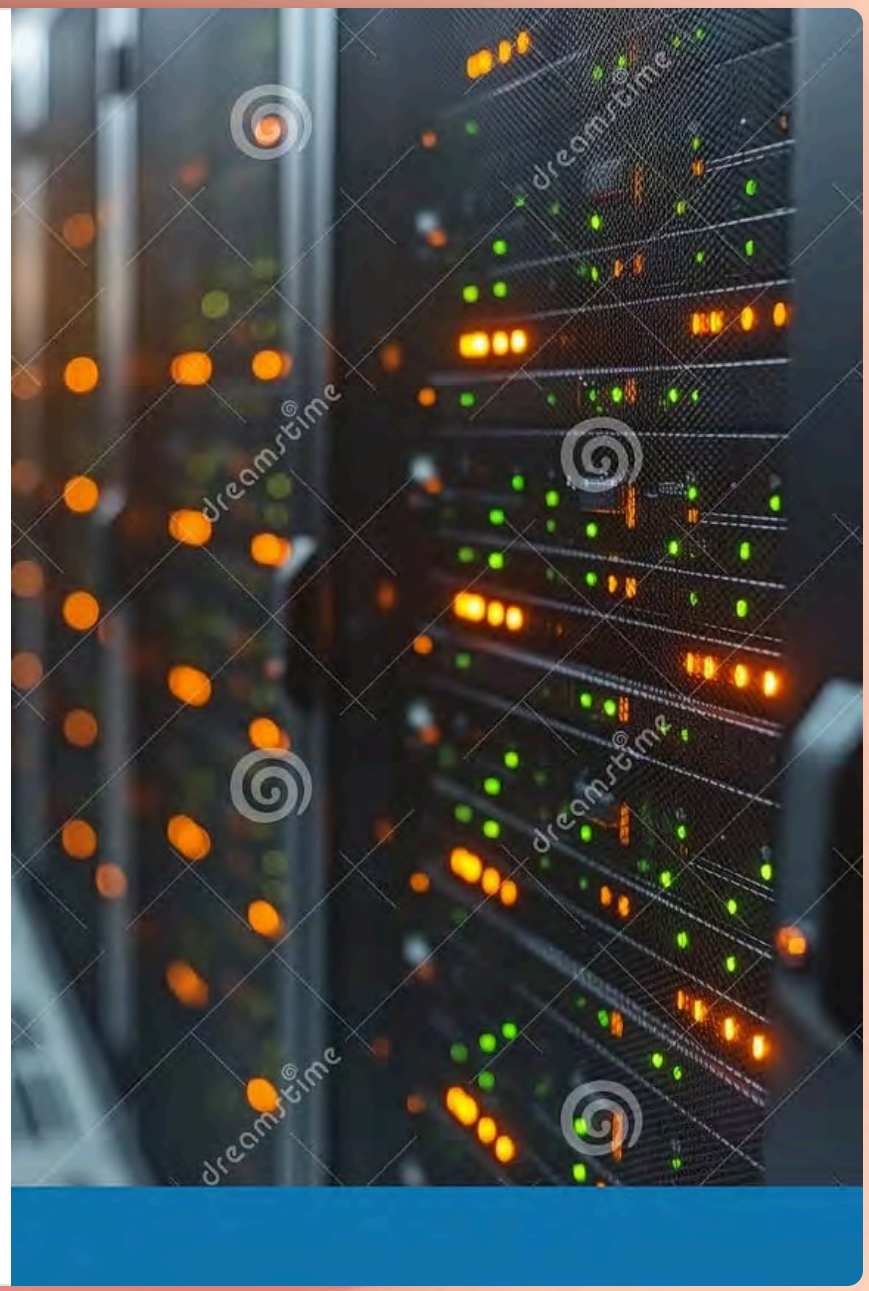
Enterprise-grade solutions (like SANs) remain expensive for mission-critical data

3

Network Bottleneck

Streaming data over networks becomes the primary constraint

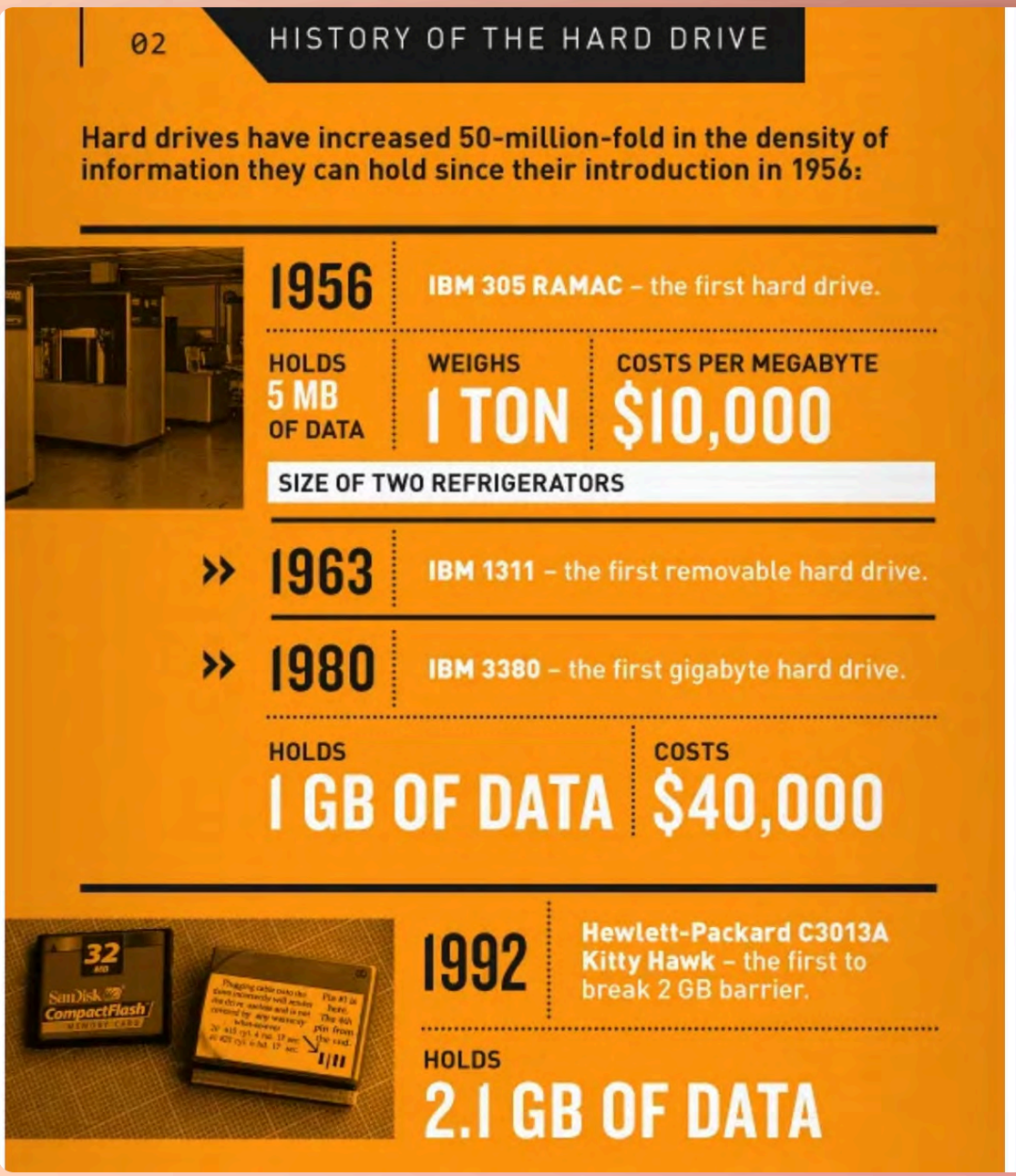
Core Challenge: We need cheap ways to store, read, and process data at scale. While storage is now affordable, efficiently moving and processing this data remains complex.



Data unit examples: Kilobyte to Yottabyte

Interactive graph exploring data scale:

Infograph

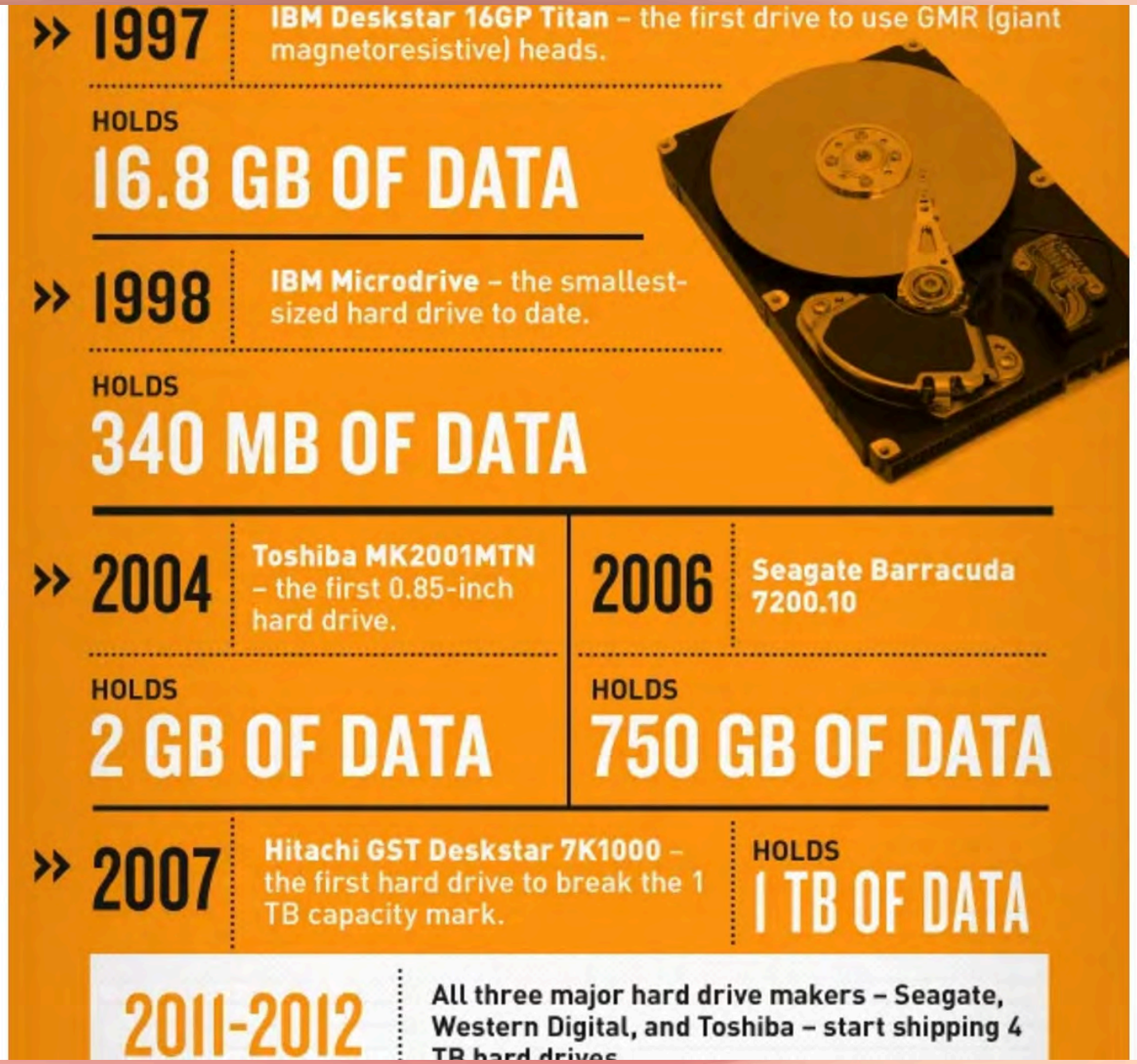


Data Units: Exabyte and Beyond

Global internet traffic, climate modeling, and astronomical data reach exabyte scale

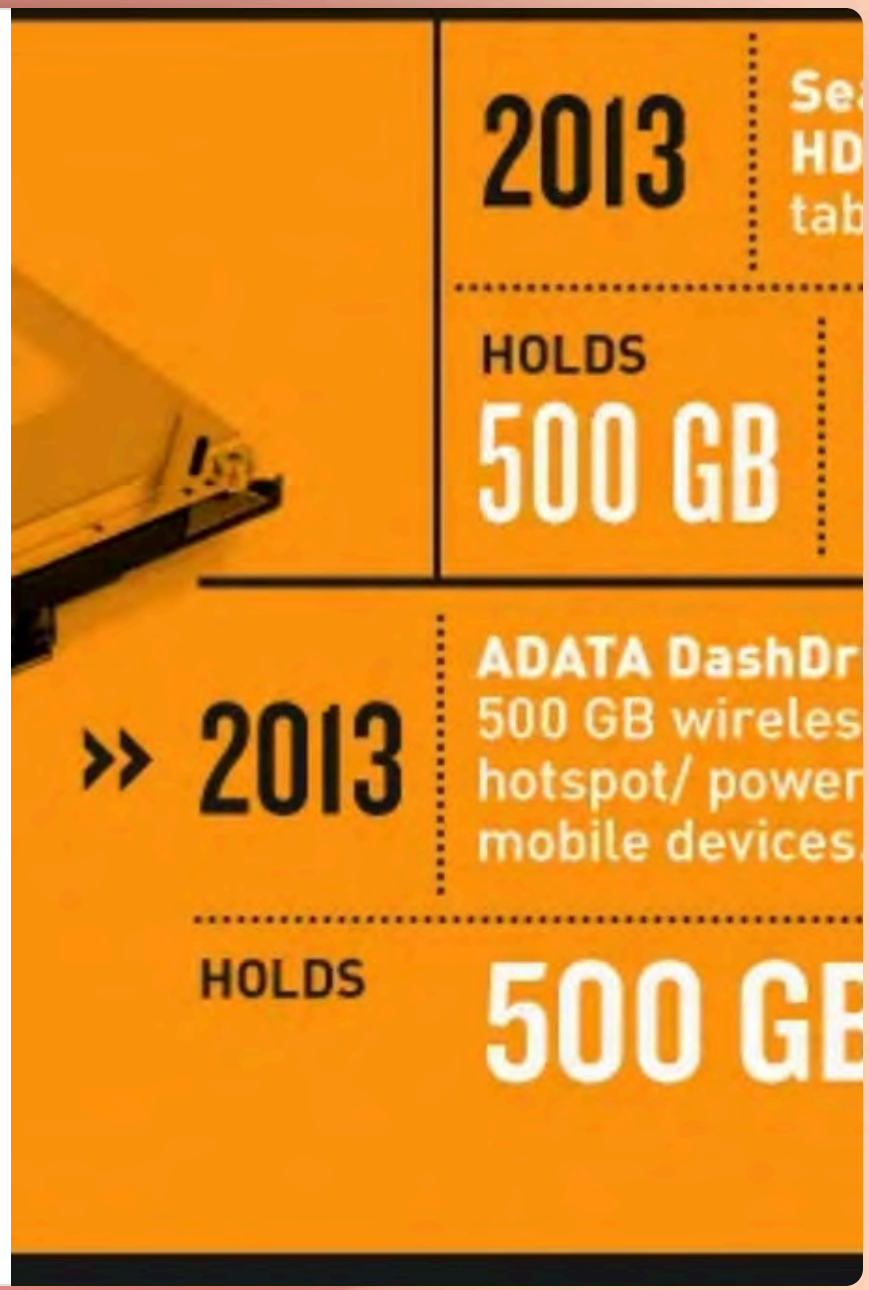
Data Units: Zettabyte Era

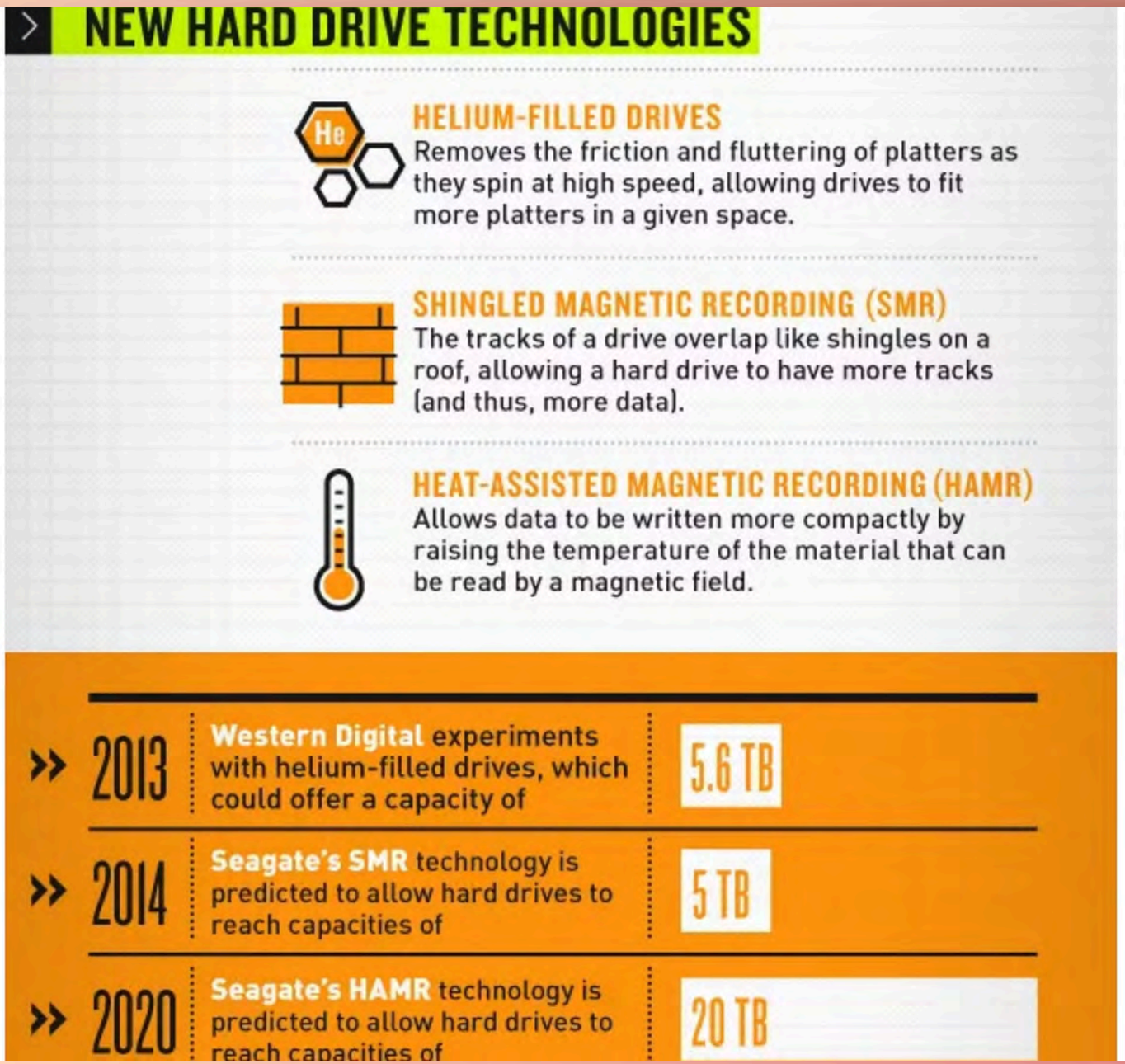
We now measure global data creation in zettabytes annually—unimaginable scale



The Scale of Modern Data

Visualizing the exponential growth from bytes to yottabytes





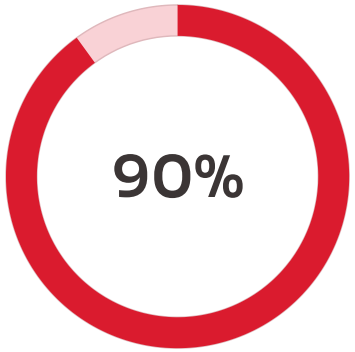
Data Size Matters

New technologies are enabling larger hard disk spaces.

- **Largest available HDD (enterprise): ~36 TB**, Seagate *Exos M* series drives.
- **Consumer models 2026:** 20 TB–32 TB models - often used in NAS or archival setups.
- **In development:** manufacturers like Seagate have roadmaps targeting **60 TB and even 100 TB HDDs by ~2030** for hyperscale storage needs.

VARIETY

2. Variety: Beyond Structured Data



Unstructured Data (or semi-structured)

Estimated percentage of all data that lacks predefined structure

Traditional Approach

- SQL: highly structured, predefined schemas
- Efficient for transactions
- Limited flexibility

Modern Reality

- Logs, scans, receipts in varying formats
- Audio preserves tone, context
- Need to store **raw data** for future processing

Hadoop's advantage: Store data first, define structure later—enabling flexibility and preserving rich information.

Three Types of Data Structures

1

Structured Data

Strict format with predefined schema—essential for operational systems like SQL databases and transaction processing

2

Unstructured Data

No predefined model—text, numbers, files mixed together. Often most valuable for analytics and data mining. Stored in systems like Hadoop

3

Semi-Structured Data

Flexible schema where each object may have different attributes. Self-documenting formats like JSON and XML. Used in systems like MongoDB

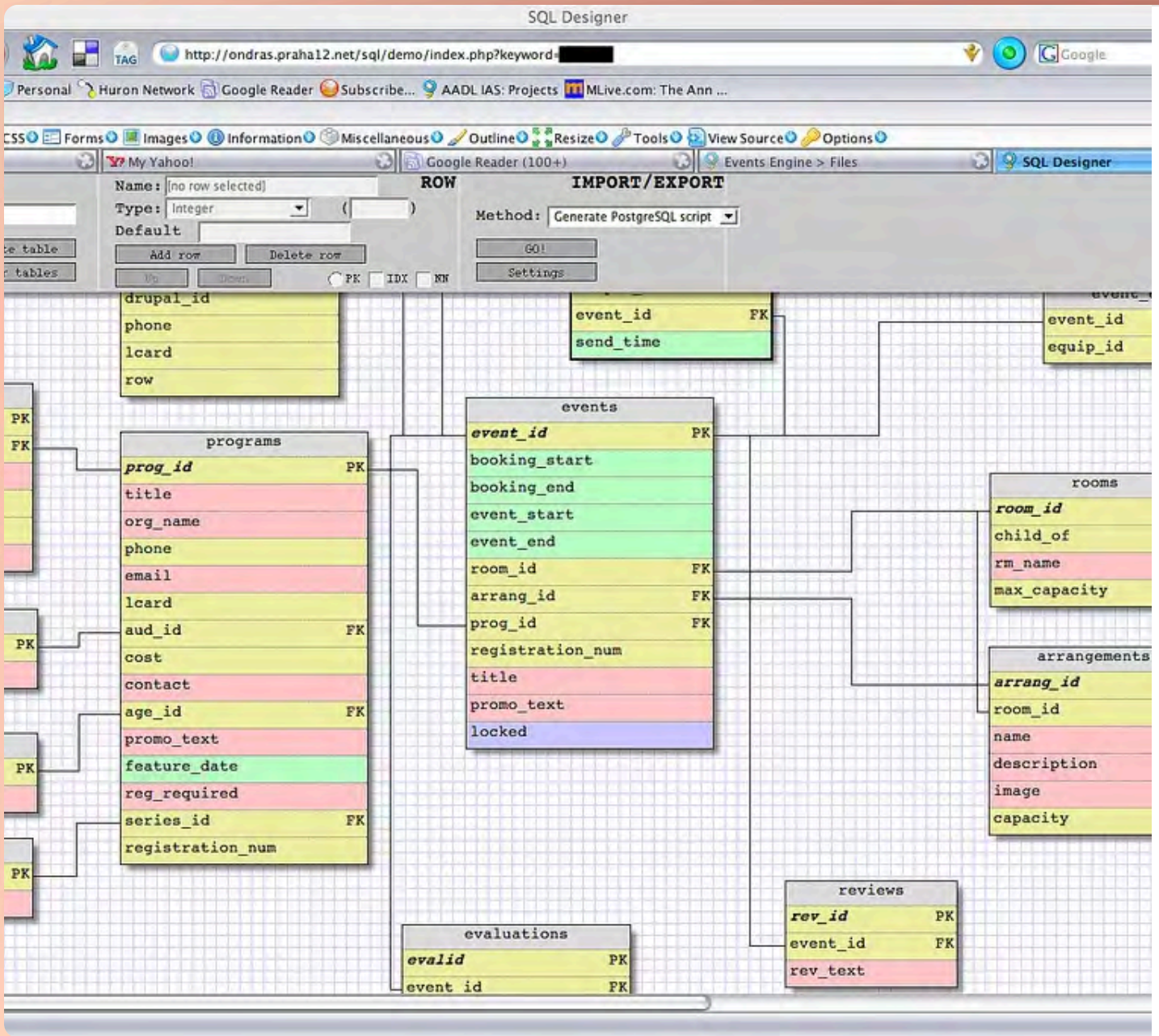
JSON: Semi-Structured Data Example

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Key Characteristics

- No fixed format
- Semi-structured
- Key-value pairs with hierarchical structure
- Smaller file size for faster transfer
- **Self-documenting** and human-readable
- "Friendly" alternative to XML

Learn more: **AWS: JSON vs XML comparison**



SQL: Structured Relational Database

Traditional relational structure: predefined tables, rows, and columns with strict schemas ensuring data integrity

Transactional Data Example

Structured transaction logs: each record follows a consistent format with defined fields and data types

Transaction_Details

ID	Transaction_Header_ID	Gross_Amount
5434125	4526622	-300.00
5420422	4513202	279.72
5415415	4508216	300.00
5413258	4506059	279.72
5434502	4526964	-279.72
5415438	4508239	279.72

Transaction_Headers

ID	Payment_Type_Id	GL_Date
4526622	26	2013-10-10 00:00:00.000
4513202	8	2013-10-05 00:00:00.000
4508216	1	2013-12-01 00:00:00.000
4506059	1	2013-01-01 00:00:00.000
4526964	26	2014-12-09 00:00:00.000
4508239	1	2014-12-09 00:00:00.000

Bill_Mapping

ID	Trans_ID_Dr	Trans_ID_Cr	Amount
2865991	5420422	5434125	279.72
2865992	5415415	5434125	20.28
2866486	5415438	5434502	279.72

Tweets | Tweets & replies | Media | Likes

Dorien Herremans @dorienherremans · Aug 27

Congrats to @ravencheuk for publishing nnAudio in IEEE Access. Anyone interested in a GPU toolbox for on-the-fly spectrogram extraction should check it out #ai #dsp #audio #ismir #music #signalprocessing #PyTorch dorienherremans.com/content/nnaudi...

1 7

Dorien Herremans @dorienherremans · Aug 12

We have a job opening for a wizard in NLP/sequential models for finance #lstm #pytorch #bert #finance #fintech #singapore dorienherremans.com/content/resear...

3

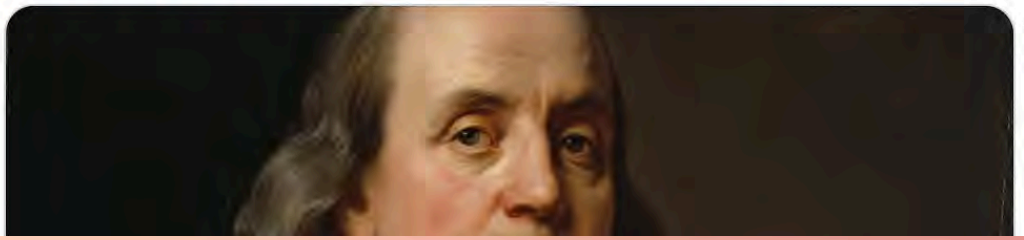
Dorien Herremans @dorienherremans · Jul 23

Nice PyTorch walk through for attention #attention #pytorch #ai #nlp nlp.seas.harvard.edu/2018/04/03/att...

1

Dorien Herremans @dorienherremans · Jul 23

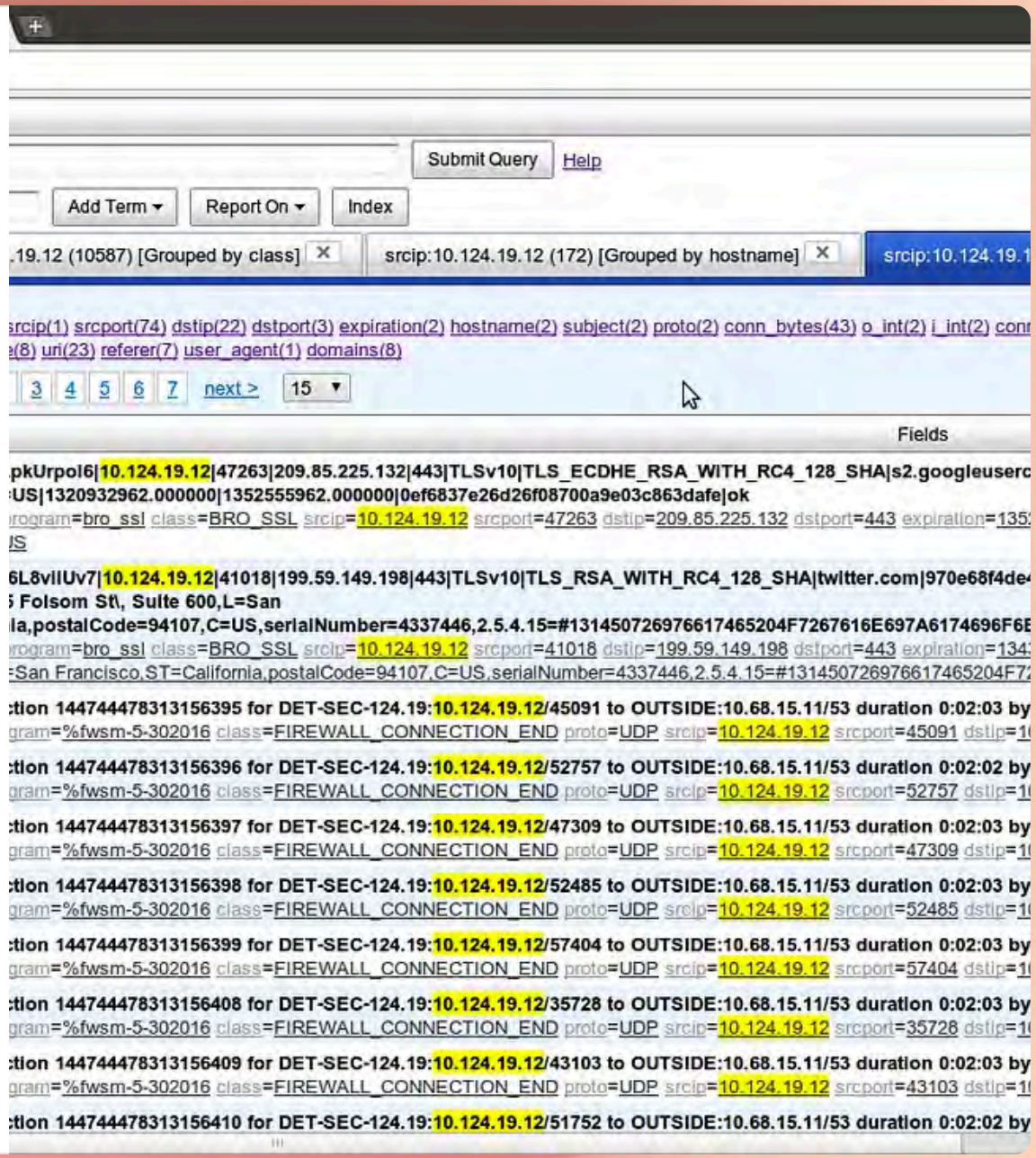
Nice text on going from word2vec to transformer models #ai #nlp #word2vec #bert #machinelearning



Social media data

Server IP Logs

Semi-structured log files: consistent patterns but flexible format allowing for varying fields and metadata



Variety in Action: Logistics Example

Delivery Optimization Problem

To assign the next pickup, which truck should we dispatch? The decision requires integrating multiple data sources:

...

Variety in Action: Logistics Example

Delivery Optimization Problem

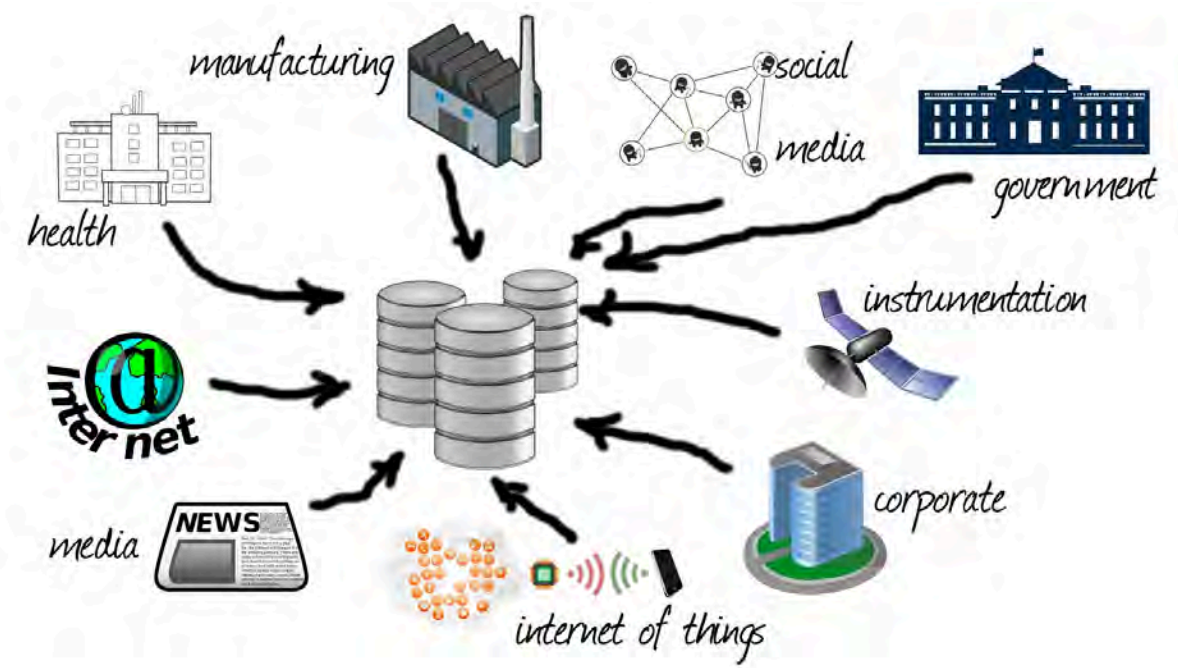
To assign the next pickup, which truck should we dispatch? The decision requires integrating multiple data sources:

- **Real-time GPS location** of all available trucks
- **Current map data** accounting for road types and speed limits
- **Live traffic conditions** from multiple providers
- **Current load weight and volume** in each vehicle
- **Fuel efficiency metrics** based on vehicle and load
- **Available cargo space** and package dimensions

Simply choosing the *closest* truck ignores critical context. True optimization requires synthesizing diverse data types in real-time.

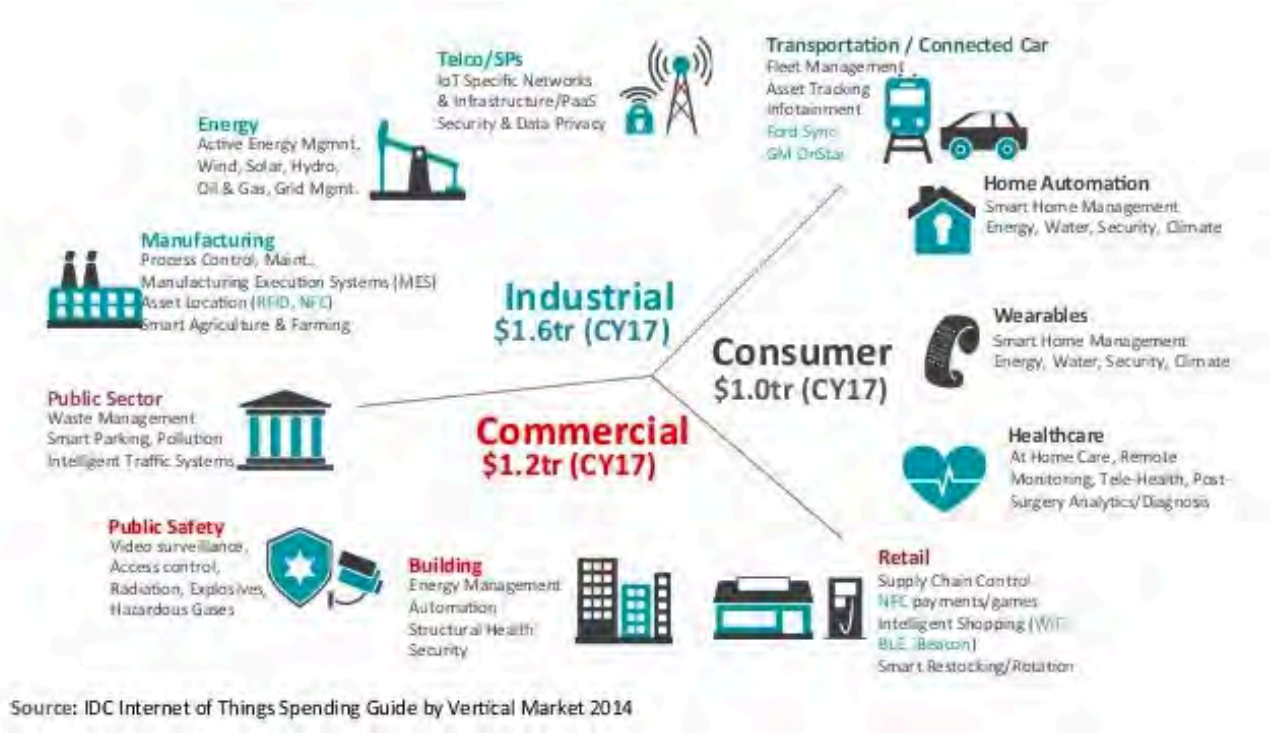
Diverse Data Sources

Data flows from countless sources: sensors, social media, transactions, logs, and user interactions—each with unique formats and update frequencies



The Internet of Things Revolution

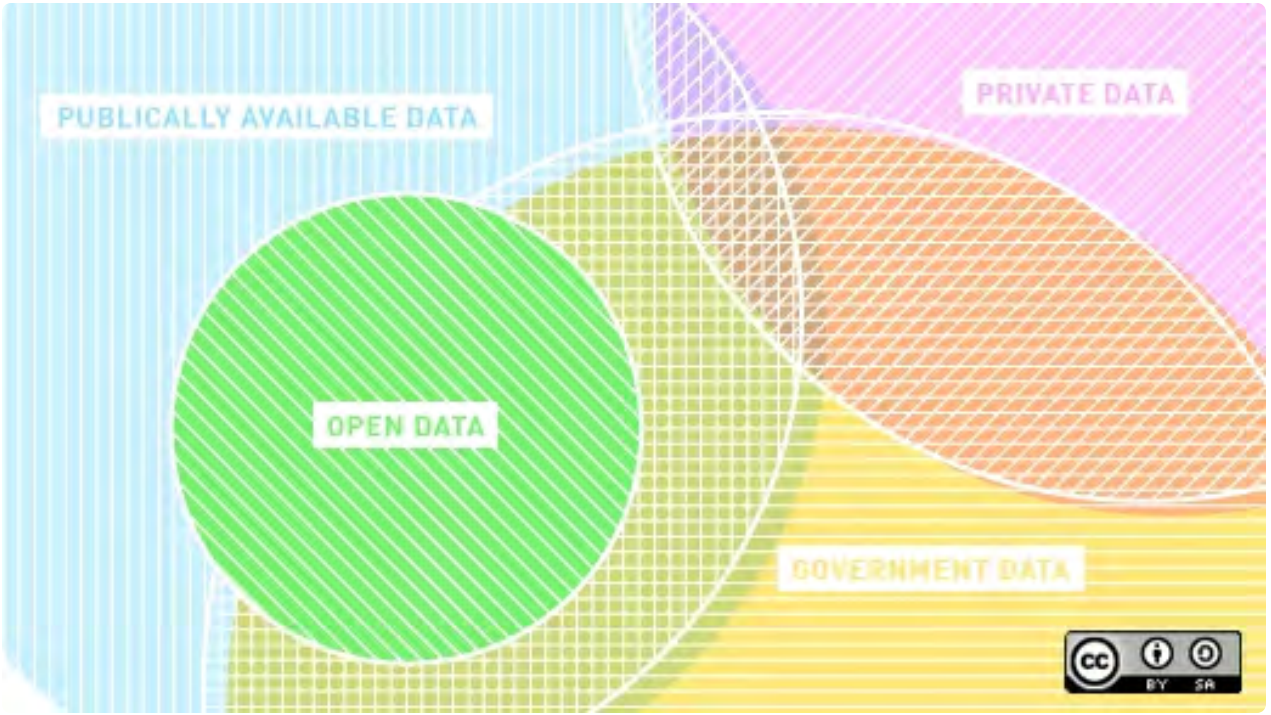
Connected devices generate continuous streams of sensor data—from smart homes to industrial equipment to wearable health monitors



Source: IDC Internet of Things Spending Guide by Vertical Market 2014

Open Data Sources

High-quality datasets are increasingly accessible through **government** portals, **academic repositories,** Kaggle, and platforms like Hugging Face and Zenodo.



VELOCITY

3. Velocity: The Speed Challenge

Several

100s of Exabytes Per Day

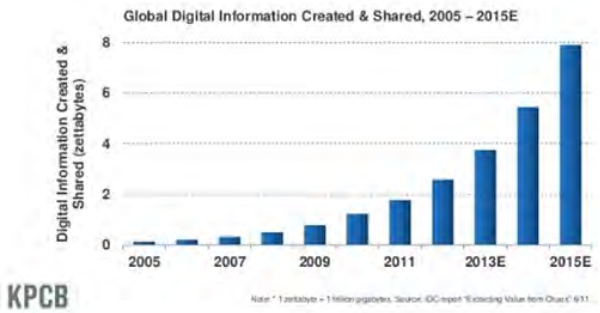
Global data creation rate (1 exabyte = 1 billion gigabytes)

Storage Requirements

- Must capture data **as it arrives** in real-time
- Cannot afford to lose transient information
- Systems must **scale with generation speed**

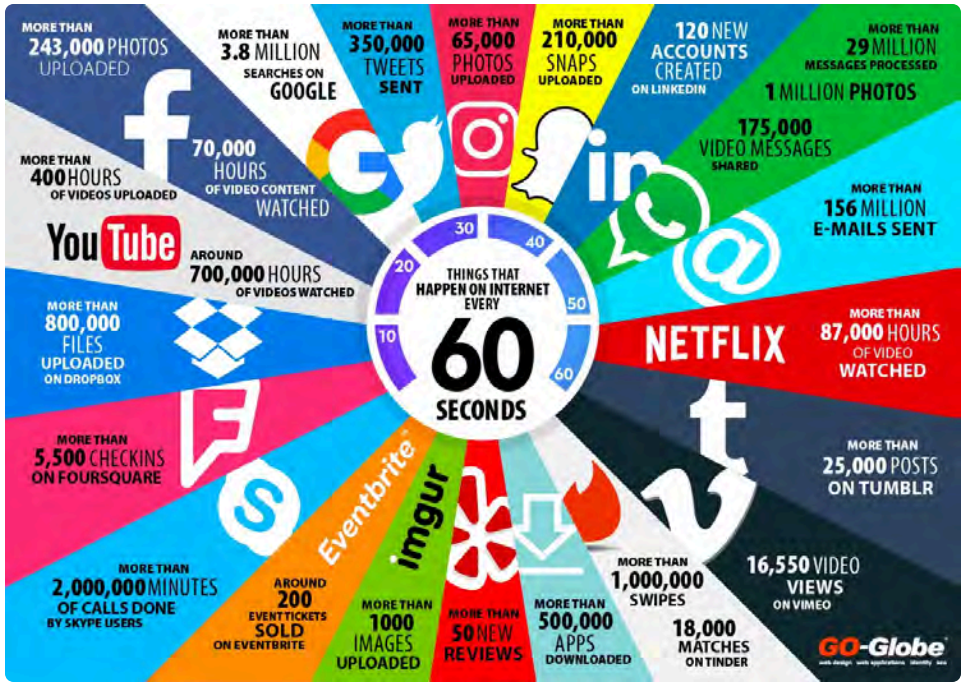
Use Case: Recommendations

E-commerce platforms need immediate access to browsing patterns, purchase history, and inventory to deliver personalized product suggestions



What Happens in 60 Seconds?

2016 infographic visualizing the astonishing volume of activity occurring every single minute across the internet



Source: https://www.go-globe.com/60-seconds/

VERACITY

The Fourth V: Veracity

Data Uncertainty

Varying levels of reliability and trustworthiness across sources

Interpretation Challenges

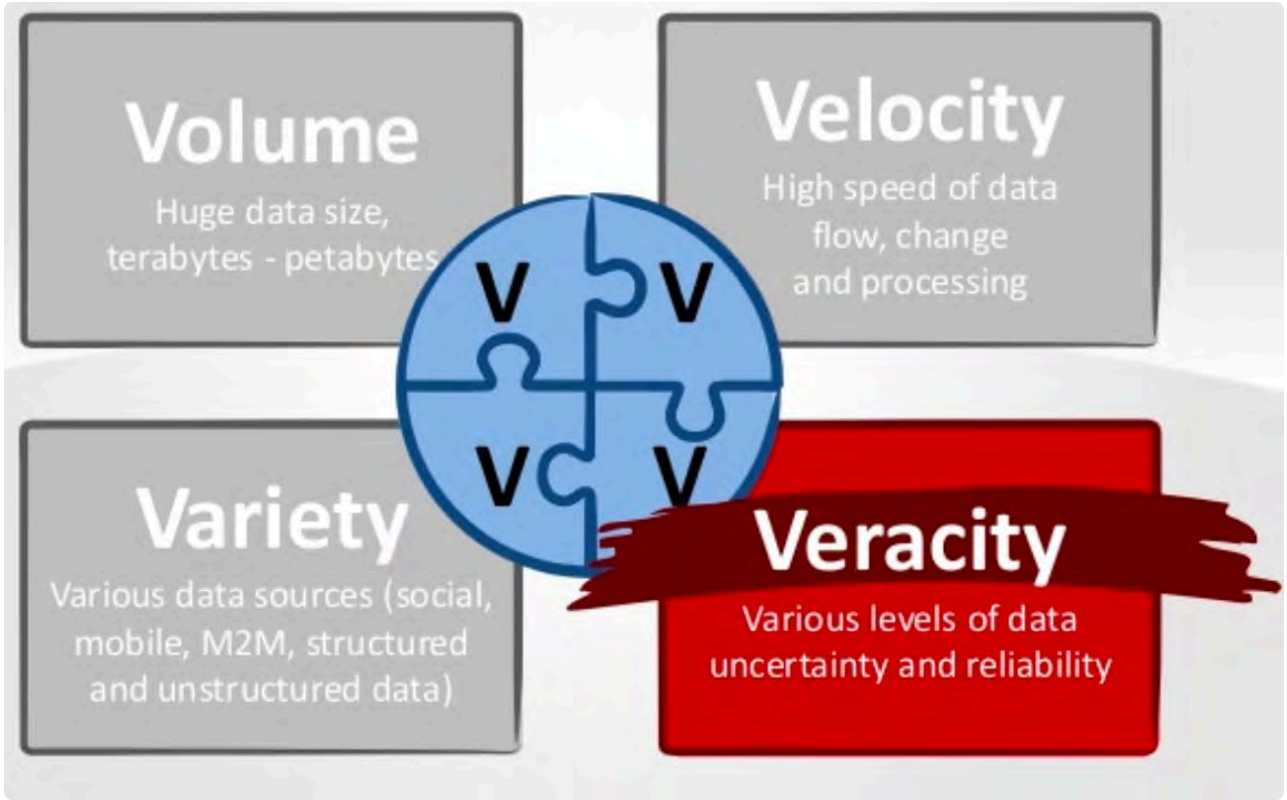
Unstructured data (text, images, speech) contains **ambiguity/double meaning** and imprecision

Quality Issues

Technical problems: inconsistent **formats**, outdated sources, incomplete records

> Structured data offers **certainty and precision**. Unstructured data requires **fuzzy interpretation**—a text message may have double meanings, an image may be ambiguous.

Some now add a **5th V**: Value—because data is only as useful as the insights it generates.



Source: InfoDiagram

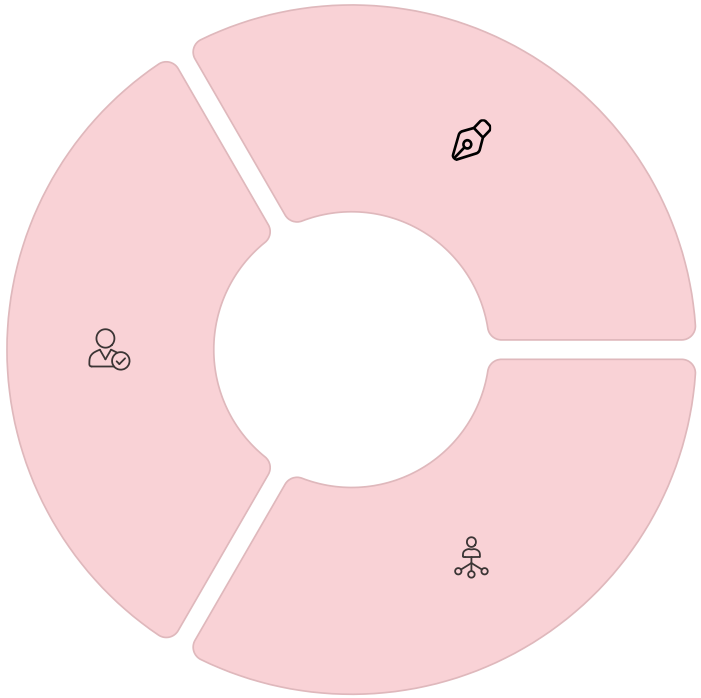
CAP THEOREM

CAP Theorem: The Impossible Triangle

> **Eric Brewer's Principle:** "It's impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees"

Consistency

Every read receives the most recent write or an error response



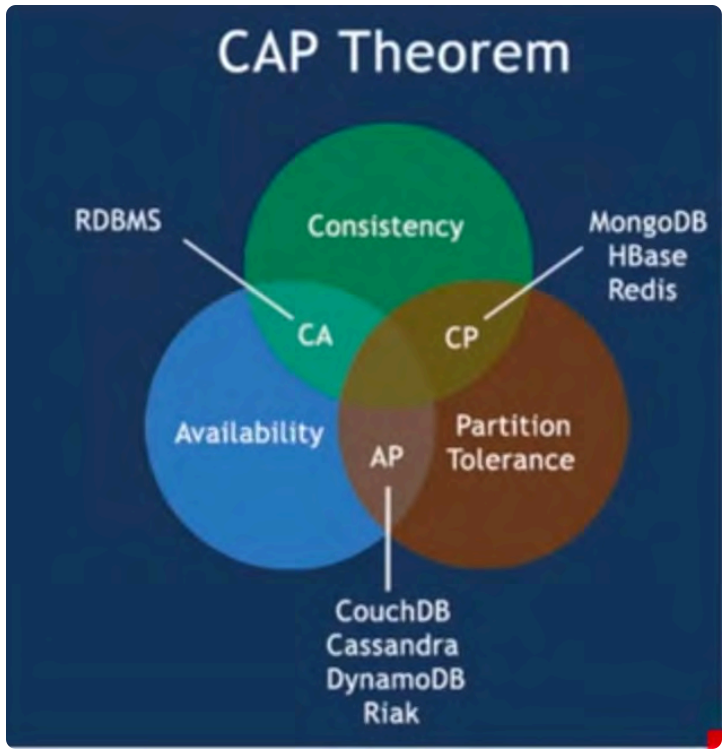
Availability

Every request gets a non-error response (without guarantee of recency)

Partition Tolerance

System operates despite network failures or message delays between nodes

CAP Theorem Trade-offs



CA Systems

Single cluster—all nodes always connected. Partitions cause temporary inconsistency until resolved

CP Systems

Some data may be inaccessible during partitions, but accessible data remains consistent and accurate

AP Systems

System stays available under partitioning, but responses may contain stale data. Resynchronizes after partition resolves

PACELC: Expanding CAP

> **Daniel Abadi's Extension:** "**If** there is a partition (P), how does the system trade off availability and consistency (A and C); **else** (E), when running normally without partitions, how does the system trade off latency (L) and consistency (C)?"

Key Insight: In distributed systems with replication, there's *always* a tradeoff between consistency and latency—even when everything is working correctly.

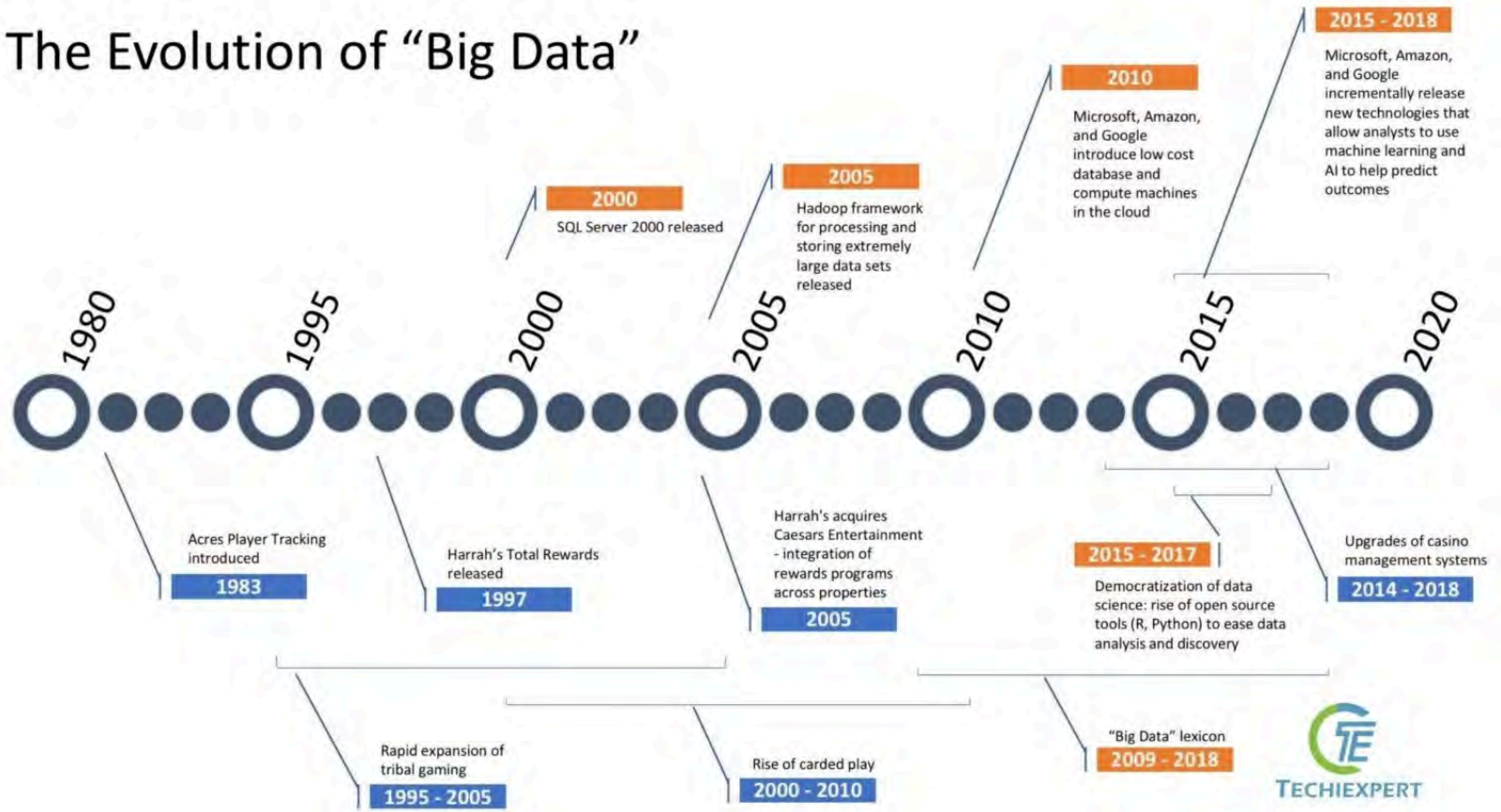
High availability requires data replication. Once you replicate, you must choose: **fast responses** (latency) or **guaranteed accuracy** (consistency).

PACELC Framework

Classification of distributed databases based on their PACELC trade-offs during both partition and normal operation scenarios

DDBS	P+A	P+C	E+L	E+C
DynamoDB	Yes		Yes[a]	
Cassandra	Yes		Yes[a]	
Cosmos DB	Yes		Yes	
Riak	Yes		Yes[a]	
VoltDB/H-Store		Yes		Yes
Megastore		Yes		Yes
BigTable/HBase		Yes		Yes
MongoDB	Yes			Yes
PNUTS		Yes	Yes	
Hazelcast IMDG[6]	Yes		Yes	Yes

The Evolution of “Big Data”



& MapReduce

The foundational framework for distributed big data processing

CHAPTER 2

The Origins: Google's Influence

2003: Google File System

1. Google needed a scalable storage solution to handle vast amounts of web data.
2. GFS introduced a distributed storage model that could store and retrieve huge datasets efficiently.

1

2

2004: MapReduce

1. Google introduced MapReduce, a parallel programming model that enabled distributed computation over large datasets.
2. It provided a way to process data across thousands of servers.

These papers laid the conceptual foundation for what would become the Hadoop ecosystem.

Hadoop Begins: The Nutch Project

The Challenge

Doug Cutting and Mike Cafarella were building **Nutch**, an open-source **web search engine**, and needed scalable infrastructure to store **massive amounts of web data**.

The Solution

Inspired by Google's research, they built:

- A GFS-like distributed file system
- A MapReduce implementation

2006 Milestone: Hadoop spun off from Nutch as an independent Apache Software Foundation project

Yahoo! Adopts and Expands Hadoop

Early Adoption

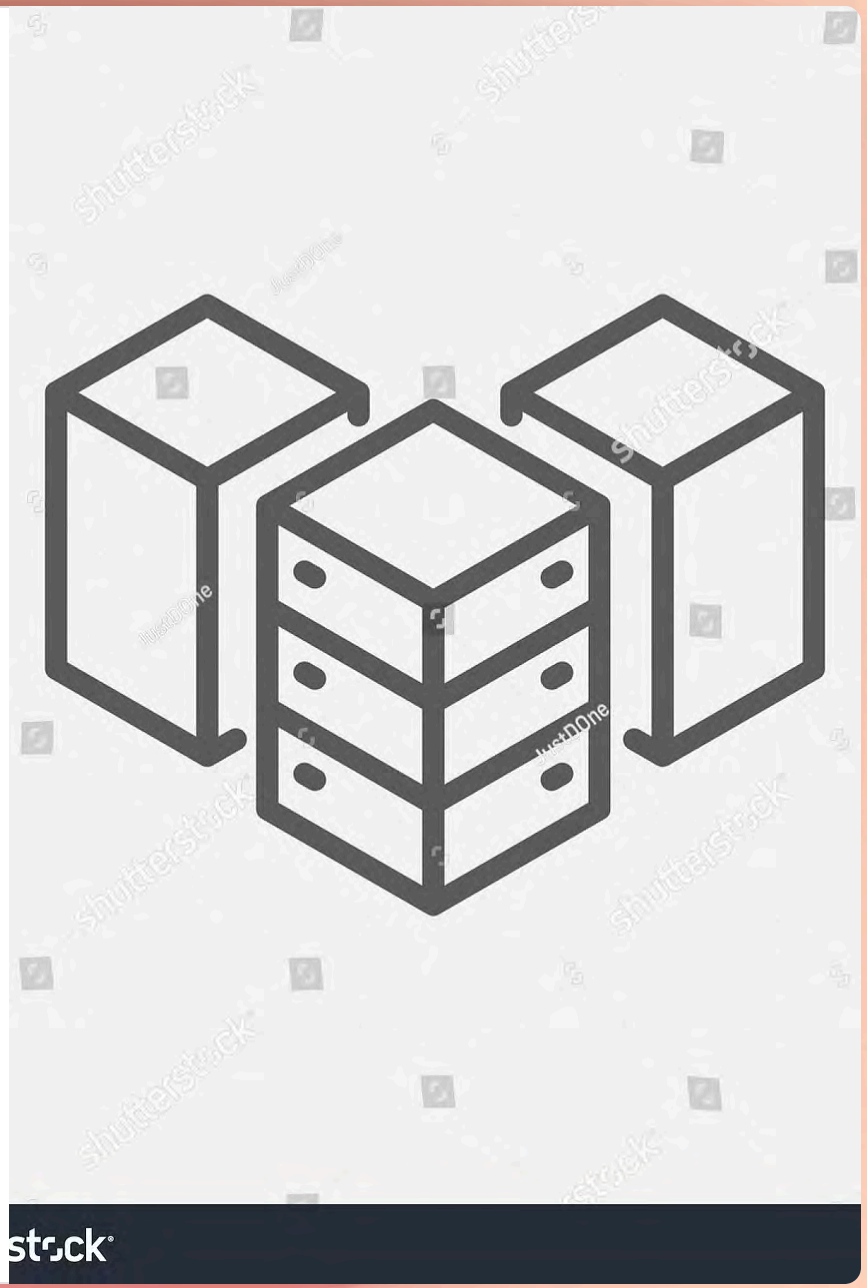
Yahoo! became the first major company to deploy Hadoop for search and data-intensive applications
→ used it to power their search engine

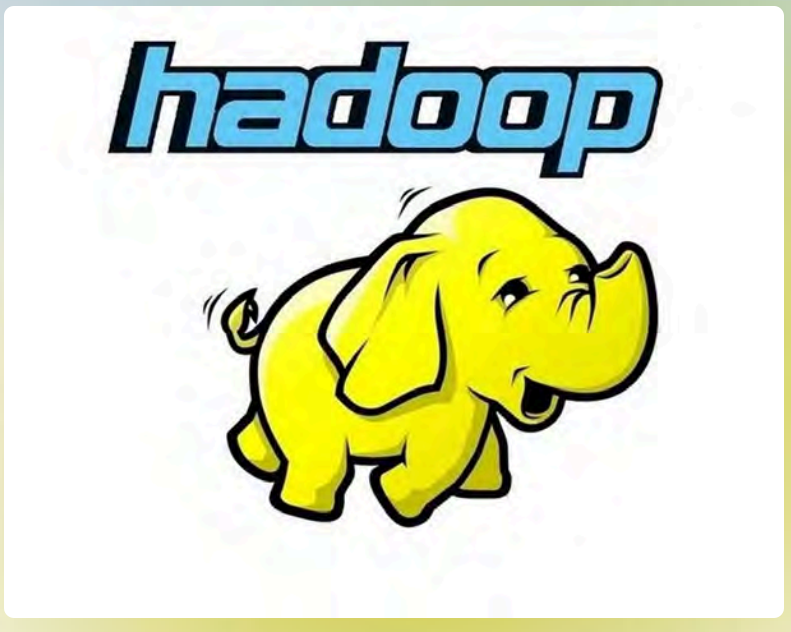
2008: Proof of Scale

Yahoo! successfully ran a 10,000-core Hadoop cluster, demonstrating enterprise viability

Apache Top-Level Project

Hadoop officially became a top-tier Apache project, solidifying its open-source future





What Is Hadoop?

> "An open-source software framework used for **distributed storage and processing** of big data using the MapReduce programming model."

Storage: HDFS

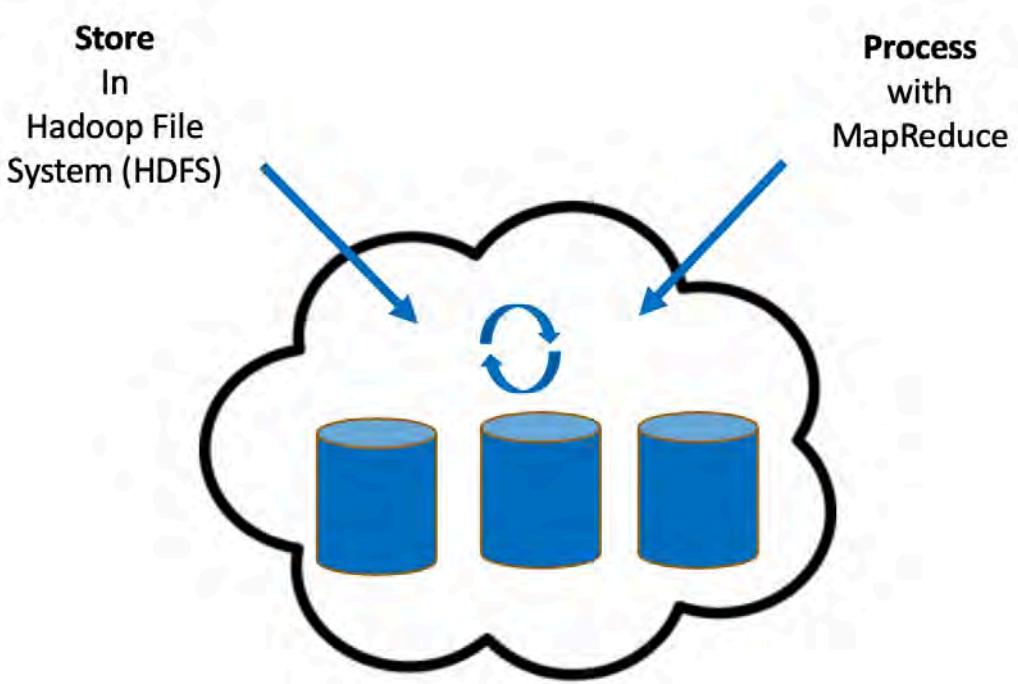
Hadoop Distributed File System—fault-tolerant, scalable storage

Processing: MapReduce

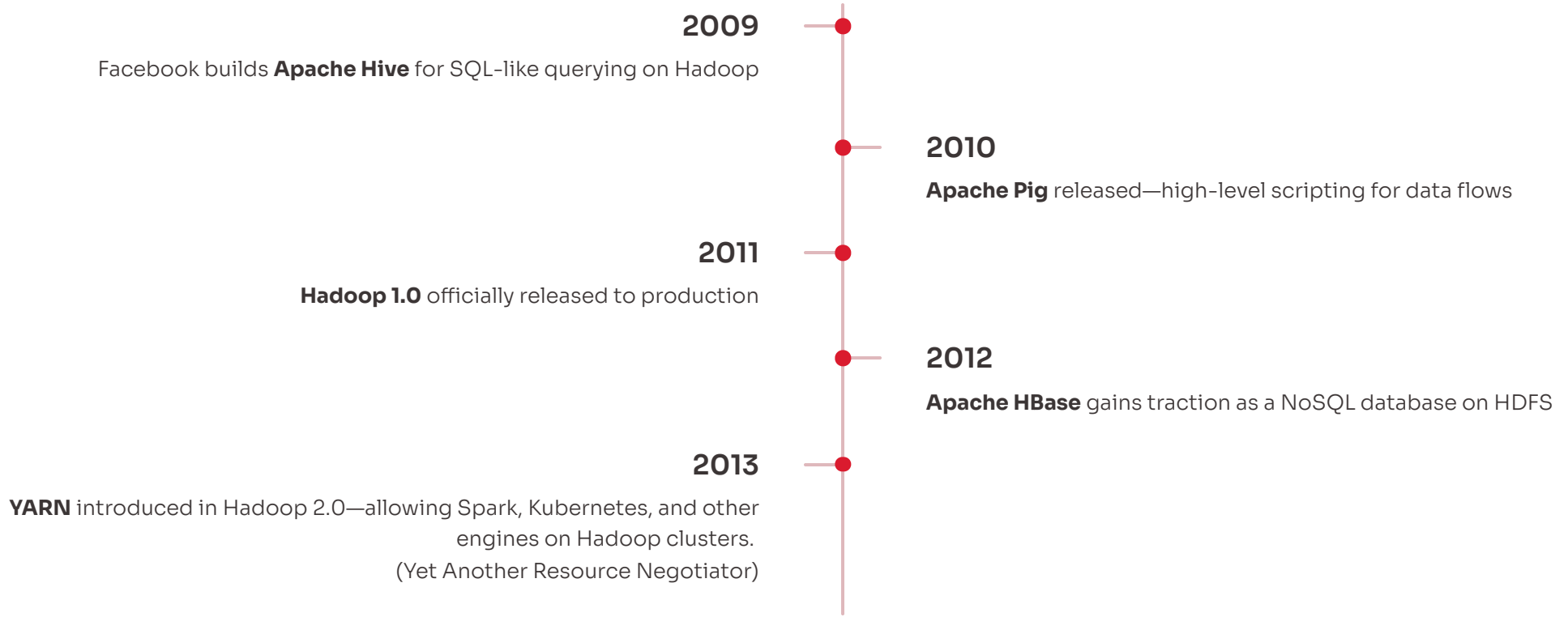
Processes data where it's stored (data locality principle)

Fun fact: Named after Doug Cutting's son's toy elephant. Coded in Java.

Core Hadoop



Hadoop Ecosystem Grows



Hadoop Ecosystem Components

Impala

Query data with SQL directly—no MapReduce overhead

Sqoop

Transfers data from SQL databases into HDFS as delimited files

Hue

Graphical web interface for cluster management and queries

Mahout

Machine learning library for scalable algorithms

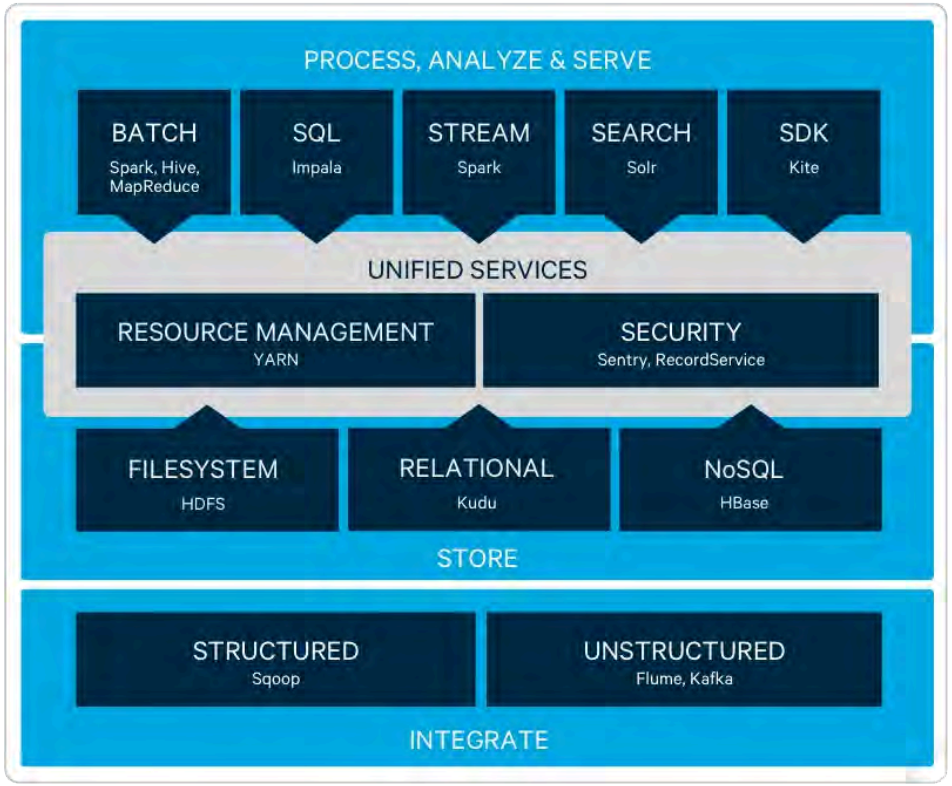
HBase

Real-time NoSQL database built on top of HDFS

Cloudera Hadoop Distribution (CDH) integrates all these tools into a unified platform.

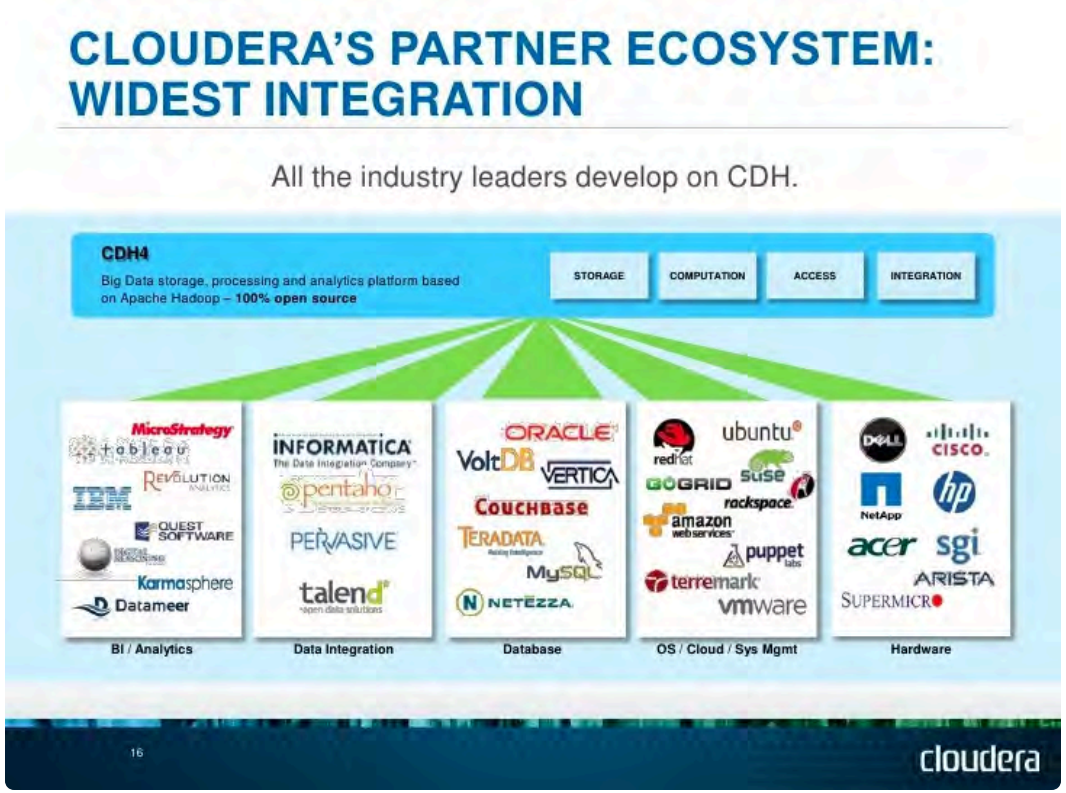
Cloudera Ecosystem Overview

A comprehensive stack showing how Cloudera integrates storage, processing, and analytics tools



Cloudera Architecture Layers

Detailed view of the layered architecture from storage through processing to analytics and management



The Rise of Spark and Alternatives

1

2014: Apache Spark

In-memory processing delivers 10-100x speed improvements over MapReduce—becomes the preferred engine

2

2016-Present: Cloud Era

AWS, Google Cloud, Azure offer managed Hadoop services—reducing on-premise complexity

3

2020s: Modern Paradigms

Data lakes, serverless computing, real-time analytics shift focus beyond traditional Hadoop architectures

HDFS

Hadoop Distributed File System

Core Characteristics

- Looks like a normal filesystem
- 64MB default block size
- Files split across blocks (blk_1, blk_2...)
- Example: 150MB file → 64MB + 64MB + 22MB

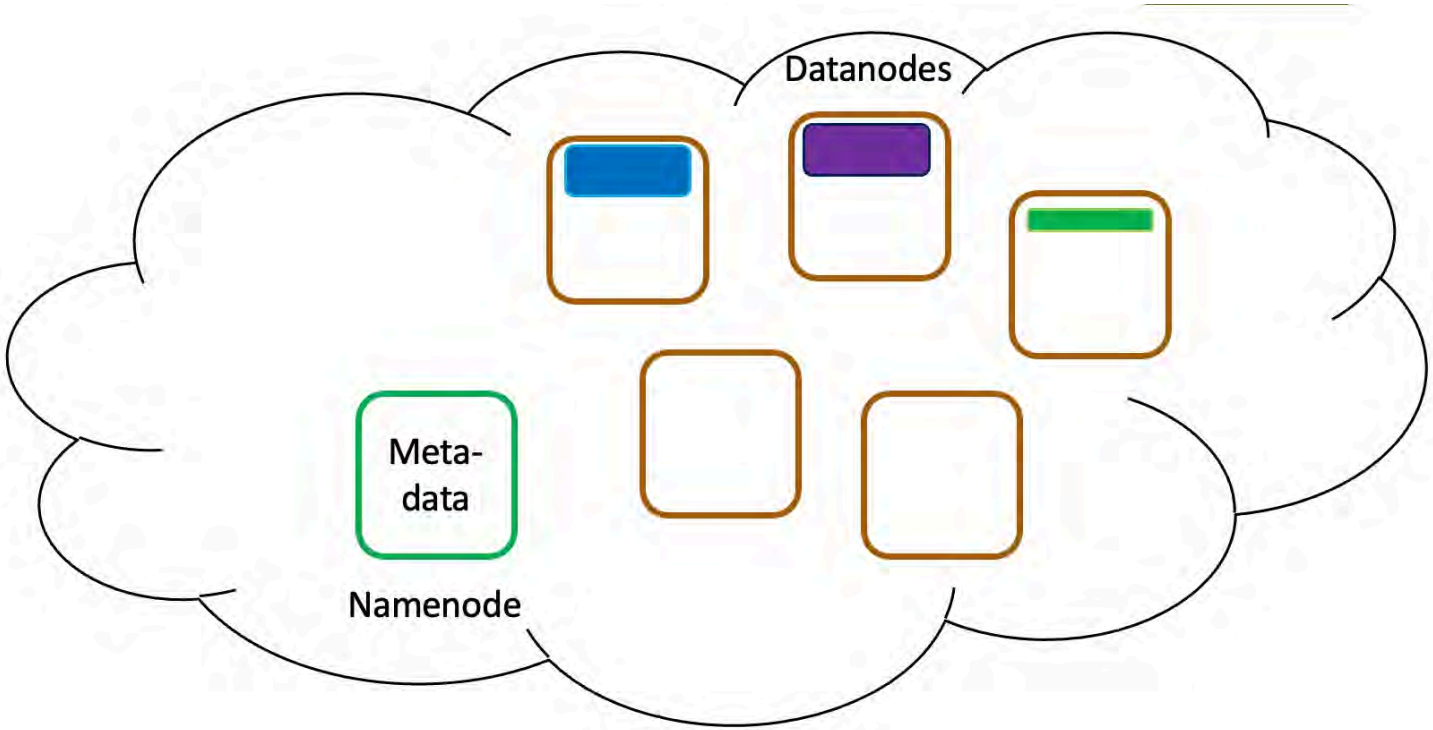
Architecture Components

NameNode: Stores metadata (where each block lives)

DataNode: Stores actual data blocks

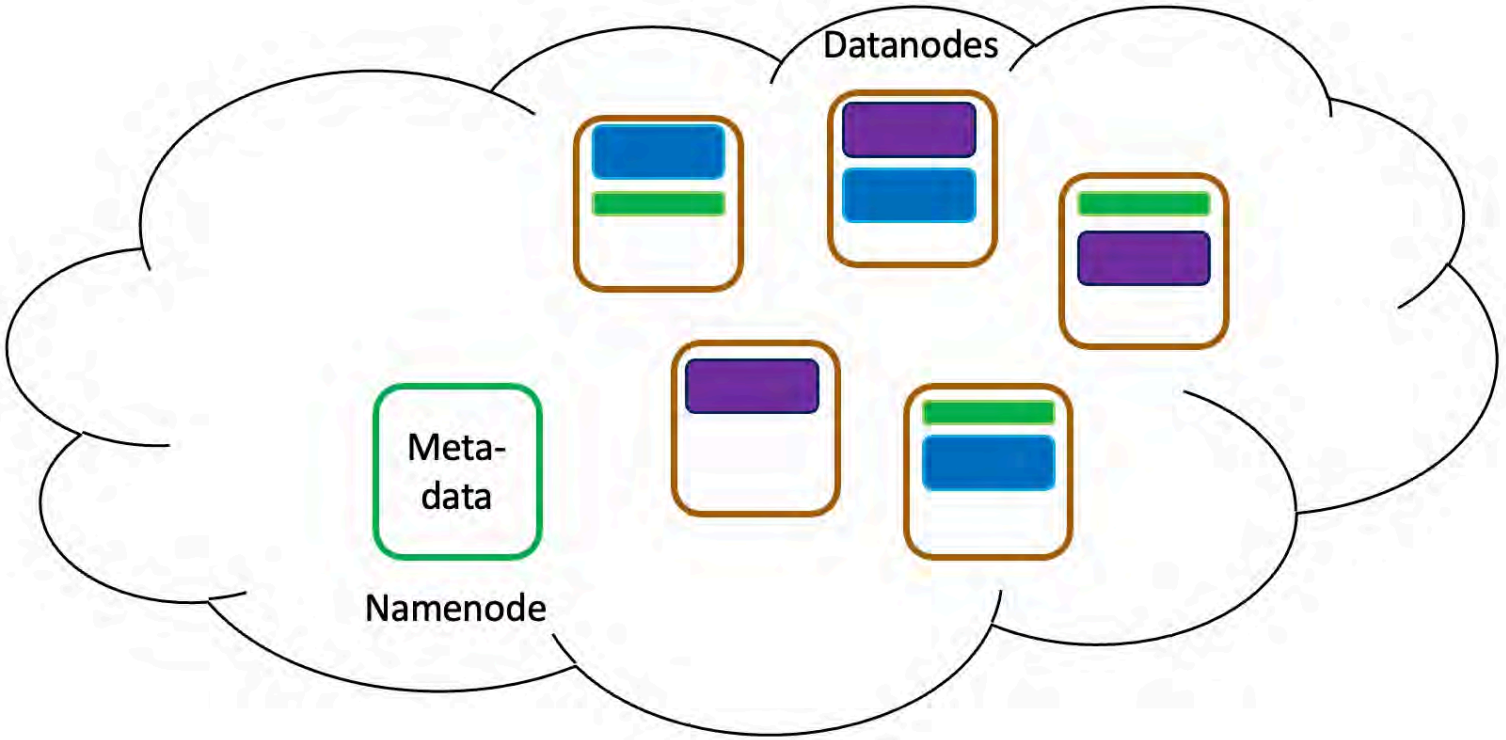
Daemons run on each cluster machine for coordination

HDFS



What if: disk failure on a data node?

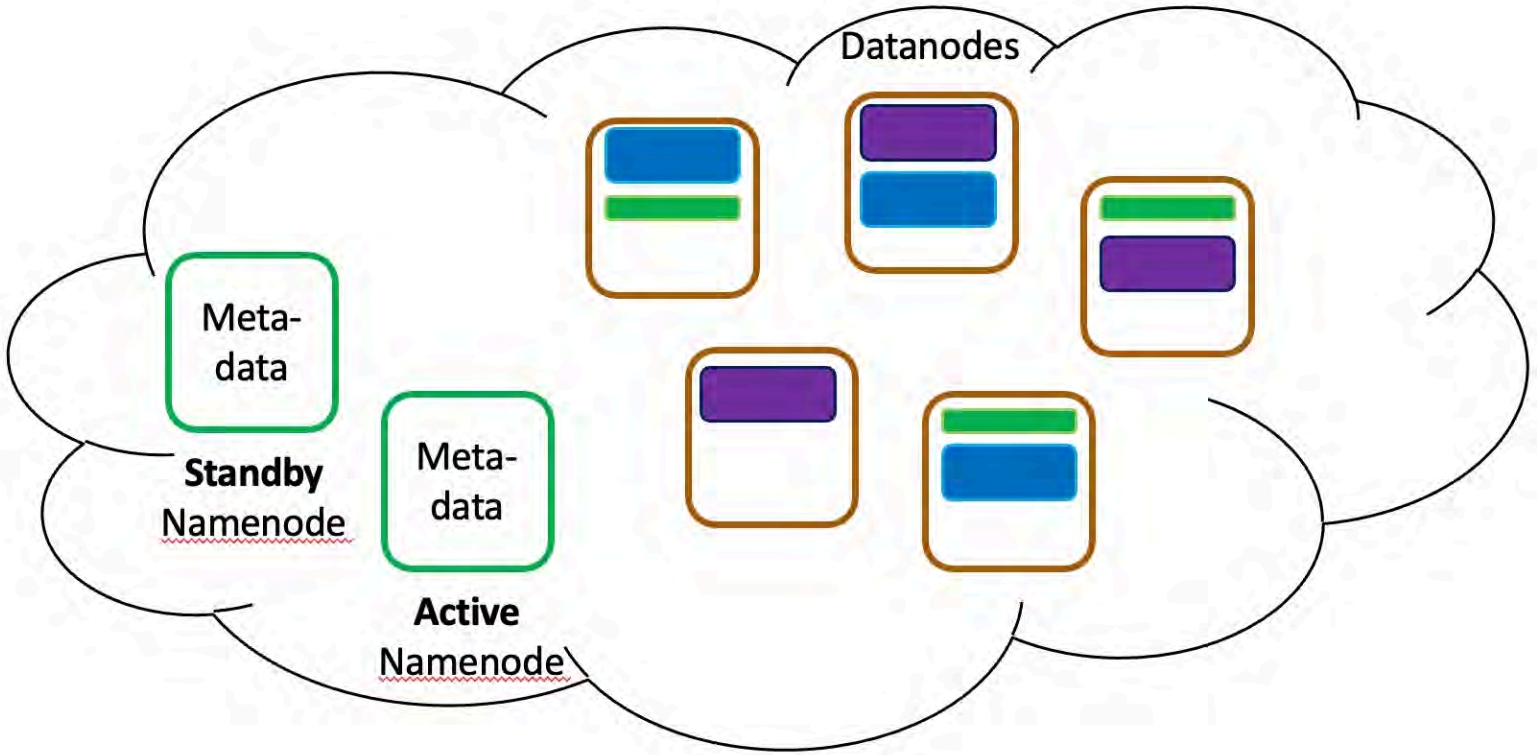
HDFS: data redundancy!



3 copies! So now data nodes can fail :)

What if: disk failure on name node?

HDFS: standby Name Node



3 copies! So now data and name nodes can fail :)

Hadoop Filesystem Commands

List Files

```
hadoop fs -ls
```

Upload File

```
hadoop fs -put myfile.txt
```

Rename File

```
hadoop fs -mv myfile.txt newname.txt
```

Delete File

```
hadoop fs -rm filename.txt
```

Create Directory

```
hadoop fs -mkdir newdir
```

Familiar syntax: Very similar to traditional Unix commands, making adoption easier for developers

MAPREDUCE

MapReduce: Why Parallel Processing?

Retailer Example

Goal: Calculate total sales per store

Traditional approach: Process sequentially from top to bottom, incrementing a hash table `<key, value>`

Problems at 1TB Scale?

- Won't work logically?
- Out of memory?
- Long time?
- Wrong answer?

MAPREDUCE

MapReduce: Why Parallel Processing?

Retailer Example

Goal: Calculate total sales per store

Traditional approach: Process sequentially from top to bottom, incrementing a hash table `<key, value>`

Problems at 1TB Scale?

- Won't work logically? ❌
- Out of memory? ✅
- Long time? ✅
- Wrong answer? ❌

MapReduce Solution

What if you had more people to help? Enter **Mappers** and **Reducers**

01

Divide

Break ledger into chunks and distribute to mapper workers

02

Map

Each mapper processes its chunk, creating intermediate `<key, value>` pairs (e.g., NYC → $4.01)

03

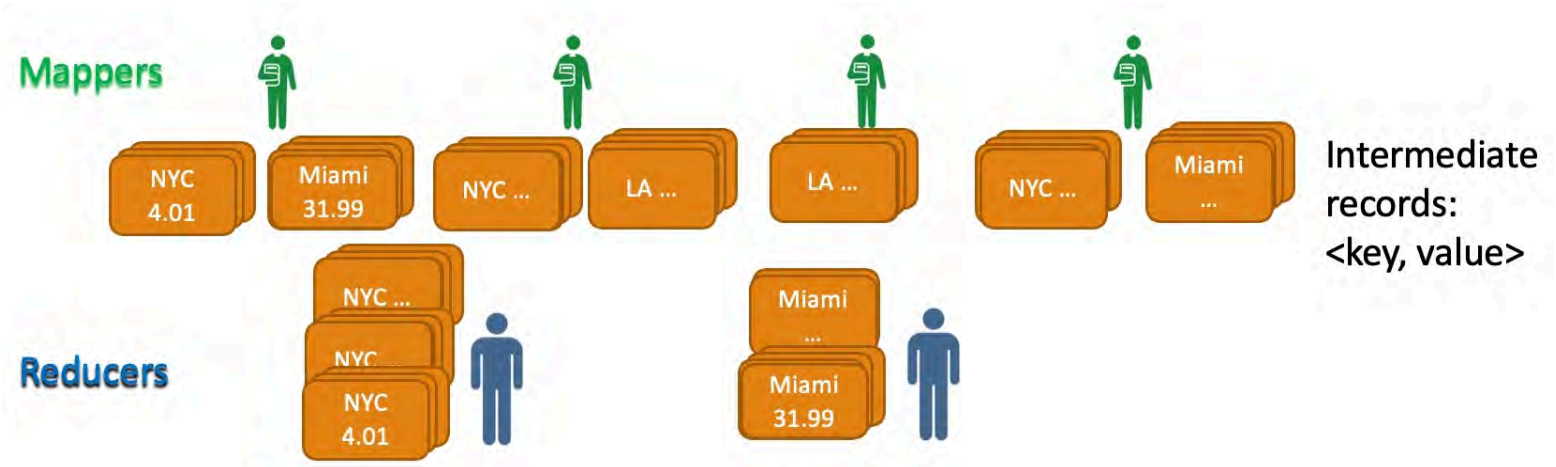
Shuffle

Reducers get assigned a store and request all records for their assigned store from all mappers

04

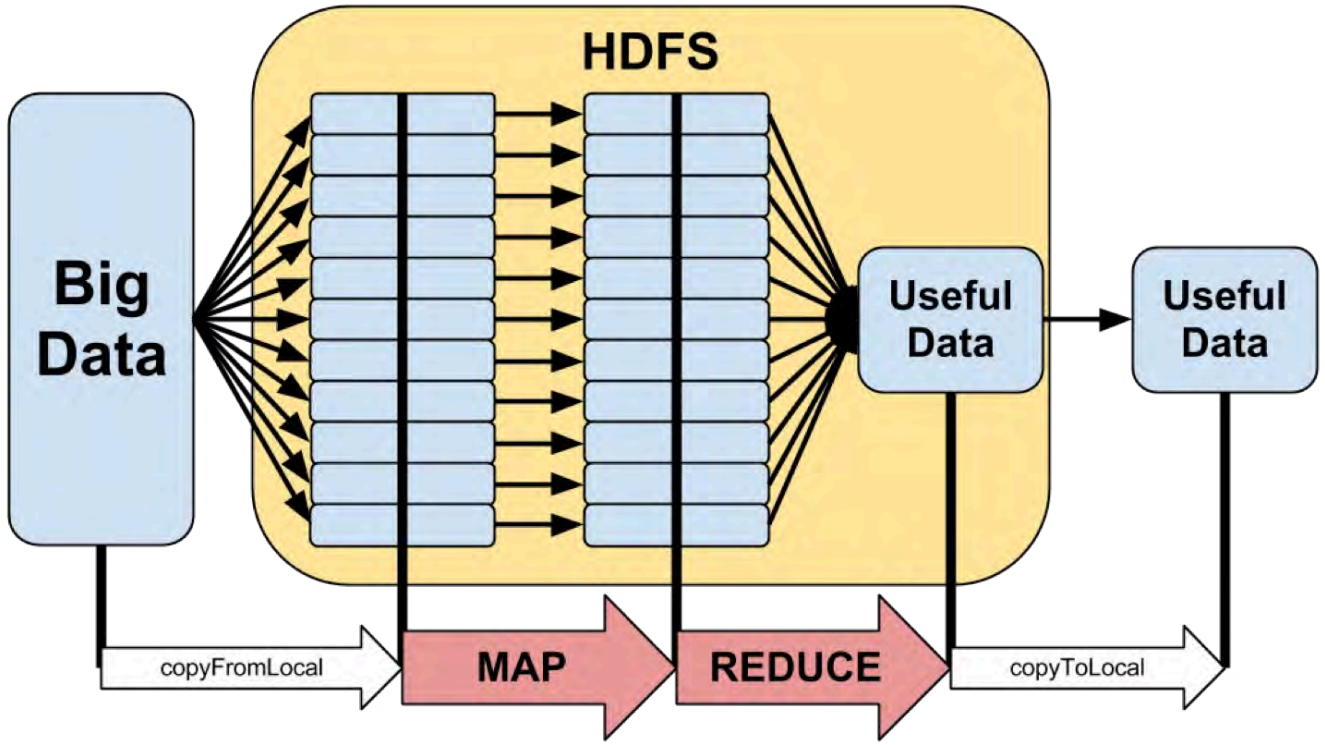
Sort & Reduce

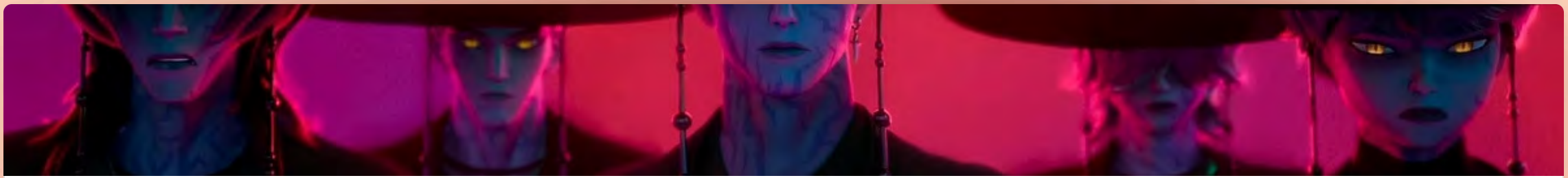
Reducers alphabetically sort their stacks and compute final totals



MapReduce Architecture

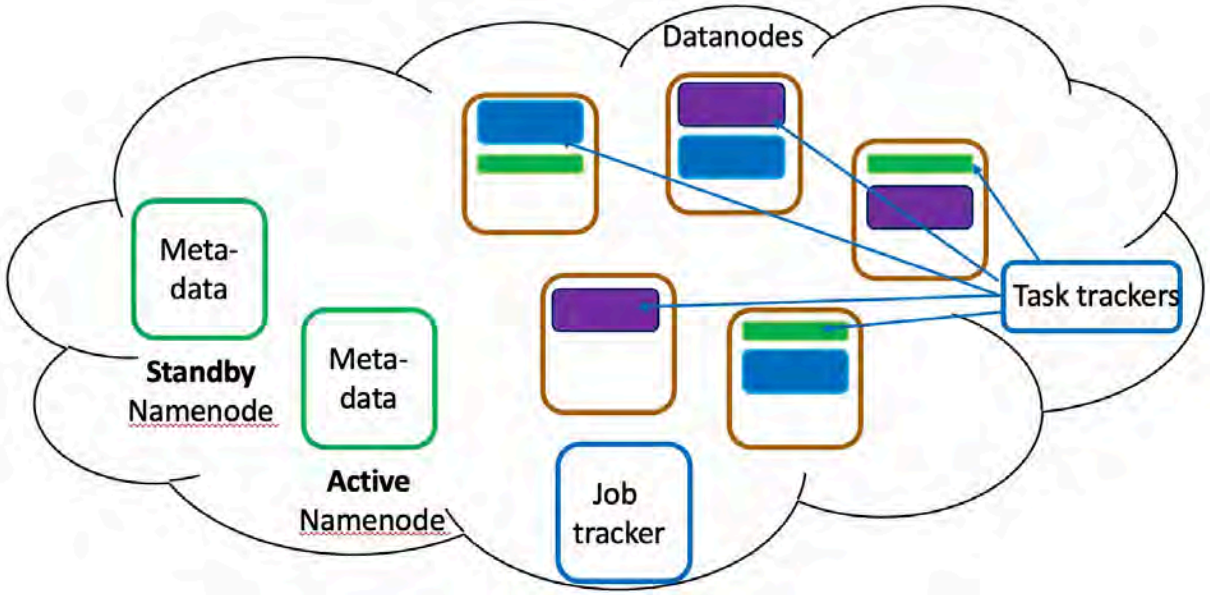
Visual flow: input splits → map phase → shuffle & sort → reduce phase → final output





MapReduce Daemons

3 copies! So now data nodes can fail :)



MapReduce Daemons

Job Tracker

Receives submitted jobs and splits work across mappers and reducers

Task Tracker

Runs on each DataNode, executes actual mapping and reducing tasks locally

Mappers

Filter and sort data, pass intermediate results to reducers

Reducers

Aggregate intermediate data (various operations possible), write final output to HDFS

Data locality advantage: Task Trackers run on the same machines as DataNodes, minimizing network traffic!
Hadoop will try to make sure a task tracker works on data from the same machine (if not already busy)

Batch vs. Real-Time Processing

Batch Processing

MapReduce paradigm

- Processes large batches at scheduled intervals
- High latency (minutes to hours)
- Efficient for historical analysis
- Use cases: data warehousing, reporting, ETL

Real-Time (Streaming)

Spark, Flink, Kafka

- Continuous processing as data arrives
- Low latency (milliseconds to seconds)
- In-memory processing for speed
- Use cases: fraud detection, social trends, IoT sensors, trading

Hadoop Today: Still Relevant?

Hadoop remains a strong choice for large-scale batch jobs and data lake storage

- Hadoop ecosystem tools (Hive, HBase, YARN) persist in enterprises with established pipelines

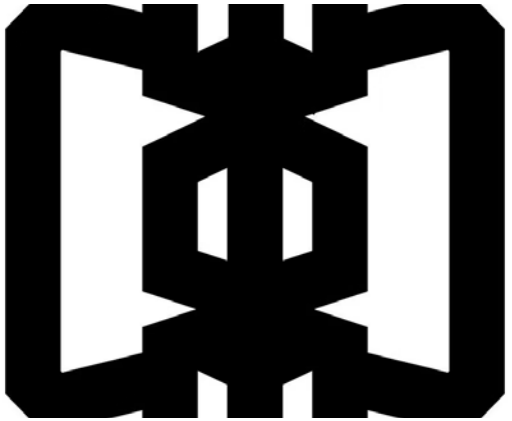
The landscape has shifted: While Hadoop pioneered distributed big data processing, newer technologies have overtaken its original dominance. Modern trends favor scalable, serverless, and real-time data solutions over traditional Hadoop clusters.

- **Apache Spark:** In-memory processing for dramatically faster analytics
- **Cloud platforms:** Managed services reduce operational complexity
- **NoSQL databases:** Purpose-built for specific query patterns

MODERN ALTERNATIVES

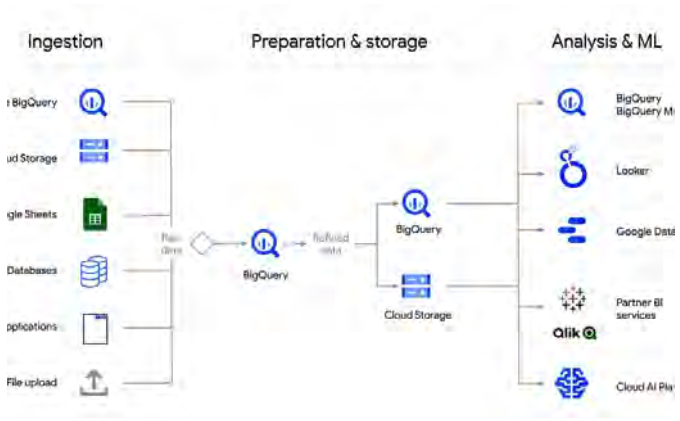
Cloud-Based Data Platforms

Cloud providers now offer **Hadoop-like functionality without cluster management complexity** (serverless architectures)



Amazon EMR

Elastic MapReduce—managed Hadoop/Spark service on AWS infrastructure



Google BigQuery

SQL-based serverless data warehouse for petabyte-scale analytics

Databricks

Unified analytics platform by Spark creators—collaborative data science, ML, big data (used by Shell, HSBC)

NoSQL Databases

For Unstructured & Semi-Structured Data

Hadoop's HDFS excelled at storage but lacked fast query capabilities. NoSQL databases fill that gap.

Apache Cassandra

Scalable, high-availability distributed NoSQL database. Used by Facebook and Instagram for massive-scale data.

MongoDB

Document-oriented NoSQL database. Popular for web and mobile applications with flexible schemas.

Google Cloud Bigtable

Google's distributed NoSQL database powering real-time analytics at planetary scale.

The Evolution Continues

1

2003-2008

Google papers inspire Hadoop—distributed storage and MapReduce processing

2

2008-2014

Hadoop ecosystem expands—Hive, Pig, HBase, YARN enable diverse workloads

3

2014-2020

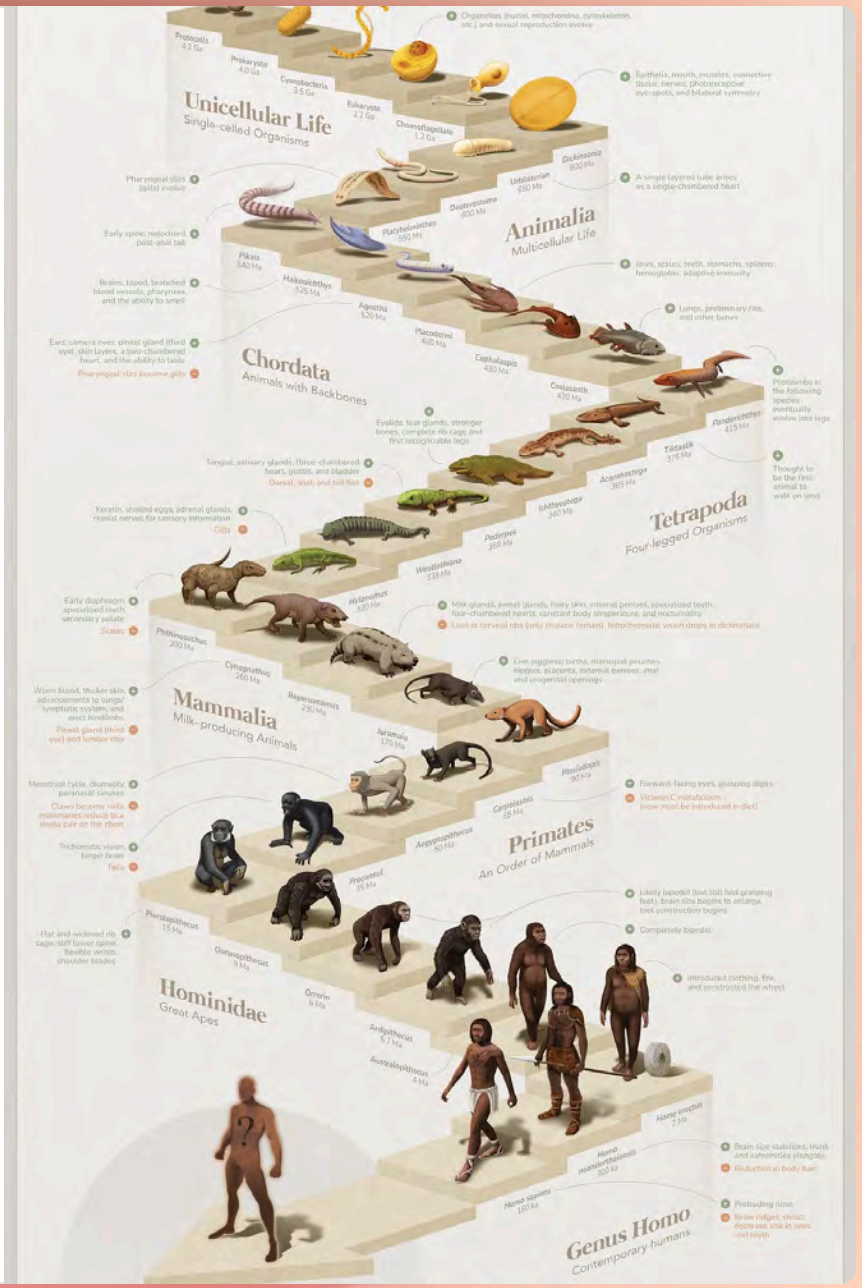
Spark and cloud platforms rise—in-memory processing and managed services

4

2020+

Modern era—serverless, real-time streaming, specialized NoSQL, data lakes

From Hadoop's pioneering vision to today's diverse ecosystem, big data continues to transform how we process and understand information.



Did You Pay Attention?

Take the Pop Quiz

Test your knowledge of big data concepts, Hadoop architecture, and modern alternatives

https://forms.gle/FpGKgKRjwBB4wrdK7