DTS207TC Database Development and Design

Lecture 6
UML for CW1Q4

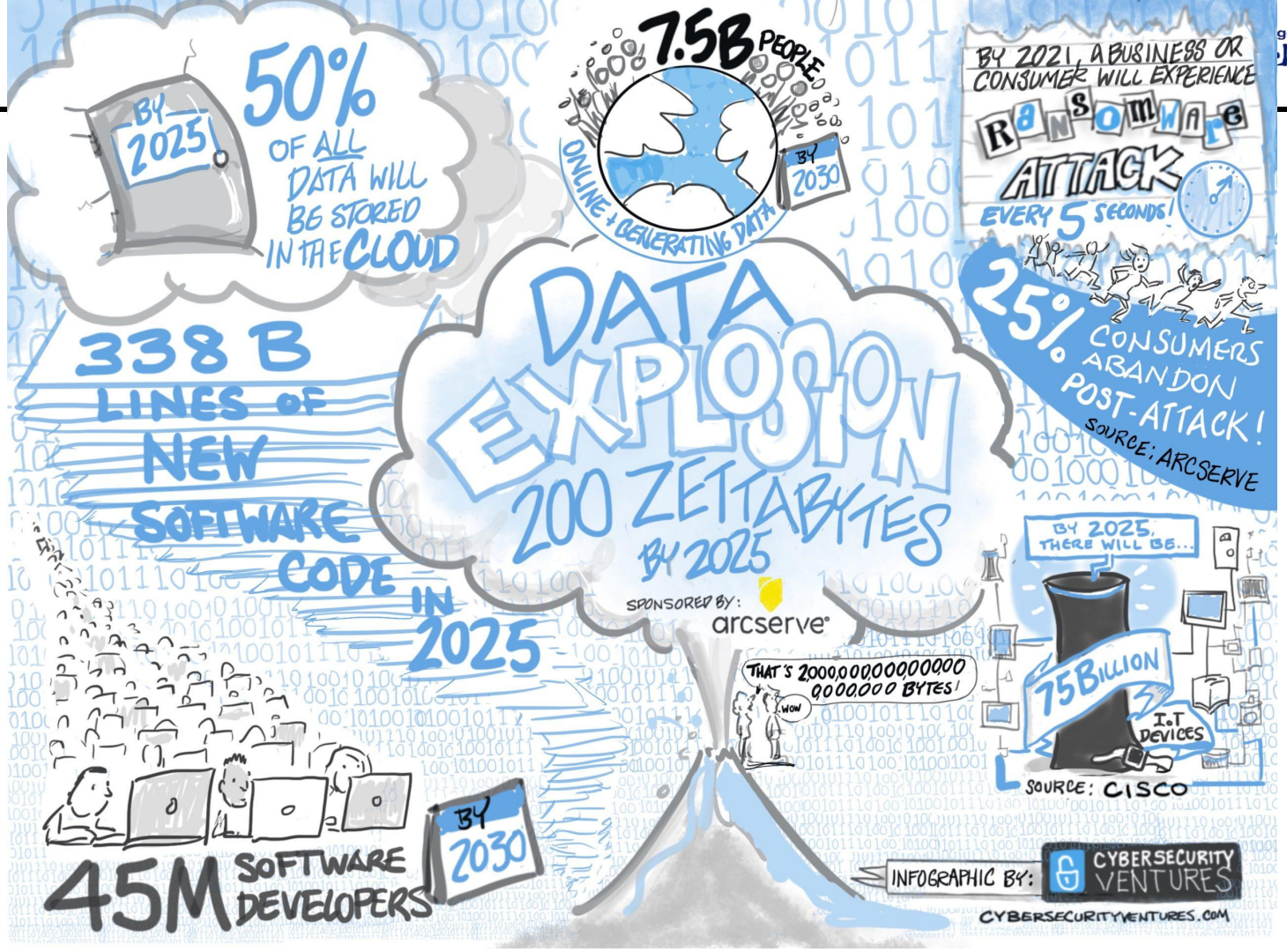
Chap 10 Big Data*

Di Zhang, Autumn 2025

*Page titles with * will not be assessed*

Outline

- Big Data
- Hadoop


BY 2025
50% OF ALL DATA WILL BE STORED IN THE CLOUD
7.5B PEOPLE
ONLINE + GENERATING DATA
BY 2030
BY 2021, A BUSINESS OR CONSUMER WILL EXPERIENCE
RANSOMWARE ATTACK
EVERY 5 SECONDS!
25% CONSUMERS ABANDON POST-ATTACK!
SOURCE: ARCSERVE
338 B LINES OF NEW SOFTWARE CODE IN 2025
DATA EXPLOSION
200 ZETTABYTES BY 2025
SPONSORED BY: arcserve
THAT'S 200,000,000,000,000,000,000 BYTES!
WOW
BY 2025, THERE WILL BE...
75 BILLION
IoT DEVICES
SOURCE: CISCO
45M SOFTWARE DEVELOPERS
BY 2030
INFOGRAPHIC BY: CYBERSECURITY VENTURES
CYBERSECURITYVENTURES.COM

www.wipro.com

BIG DATA

Big Data is data that is too large, complex and dynamic for any conventional data tools to capture, store, manage and analyze.

The right use of Big Data allows analysts to spot trends and gives niche insights that help create value and innovation much faster than conventional methods.

The "three V's", i.e the Volume, Variety and Velocity of the data coming in is what creates the challenge.

VOLUME

Amount of Big Data stored across the world (in petabytes)

VARIETY

PEOPLE TO PEOPLE
NETIZENS, VIRTUAL COMMUNITIES, SOCIAL NETWORKS, WEB LOGS...

PEOPLE TO MACHINE
ARCHIVES, MEDICAL DEVICES, DIGITAL TV, E-COMMERCE, SMART CARDS, BANK CARDS, COMPUTERS, MOBILES...

MACHINE TO MACHINE
SENSORS, GPS DEVICES, BAR CODE SCANNERS, SURVEILLANCE CAMERAS, SCIENTIFIC RESEARCH...

VELOCITY

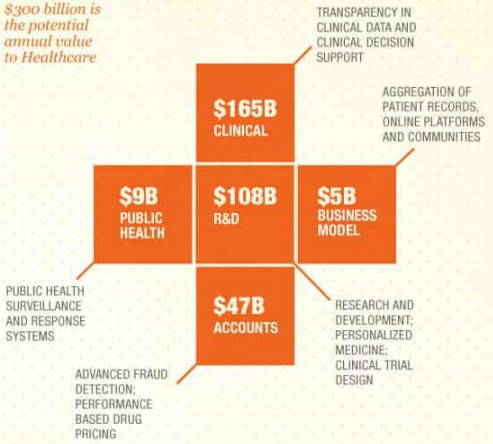
2.9 MILLION EMAILS SENT EVERY SECOND

20 HOURS OF VIDEO UPLOADED EVERY MIN

50 MILLION TWEETS PER DAY

CASE STUDY - Healthcare

$300 billion is the potential annual value to Healthcare

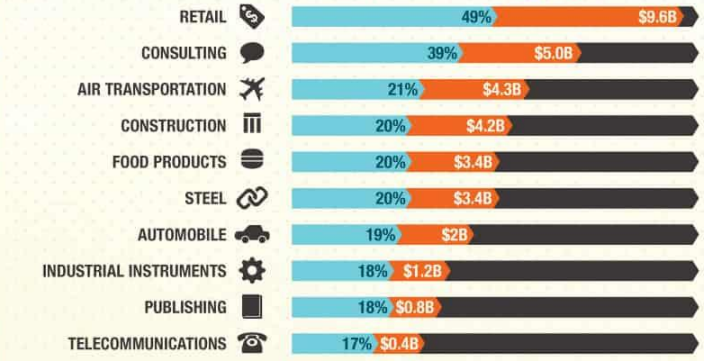


57.6% OF ORGANIZATIONS SURVEYED SAY THAT BIG DATA IS A CHALLENGE

72.7% CONSIDER DRIVING OPERATIONAL EFFICIENCIES TO BE THE BIGGEST BENEFIT OF A BIG DATA STRATEGY

50% SAY THAT BIG DATA HELPS IN BETTER MEETING CONSUMER DEMAND AND FACILITATING GROWTH

VALUE



40% PROJECTED GROWTH IN GLOBAL DATA CREATED PER YEAR

5% PROJECTED GROWTH IN GLOBAL IT SPENDING PER YEAR

The estimated size of the digital universe in 2011 was 1.8 zettabytes. It is predicted that between 2009 and 2020, this will grow 44 fold to 35 zettabytes per year. A well defined data management strategy is essential to successfully utilize Big Data.

Sources - 1 Reaping the Rewards of Big Data - Wipro Report 2 Big Data: The Next Frontier for Innovation, Competition and Productivity - McKinsey Global Institute Report 3 comScore, Radicati Group 4 Measuring the Business Impacts of Effective Data - study by University of Texas, Austin 5 US Department of Labour

DO BUSINESS BETTER

NYSE:WIT | OVER 130,000 EMPLOYEES | 54 COUNTRIES | CONSULTING | SYSTEM INTEGRATION | OUTSOURCING

WIPRO Applying Thought

Visualization: Sevenfifteen

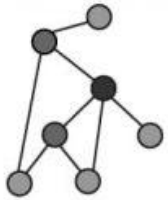
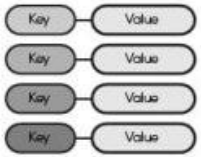
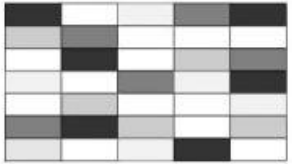
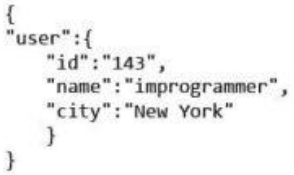
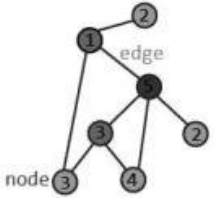
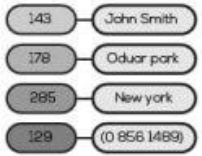
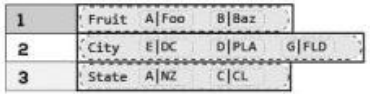
6 Steps in CRISP-DM

The Standard Data Mining Process

Challenge and Response

- Volume
  - Distributed Computing
- Velocity
  - ACID->BASE: NoSQL
- Variety
  - Unstructed Text

NoSQL DATABASE TYPES

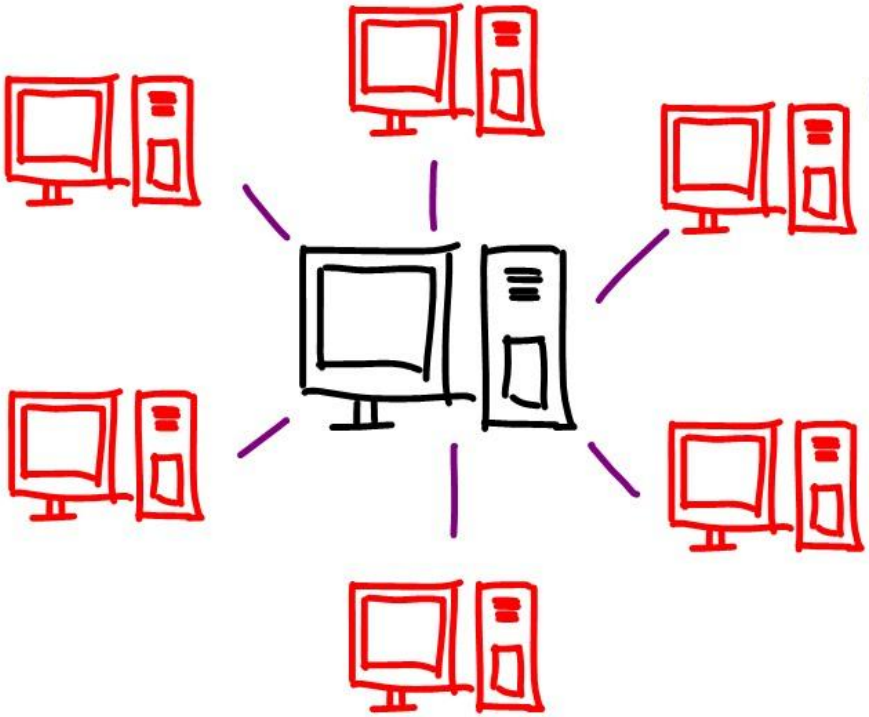
Document	Graph	Key-Value	Wide-Column
			
			
mongoDB, ArangoDB, CouchDB	AllegroGraph, neo4j	redis, Ignite, MEMCACHED, riak	cassandra, HBASE, SCYLLA

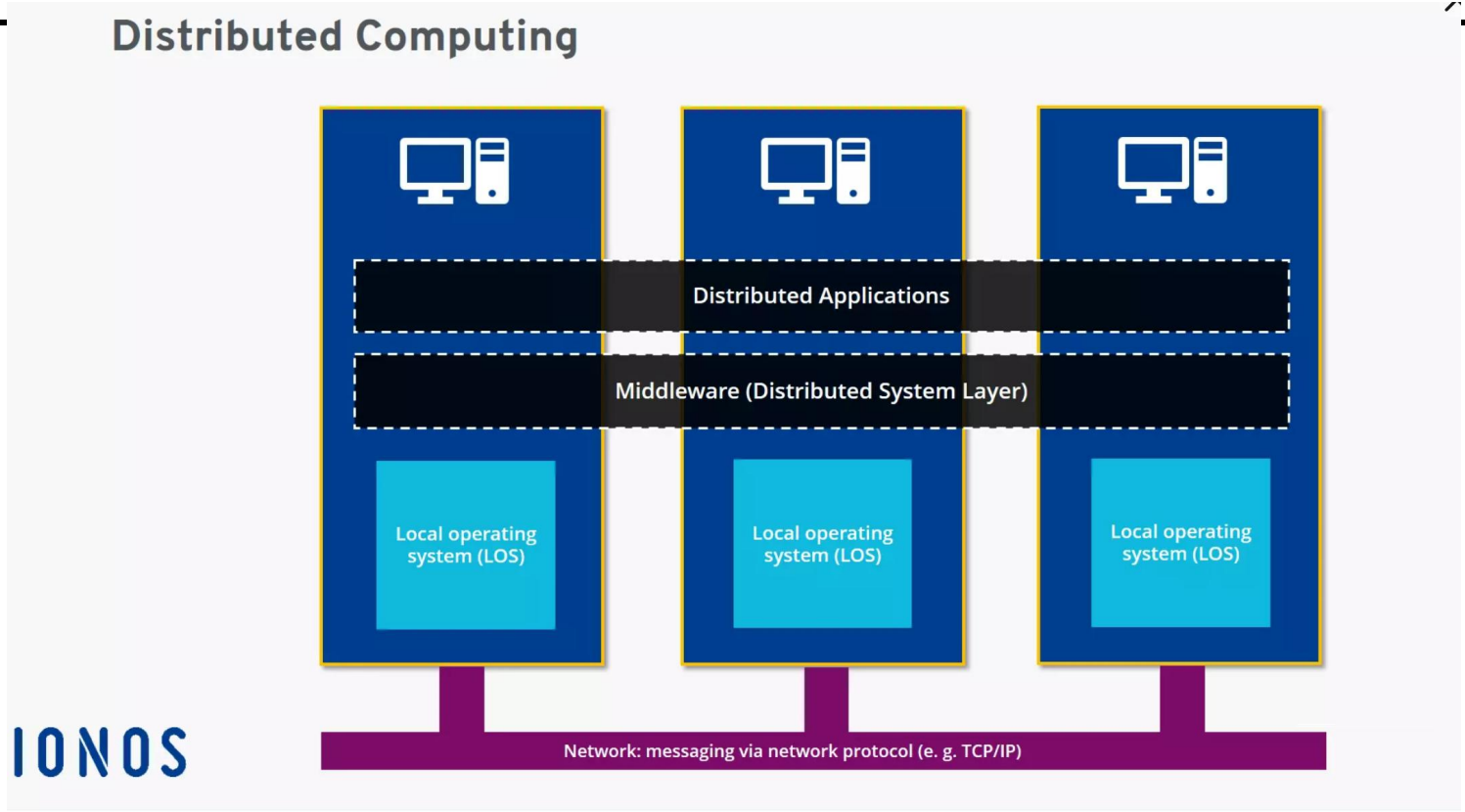
```
{
"user":{
    "id":"143",
    "name":"improgrammer",
    "city":"New York"
    }
}
```


1	Fruit	A|Foo	B|Baz		
2	City	E|DC	D|PLA	G|FLD	
3	State	A|NZ	C|CL		

-

Distributed Computing





Distributed applications can solve problems across devices in a computer network. When used in conjunction with middleware, they can optimize operational interactions with locally accessible hardware and software.

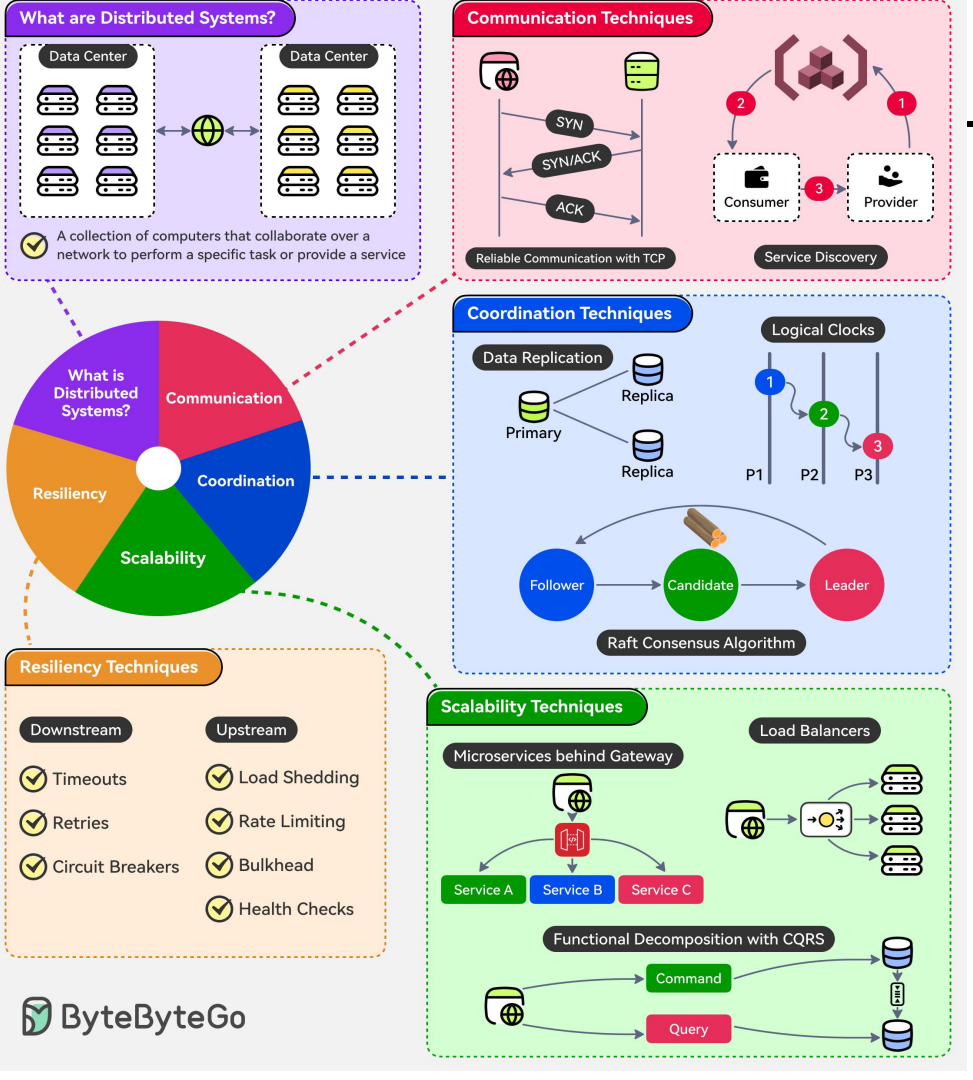
Distributed Computing

Advantages

- 🔒 Improved Security
- 📏 Flexibility and Scalability
- 🚀 Enhanced Performance
- ⚡ Low Latency
- 🔗 Increased Reliability
- 💾 Expanded Storage
- $ Cost effectiveness

Disdvantages

- 💰 Maintenance Fees
- 💔 Component Failure
- 🌐 Bandwidth Restrictions
- 🧩 Systems Complexity
- 🚧 Development Obstacles
- 🐢 Slow Network Transfers
- ❓ Issues with Standardization


What are Distributed Systems?
Data Center
Data Center
A collection of computers that collaborate over a network to perform a specific task or provide a service
Communication Techniques
SYN
SYN/ACK
ACK
Reliable Communication with TCP
1
2
3
Consumer
Provider
Service Discovery
What is Distributed Systems?
Communication
Coordination
Scalability
Resiliency
Coordination Techniques
Data Replication
Primary
Replica
Replica
Logical Clocks
P1
P2
P3
Follower
Candidate
Leader
Raft Consensus Algorithm
Resiliency Techniques
Downstream
Timeouts
Retries
Circuit Breakers
Upstream
Load Shedding
Rate Limiting
Bulkhead
Health Checks
Scalability Techniques
Microservices behind Gateway
Service A
Service B
Service C
Load Balancers
Functional Decomposition with CQRS
Command
Query
ByteByteGo

Distributed System

- A distributed system is a collection of independent computers that appear to users as a single coherent system
- Focuses on providing reliability, availability, & fault tolerance by distributing data & processes across multiple nodes
- To ensure system resilience, fault tolerance, & scalability by distributing services or data
- Involves systems such as distributed databases, message queues, or microservices

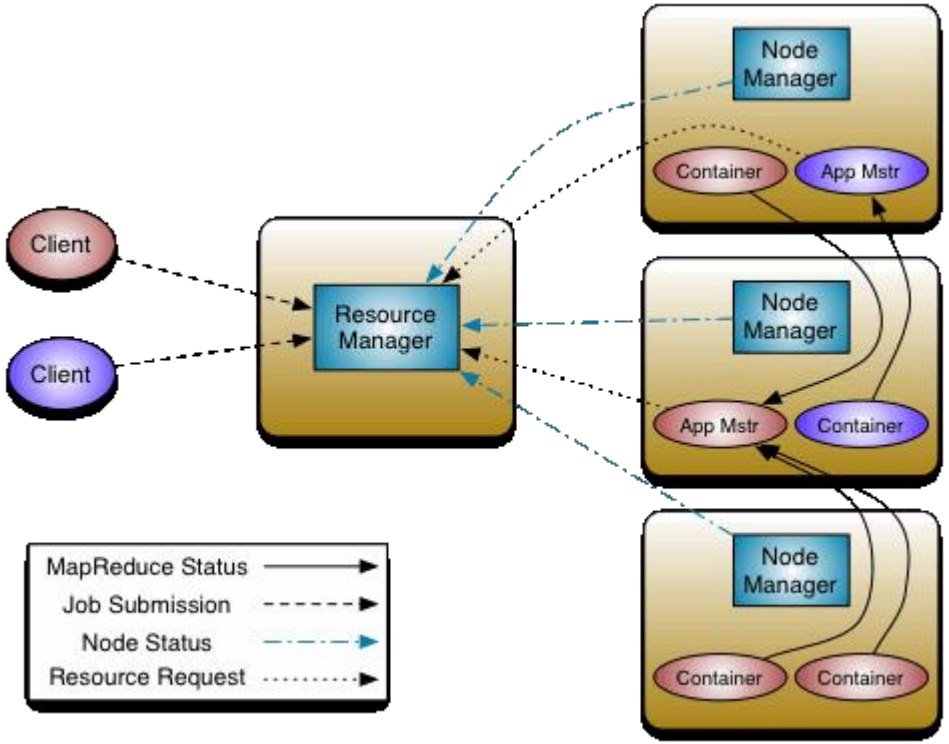
VS

Distributed Computing

- Distributed computing refers to using multiple computers to perform computational tasks collaboratively.
- Focuses on splitting large computational tasks into smaller chunks that can be executed concurrently.
- To solve complex computational problems by breaking them down and distributing tasks across multiple nodes.
- Involves dividing a large computational workload into smaller tasks

Aspect	Parallel Computing	Distributed Computing
Primary Goal	Increase speed, reduce execution time for a single, complex task.	Resource sharing, scalability, fault tolerance, solving large-scale problems.
System Architecture	Tightly-coupled. Processors connected via high-speed buses, shared memory, or specialized networks (InfiniBand).	Loosely-coupled. Computers (nodes) connected via standard networks (Ethernet, Internet).
Memory Model	Often Shared Memory. All processors access a common memory address space.	Often Distributed Memory. Each node has its own private memory; data is exchanged via message passing (e.g., MPI).
Communication & Coordination	Very low latency, very high bandwidth. Processes/threads require tight synchronization.	Higher latency, lower bandwidth. Nodes collaborate through asynchronous message passing.
User Perspective	Feels like using a single, powerful supercomputer.	Feels like using a single, coherent system (even though it's composed of many machines).
Scaling Method	Vertical Scaling (Scale-up). Adding more CPUs/cores within a single node. Limited by physical constraints.	Horizontal Scaling (Scale-out). Adding more commodity computer nodes. Theoretically unlimited.
Fault Tolerance	Typically weak. The failure of a single processor can cause the entire computation to fail.	Inherently strong. Designed with node failure in mind. Tasks can be restarted on other nodes if one fails.
Typical Applications	Scientific computing, 3D rendering, weather simulation, genomic sequence analysis.	Large web services (Google, Amazon), Big Data processing (Hadoop/Spark), blockchain, cloud platforms.
Programming Models/Tools	OpenMP (shared memory), MPI (distributed memory, for tight clusters), CUDA (GPU).	Hadoop, Spark, Akka, gRPC, and various cloud APIs (AWS, Azure).

Example: Hadoop



Why Hadoop belongs to distributed Computing, but MPI not?

- 1. Primary Goal: Scale vs. Speed
  - Extreme Speed vs. the Ability to Process Massive Data

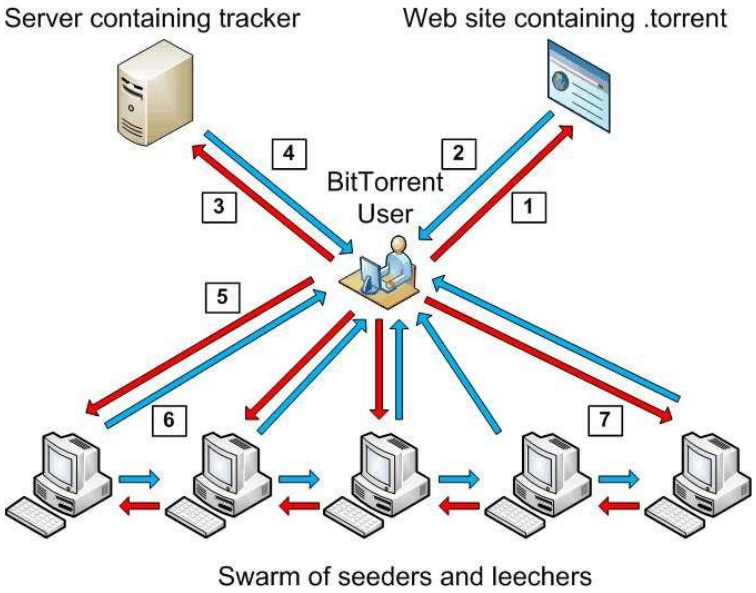
- 2. System Architecture: Fault Tolerance vs. Performance (the most critical difference)
  - Parallel Computing (MPI): Tightly coupled. After tasks are distributed to multiple nodes, frequent communication and synchronization between nodes are required.
  - Hadoop: Loosely coupled. Its core design philosophies are "mobile computing is cheaper than moving data" and "hardware failures are the norm, not the exception."

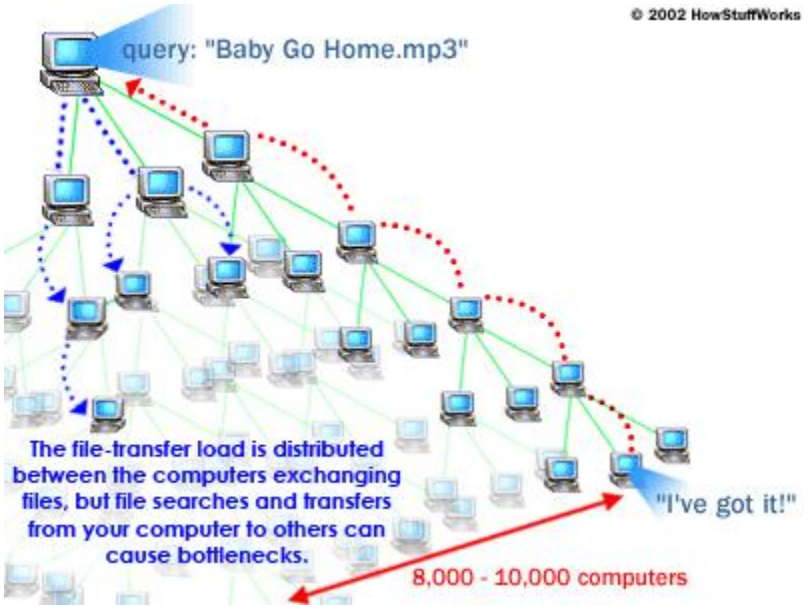
- 3. Communication Model: Message Passing vs. Data Sharing
  - Parallel Computing (MPI): Frequent, low-latency, bidirectional communication between processes is achieved through a message passing interface.
  - Hadoop MapReduce: There is almost no communication between processes. This "share nothing" architecture is a typical distributed system design.

- 4. Problem Domain: Single Complex Task vs. Decomposable Batch Tasks
  - Parallel Computing: Excels at solving a single, complex problem that requires a high degree of coordination (such as simulating an entire weather system). Hadoop MapReduce excels at batch processing problems that can be clearly decomposed into a large number of independent subtasks (such as counting the number of visits to each URL in a massive log). Each subtask processes a small portion of the data and does not depend on the intermediate results of other subtasks.

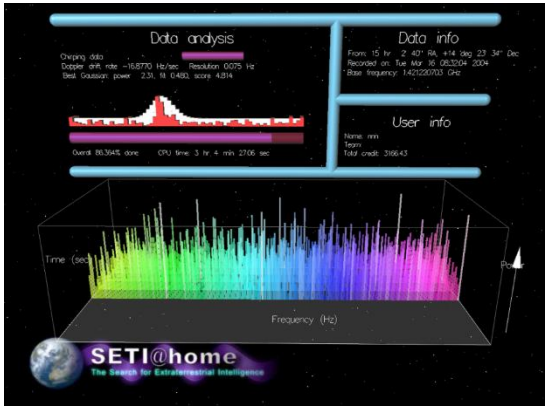
- An analogy: MPI is like a symphony orchestra. Hadoop is like a courier company's sorting center.

Example: P2P Download



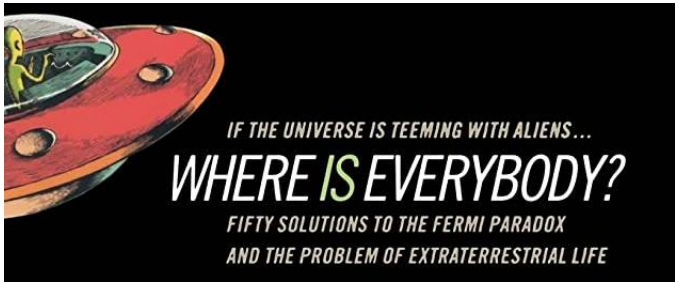


Example: the ‘Slowest’ Distributed System



Search Alien Civilization, using volunteer PCs on the internet

- Found nothing…
  - Fermi Paradox



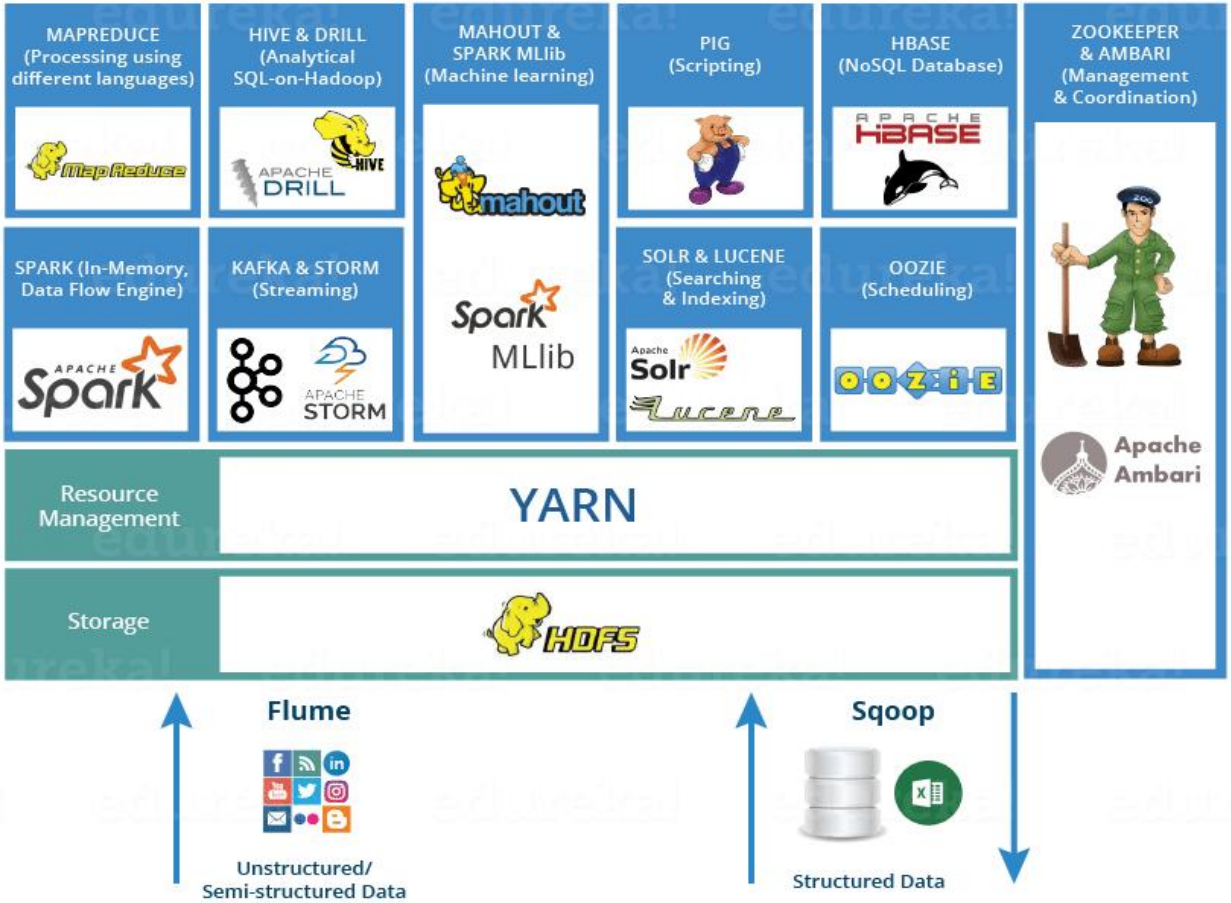
What is Hadoop?

- An open-source, reliable, and scalable distributed computing framework developed by the Apache Foundation.

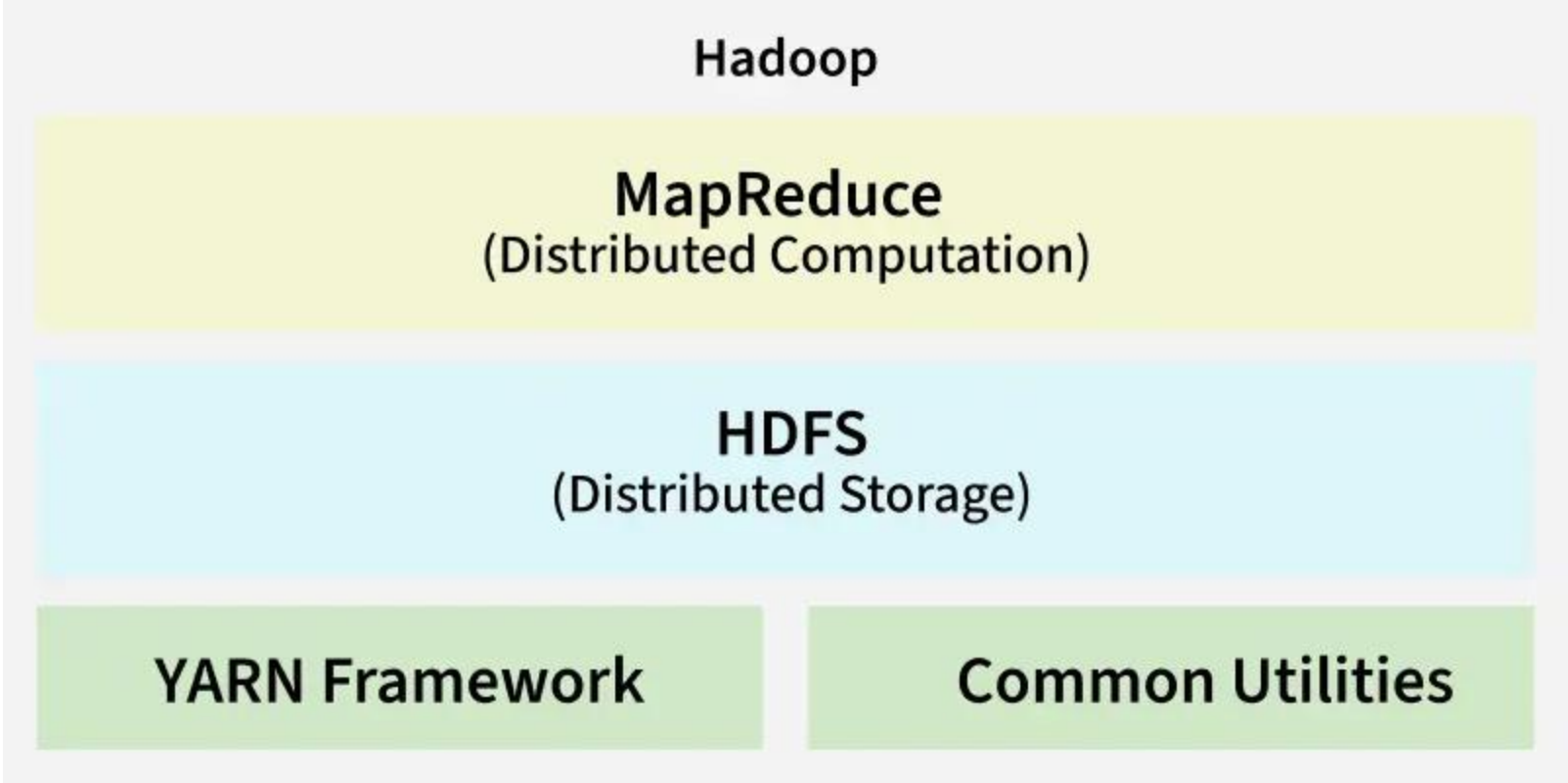
- Core ideas: Derived from Google's GFS and MapReduce papers.

- Key advantages: High fault tolerance, high throughput, and low cost (using commodity hardware).

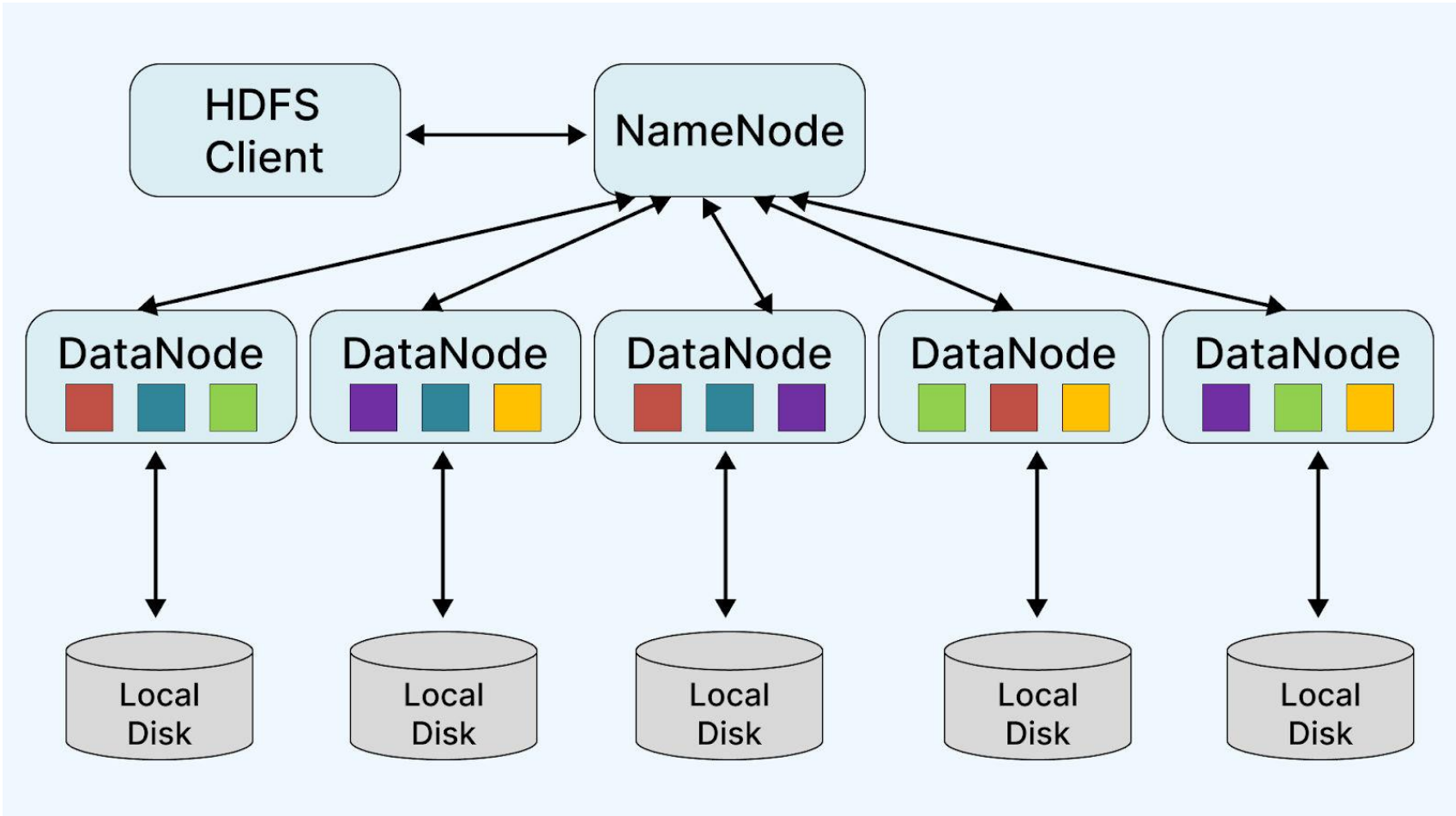
Hadoop Ecosystem



Hadoop Architecture

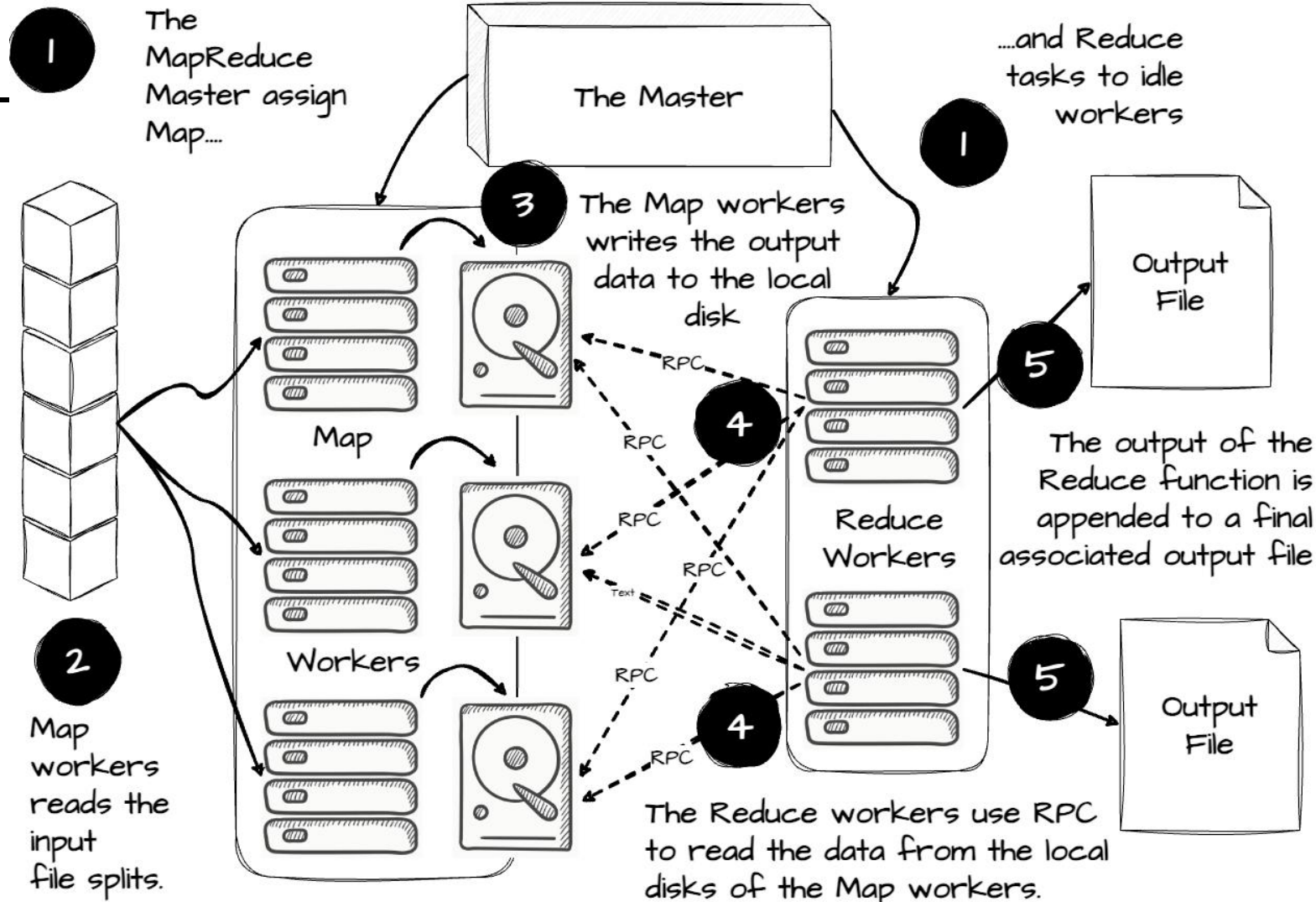


Hadoop Cluster



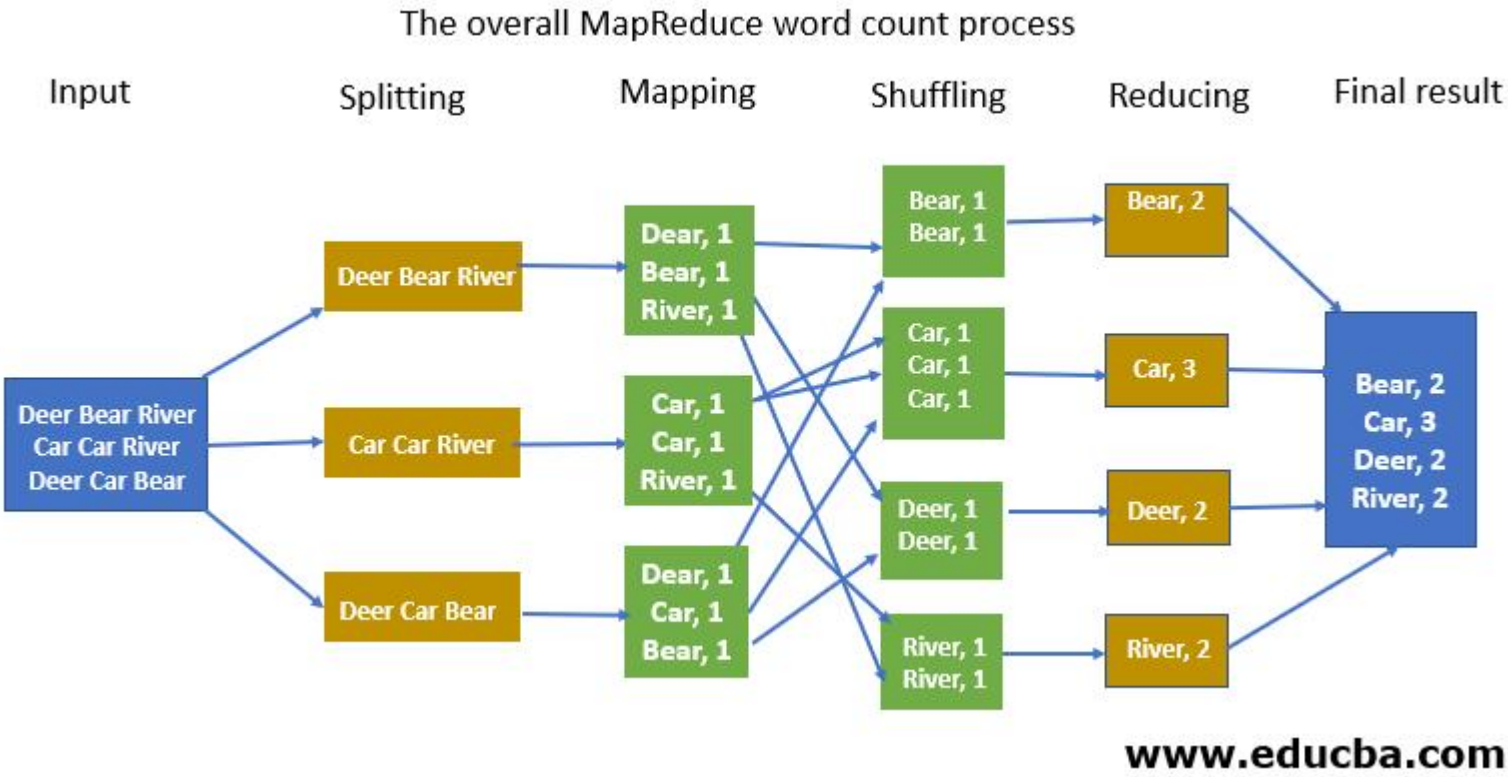
What is MapReduce?

- A programming model and computing framework for large-scale data processing.
  - The core concept is "divide and conquer."
  - Break down complex tasks into multiple smaller tasks, process them in parallel across a cluster, and then aggregate the results.


1
The MapReduce Master assign Map....
The Master
....and Reduce tasks to idle workers
1
3
The Map workers writes the output data to the local disk
Output File
5
RPC
4
Map
RPC
The output of the Reduce function is appended to a final associated output file
RPC
Reduce Workers
RPC
Text
Workers
RPC
5
Output File
4
RPC
2
Map workers reads the input file splits.
The Reduce workers use RPC to read the data from the local disks of the Map workers.

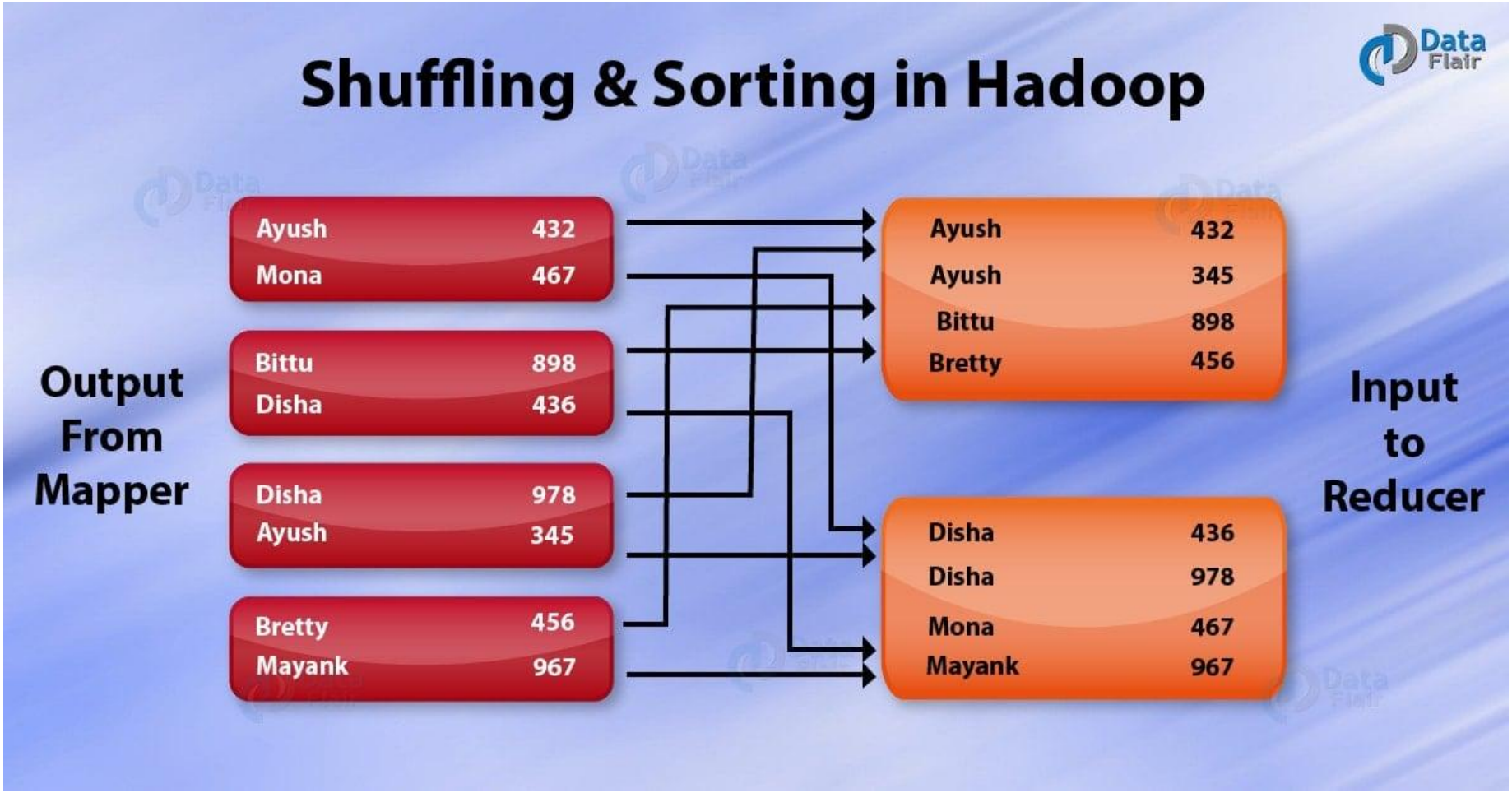
A vivid example: counting word frequencies in a library

- Question: How can we quickly count the number of occurrences of each word in all the books in the library?
- Traditional method: One person counts each book one by one. Too slow!
- MapReduce Method:
  - Step 1: Map: Have multiple people (workers) each be assigned a few books, and each count the number of occurrences of a word in their own book, outputting an intermediate result such as (word, 1).
  - Step 2: Shuffle & Sort: Group and sort all intermediate results by word, so that all records for the same word are grouped together.
  - Step 3: Reduce: Have multiple people (workers) each be responsible for a subset of words, and sum up all the counts for each word to produce the final result (word, total counts).


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Dear, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Dear, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2
www.educba.com

Code

- https://www.geeksforgeeks.org/python/hadoop-streaming-using-python-word-count-problem/

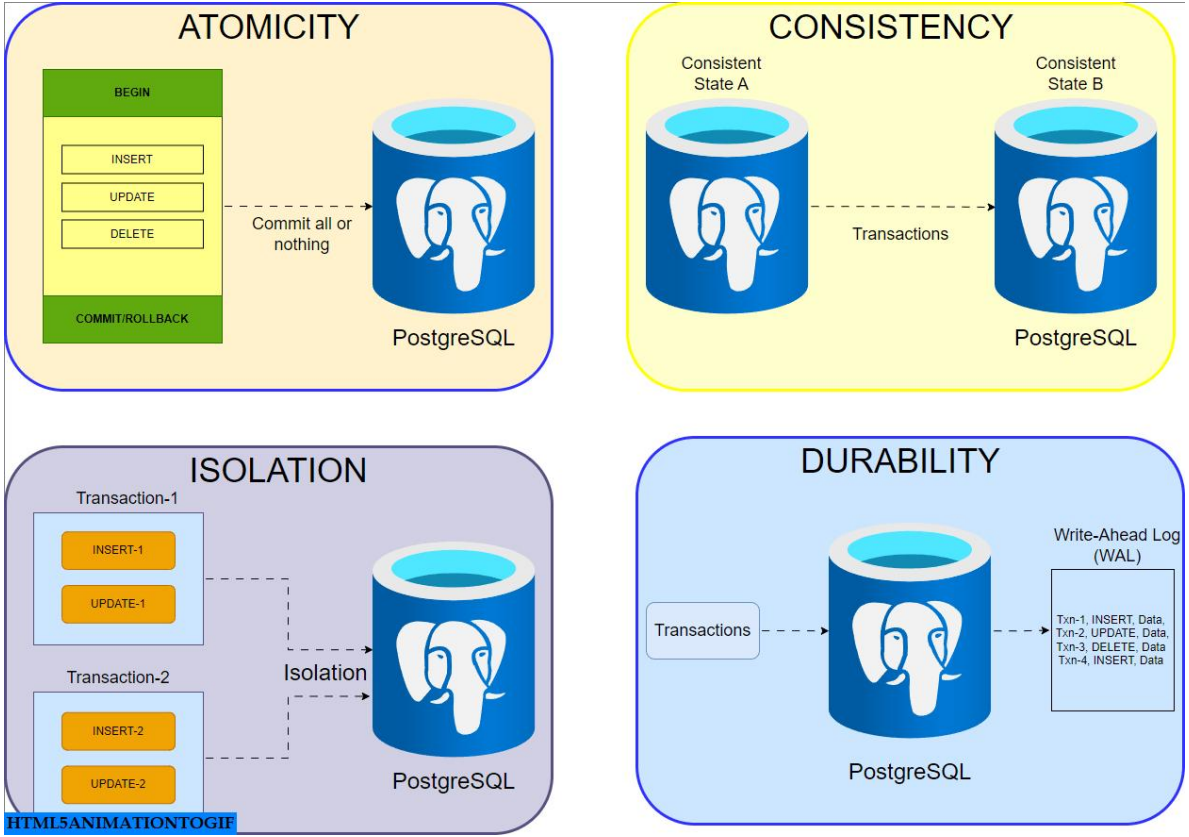


- MapReduce = Map + Shuffle + Reduce

Querying Big Data

- Transaction processing systems that need very high scalability
  - Many applications willing to sacrifice ACID properties and other database features, if they can get very high scalability
- Query processing systems that
  - Need very high scalability, and
  - Need to support non-relation data

ACID

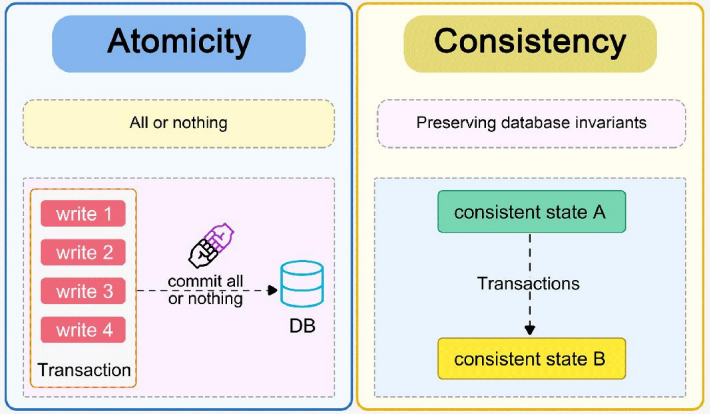

ATOMICITY
BEGIN
INSERT
UPDATE
DELETE
COMMIT/ROLLBACK
Commit all or nothing
PostgreSQL
CONSISTENCY
Consistent State A
Consistent State B
Transactions
PostgreSQL
ISOLATION
Transaction-1
INSERT-1
UPDATE-1
Transaction-2
INSERT-2
UPDATE-2
Isolation
PostgreSQL
HTML5ANIMATIONTOGIF
DURABILITY
Transactions
Write-Ahead Log (WAL)
Txn-1, INSERT, Data,
Txn-2, UPDATE, Data,
Txn-3, DELETE, Data
Txn-4, INSERT, Data
PostgreSQL

Example of ACID

- Let's combine the four ACID properties into one example:
- **Operation:** Transfer 100 yuan from Account A to Account B.
  - **Atomicity:** Ensures that the two operations—"deduct 100 from A" and "add 100 to B"—are bundled together. They either both succeed or both fail.
  - **Consistency:** Ensures that the total amount in the database remains unchanged before and after the transfer (the sum of A and B remains the same), and account balances do not become negative.
  - **Isolation:** During the transfer process, if another transaction queries the balance of Account A, it will either see the balance before the transfer or the balance after the transfer, but never an "intermediate state" (for example, seeing that A has been debited but B has not yet received the money).
  - **Durability:** Once the bank system prompts "Transfer Successful," the result is permanently effective and will not disappear due to a system restart.

ACID vs. BASE

What does ACID Mean?



BASE

Database Transaction Model

Basically Available

Appears to work most of the time

Soft State

Stores may not be write-consistent. Replicas may not be mutually consistent with each other.

Eventual Consistency

Writes will respond with SUCCESS before the data is actually persisted to disk. This is one of the factors that contribute to the huge performance benefit.

The drawback? You might not see the data reflected in time.

Key Advantage

Some applications don't benefit from the strict consistent of the ACID model.

For example, Twitter users may not mind seeing their tweet take a short moment before seeing it posted.

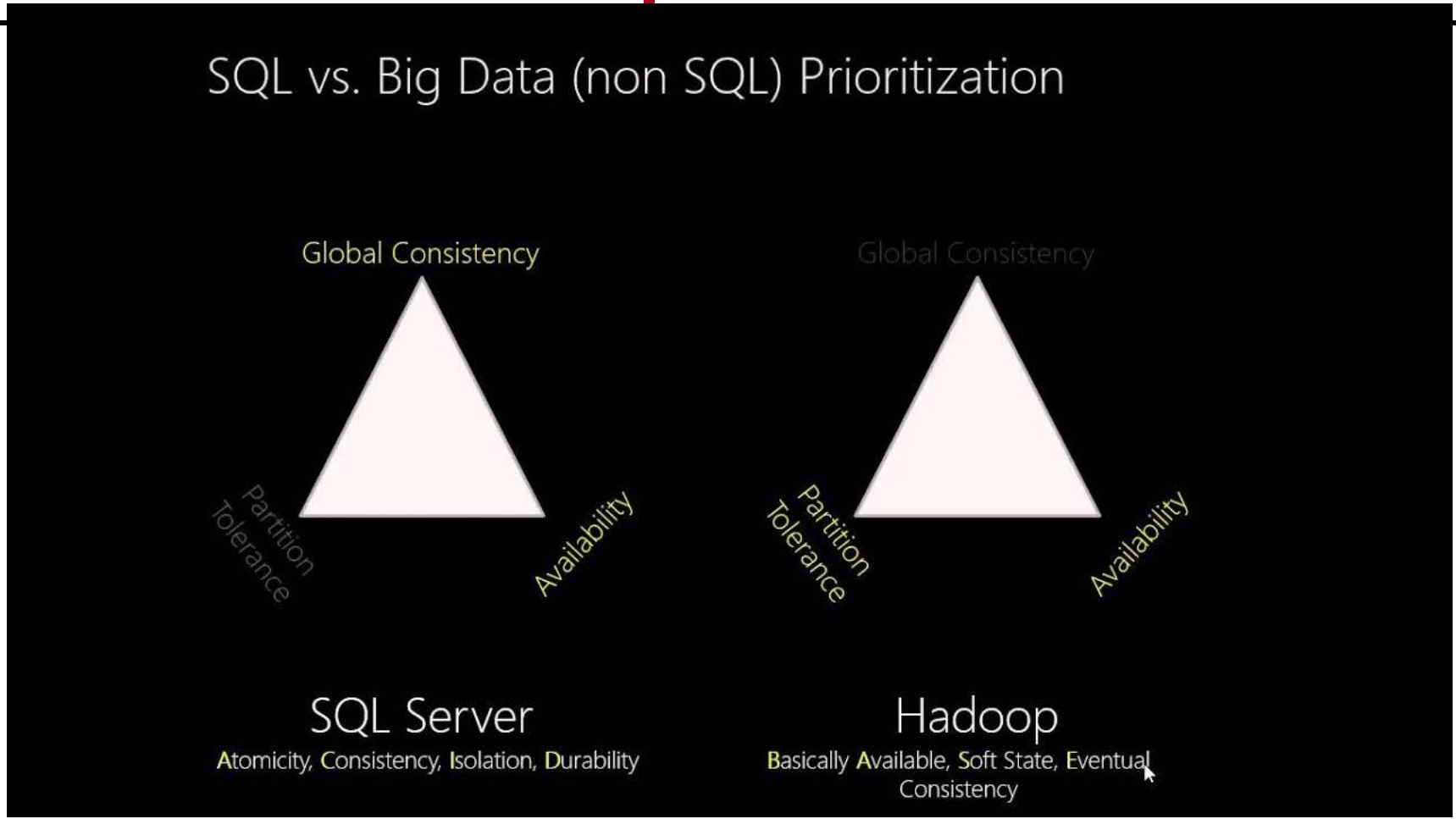
Make sure it fits your use-case

A popular example of when NOT to use BASE: financial transactions.

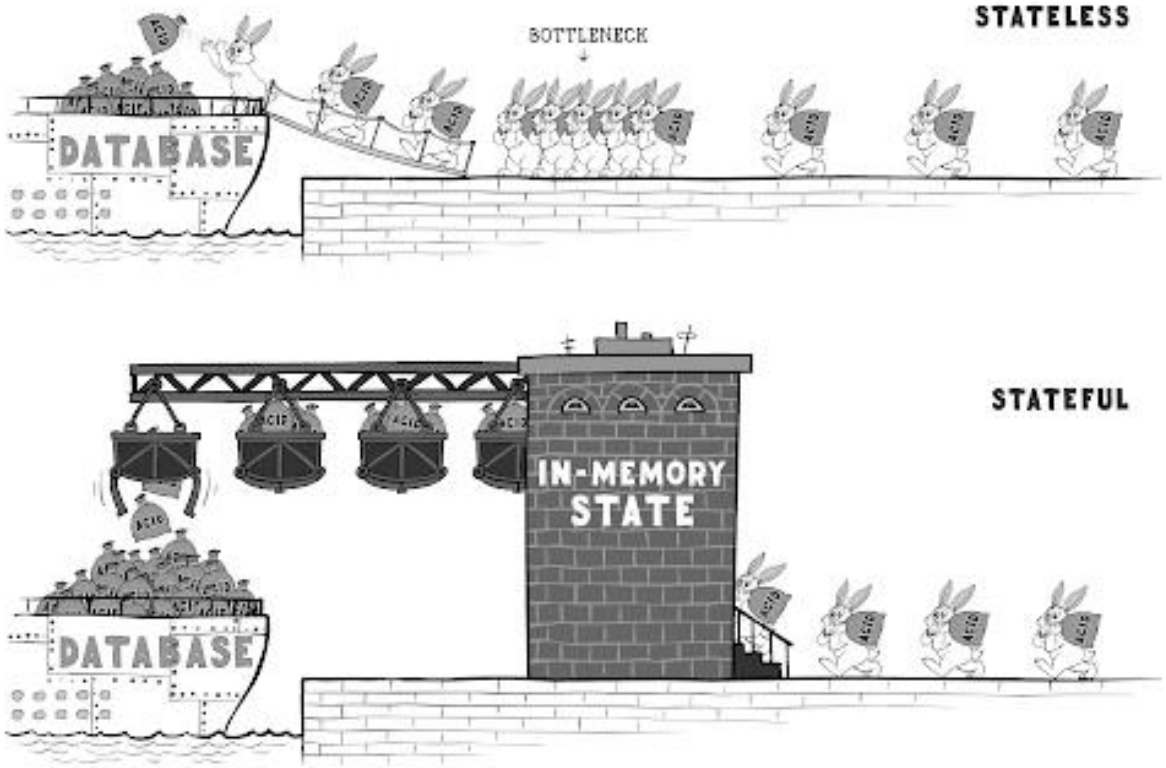
Every single transaction is important.

Every single transaction needs to be properly reflected and kept consistent throughout the entire system.

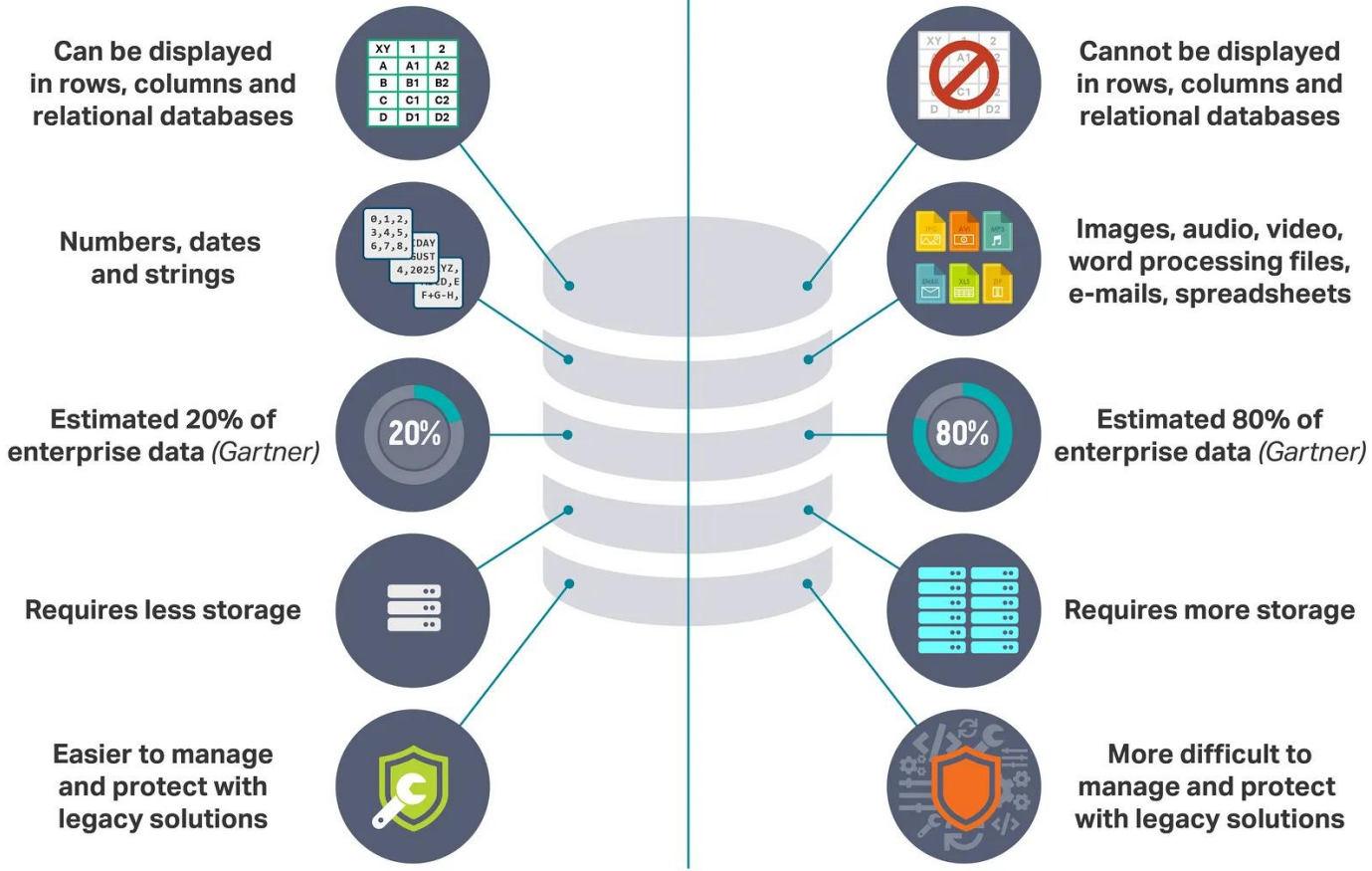
BASE for Hadoop



Stateless


STATELESS
BOTTLENECK
DATABASE
STATEFUL
IN-MEMORY STATE
DATABASE

Structured Data vs Unstructured Data

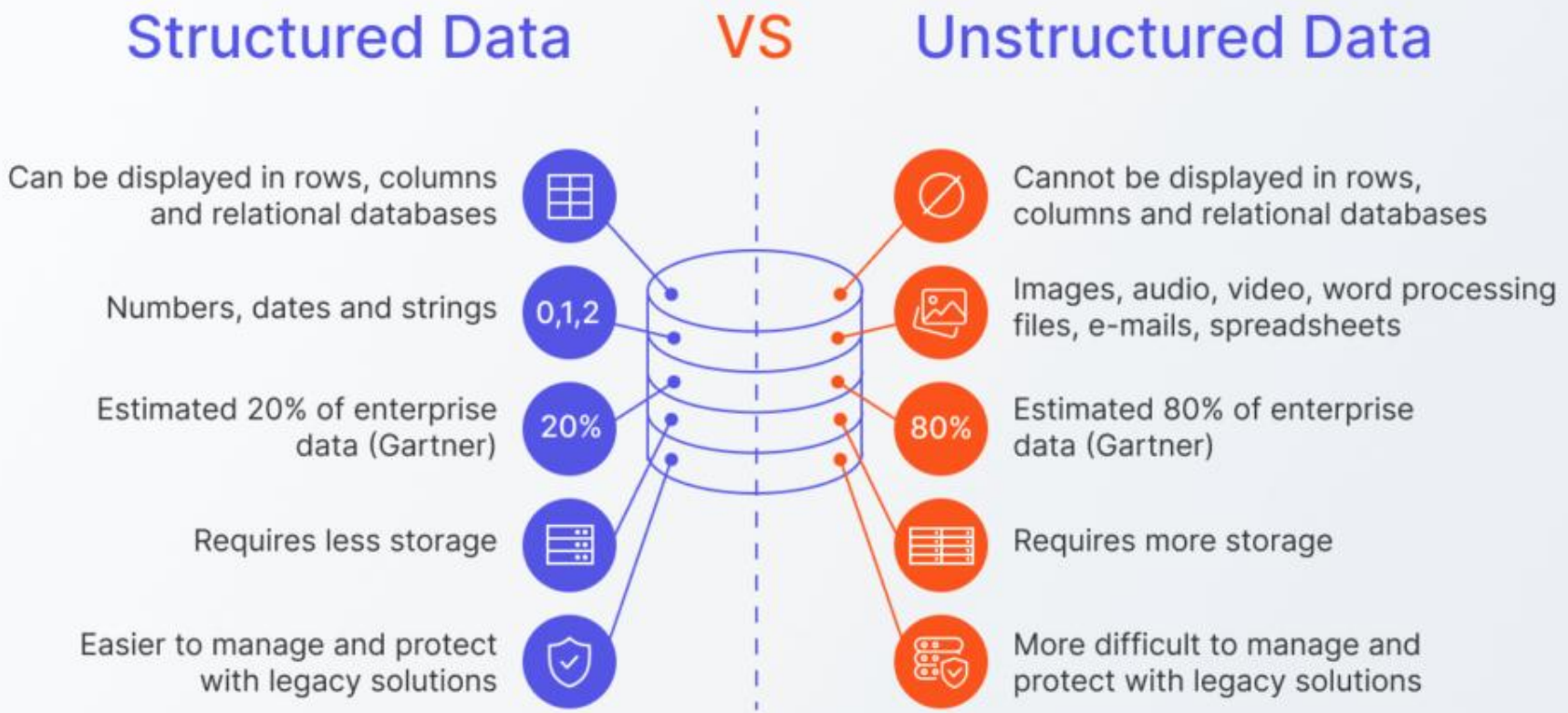


Example: Unstructured Data

- Variety: This type of data can include a wide range of formats, such as:
  - Text documents (e.g., emails, reports, articles)
  - Multimedia files (e.g., images, audio, video)
  - Social media content (e.g., posts, comments, tweets)
  - Web pages and blogs

Applications

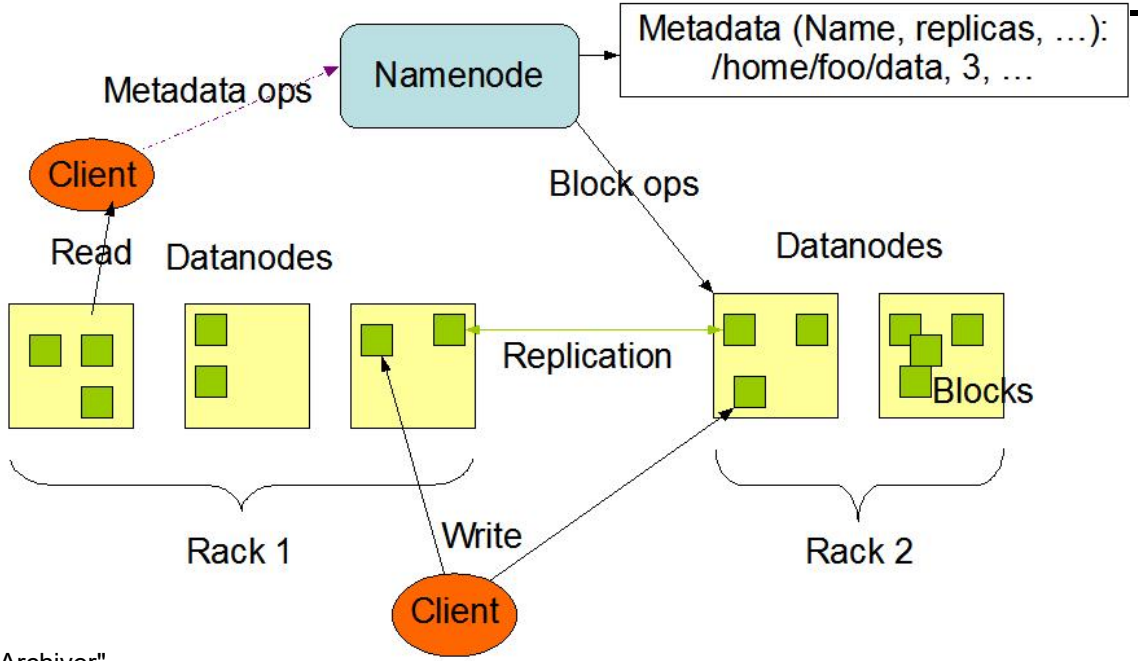
- Unstructured data is already being used across industries:
  - Healthcare: Doctors use unstructured patient records, lab notes and imaging reports to diagnose and personalize treatment.
  - Retail: Analyzing customer reviews and social media comments to improve product quality and customer experience.
  - Finance: Processing news feeds, analyst reports and customer emails to manage risk and improve investment decisions.
  - Legal: Automating document review and e-discovery in law firms through text mining.
  - Media & Entertainment: Recommending content based on viewing habits, comments and user preferences.


Structured Data
VS
Unstructured Data
Can be displayed in rows, columns and relational databases
Numbers, dates and strings
0,1,2
Estimated 20% of enterprise data (Gartner)
20%
Requires less storage
Easier to manage and protect with legacy solutions
Cannot be displayed in rows, columns and relational databases
Images, audio, video, word processing files, e-mails, spreadsheets
Estimated 80% of enterprise data (Gartner)
80%
Requires more storage
More difficult to manage and protect with legacy solutions


PCI
Sensitive
PHI Stale data PII IP
Access control Shared links
Sensitive Secret Public PHI
PHI Stale data PII IP PCI
4 REASONS
UNSTRUCTURED DATA
REMAINS A CHALLENGE
AND HOW TO SOLVE THEM
01
Data Proliferation and Lack of Visibility
As unstructured data grows it becomes more difficult to organize and secure. Implement a solution to continuously discover, classify, and manage unstructured data to improve governance, strengthen data protection, and optimize costs.
02
Scalability and Speed Limitations
Existing solutions & approaches can't keep up with the rapidly evolving data and risk landscape. Stay proactive by using scalable platforms that deploy A.I. and machine learning to accurately scan and process unstructured data as fast as it's being generated.
03
Data Siloes and Integration Challenges
Fragmentation hampers effective data management and protection. Implement a unified data management platform that can bridge these silos to enhance data visibility and control across all enterprise data repositories.
04
Lack of Ownership and Accountability
Who is responsibile for managing and securing your data? Develop a strategy to continuously discover and classify unstructured data and ensure there is a convergence of data management initiatives across departments to effectively assign roles and responsibilities.

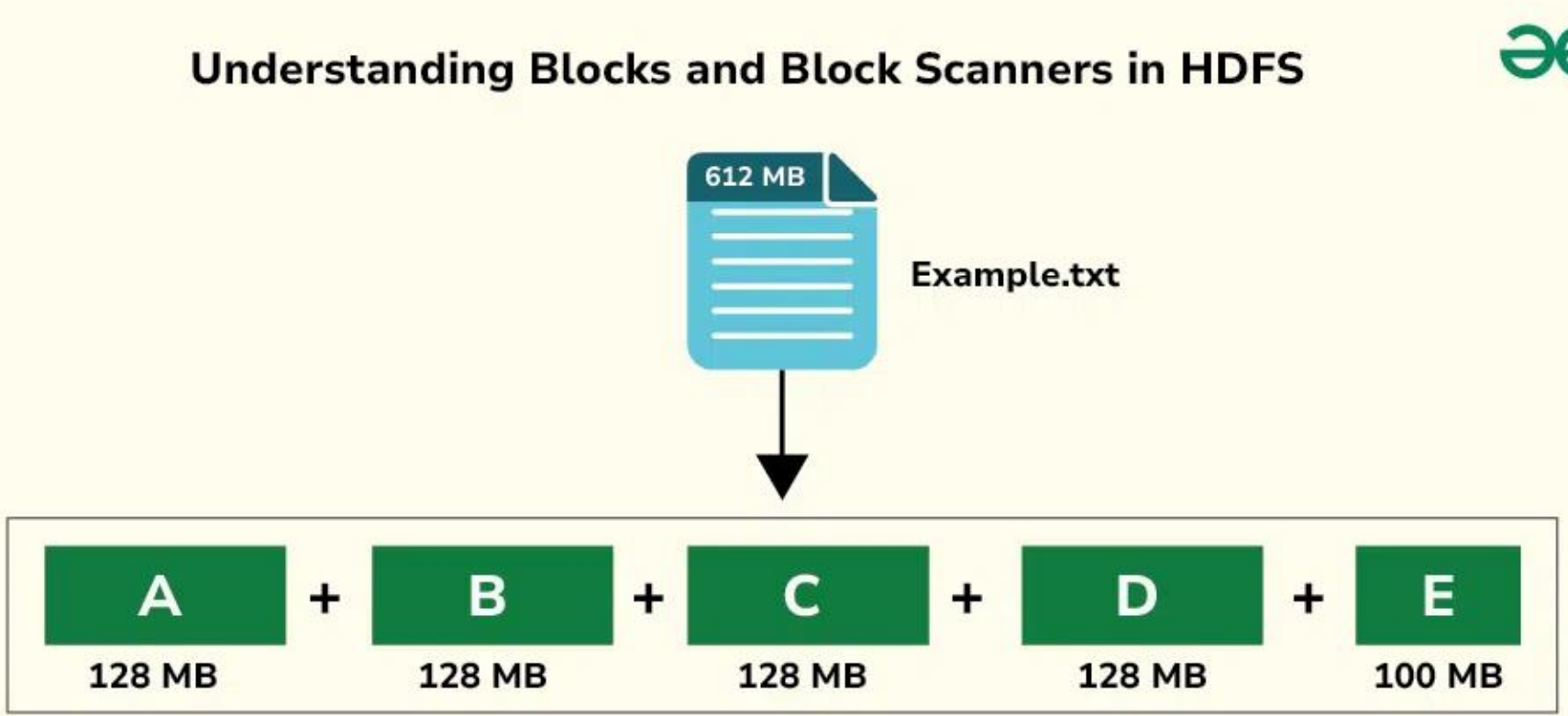
HDFS: Design Goals

- Storing extremely large files (GB/TB levels).
- Streaming data access (write once, read many times).
- Deployment on inexpensive commodity hardware.
- High fault tolerance.



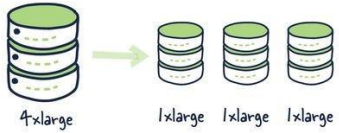
- NameNode (Master): "Archiver".
  - Responsibilities: Manages the file system namespace (metadata) and maintains the file system tree.
  - Does not store actual data.
- DataNode (Slave): "Storekeeper".
  - Responsibilities: Stores actual data blocks and responds to client read and write requests.

HDFS Block

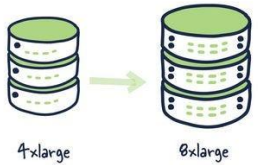

Understanding Blocks and Block Scanners in HDFS
612 MB
Example.txt
A
B
C
D
E
128 MB
128 MB
128 MB
128 MB
100 MB

DATABASE SHARDING EXPLAINED

Horizontal scaling



Vertical scaling



Partition Strategies

Customer ID	First name	Last name	Favorite color
1	TAEKO	OHNUKI	blue
2	O.V.	WRIGHT	green
3	SELDA	BAGCAN	purple
4	JIM	PEPPER	aubergine

Vertical Partitions

VP1

Customer ID	First name	Last name
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDA	BAGCAN
4	JIM	PEPPER

VP2

Customer ID	Favorite color
1	blue
2	green
3	purple
4	aubergine

Horizontal Partitions

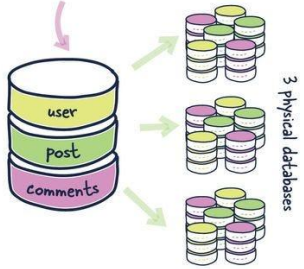
HP1

Customer ID	First name	Last name	Favorite color
1	TAEKO	OHNUKI	blue
2	O.V.	WRIGHT	green

HP2

Customer ID	First name	Last name	Favorite color
3	SELDA	BAGCAN	purple
4	JIM	PEPPER	aubergine

Monolith



Range Based Sharding

Product	Price
WIDGET	$118
GIZMO	$88
TRINKET	$37
THINGAMAJIG	$18
DOODAD	$60
TCHOTCHKE	$999

$0–$49.99

Product	Price
TRINKET	$37
THINGAMAJIG	$18

$50–$99.99

Product	Price
GIZMO	$88
DOODAD	$60

$100+

Product	Price
WIDGET	$118
TCHOTCHKE	$999

Even distribution

Uneven distribution

Key Based Sharding

Shard Key

Column 1	Column 2	Column 3
A		
B		
C		
D		

Hash function

Column 1	Hash values
A	1
B	2
C	1
D	2

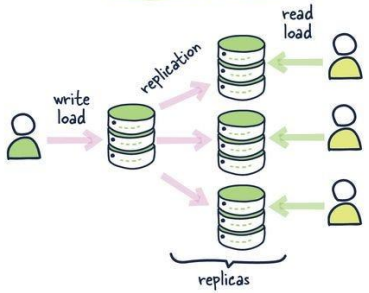
Shard 1

Column 1	Column 2	Column 3
A		
C		

Shard 2

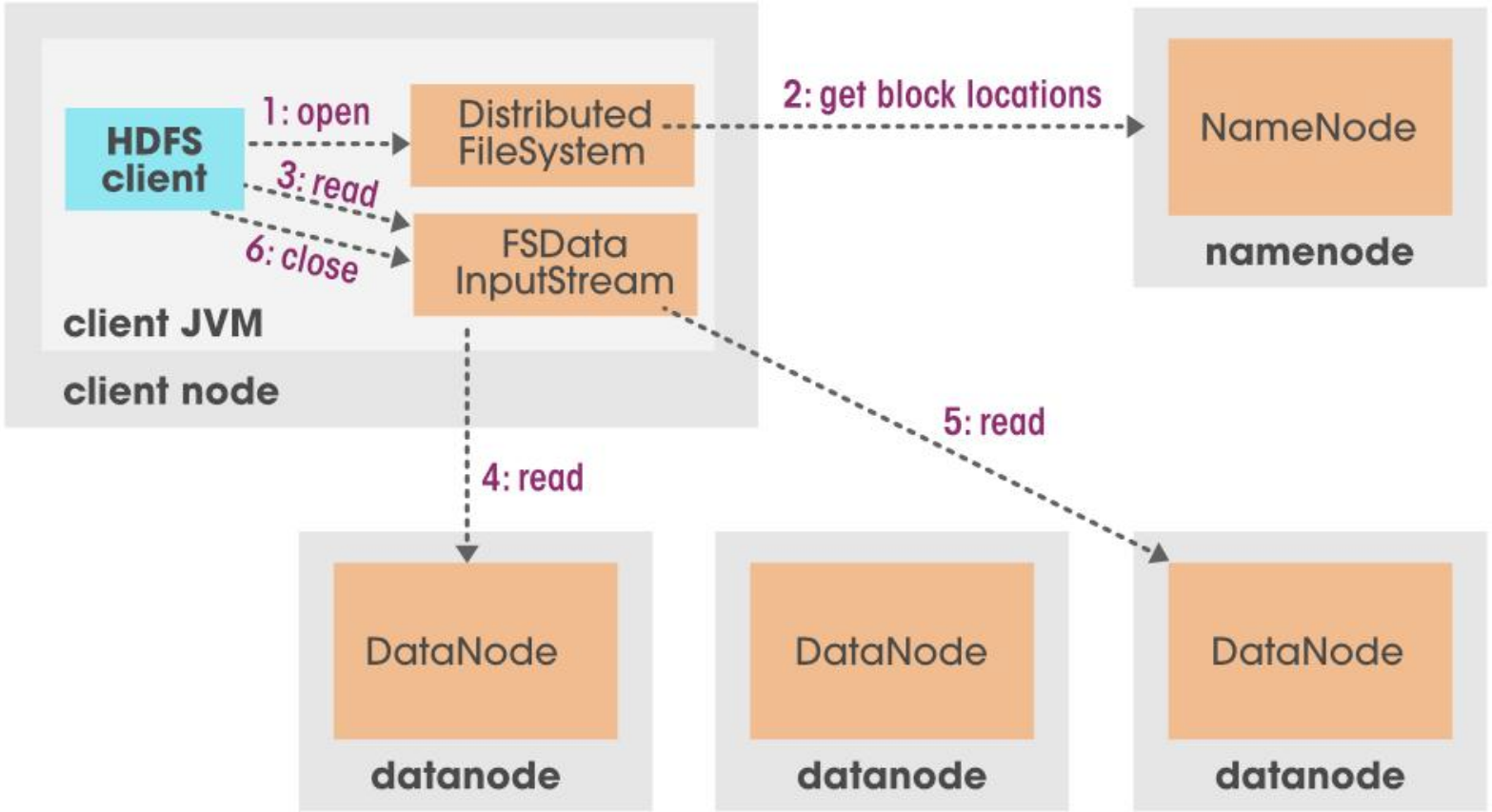
Column 1	Column 2	Column 3
B		
D		

Scaling Reads

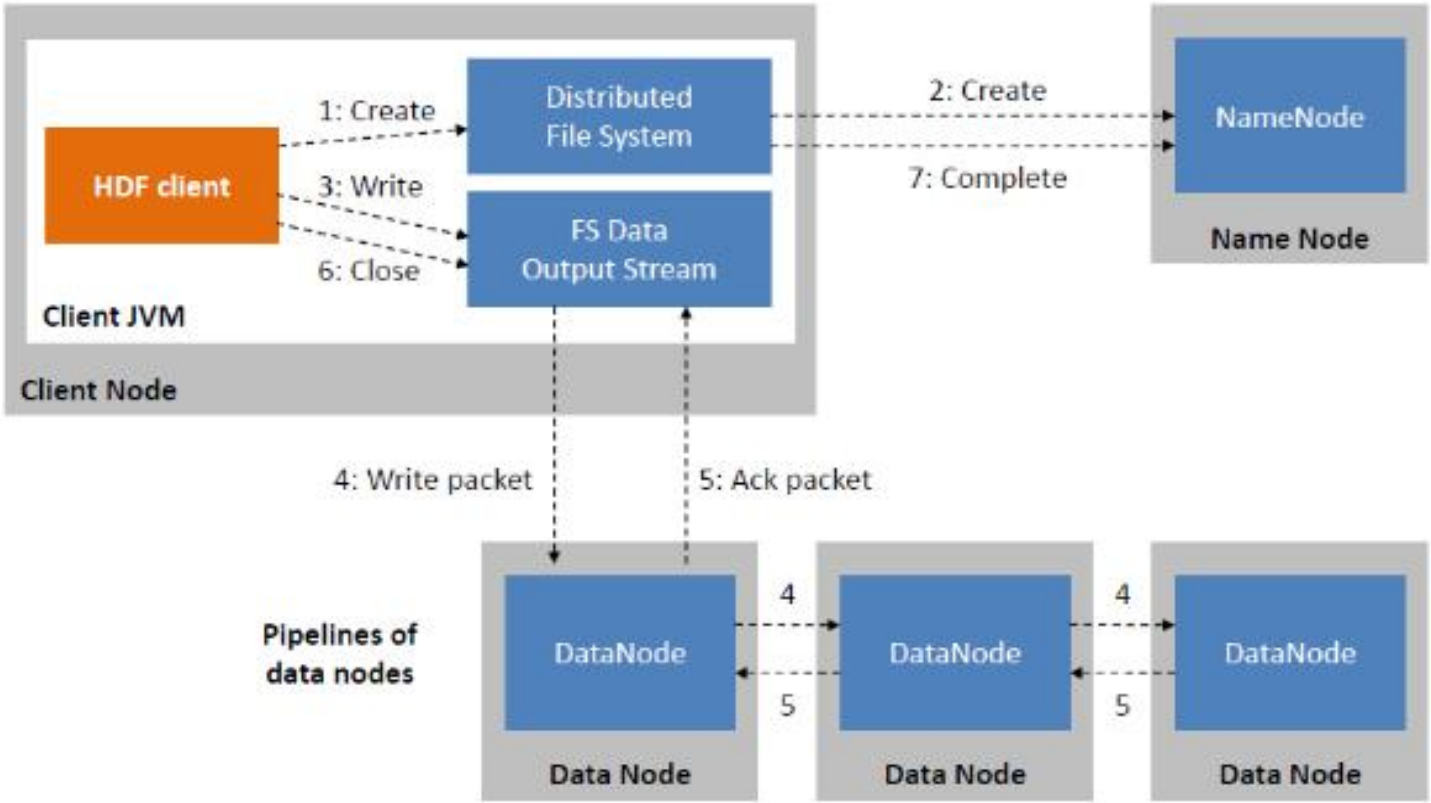


architecture notes

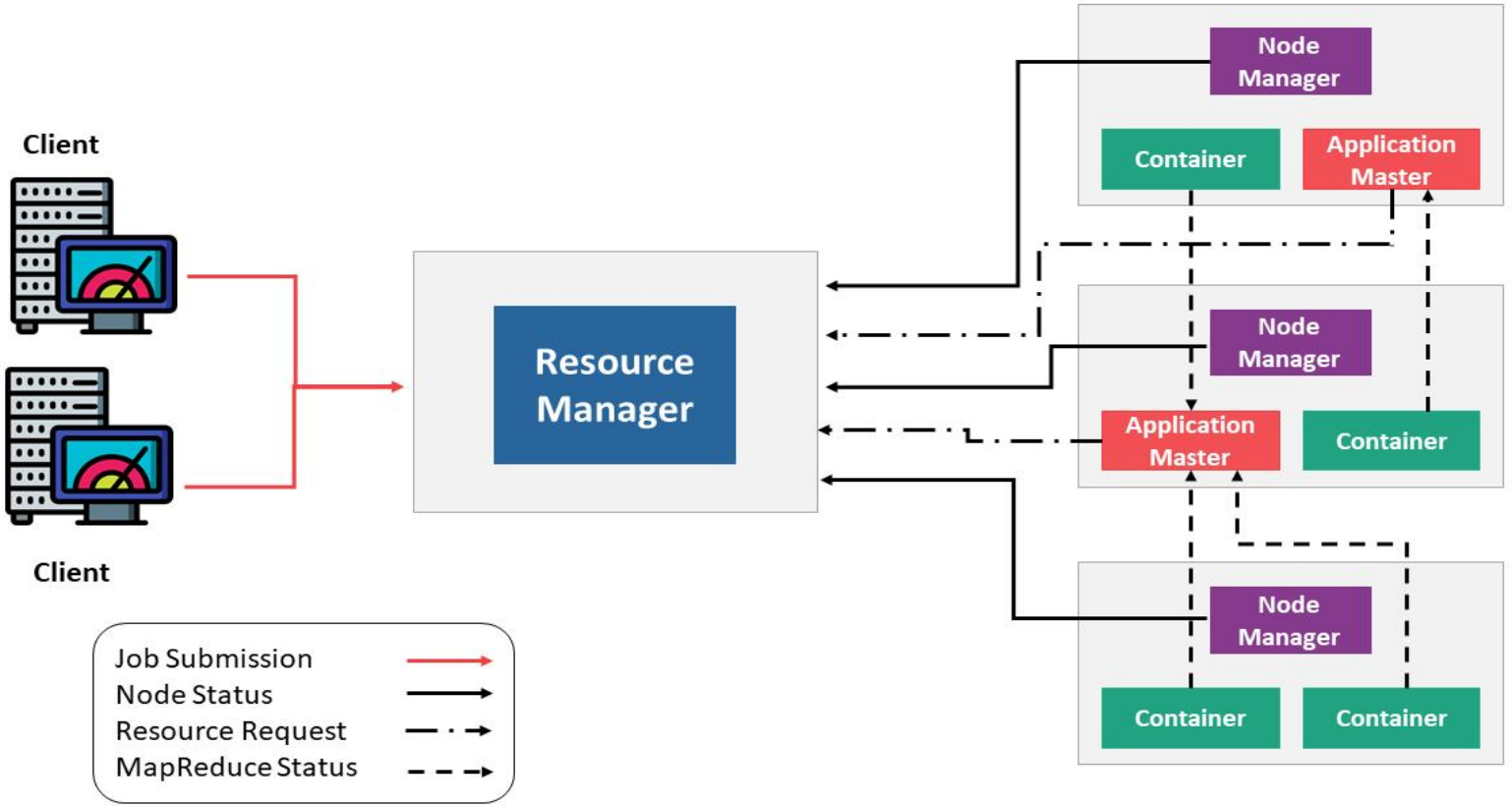
Read Process



Write Process



YARN


Client
Client
Resource Manager
Node Manager
Container
Application Master
Node Manager
Application Master
Container
Node Manager
Container
Container
Job Submission
Node Status
Resource Request
MapReduce Status

What is Apache Hive?

www.decipherzone.com

How it Works?

The working flow of Hive entails converting queries of HiveQL into Tez jobs or MapReduce as both operate on Yet Another Resource Negotiator (YARN). These queries then run on distributed storage solutions like Amazon S3 or HDFS.

OVERVIEW

Apache Hive refers to a fault-tolerant, distributed **data warehouse system** that allows massive-scale data analytics.

FEATURES

- Hive Server 2
- Hive ACID
- Hive Metastore Server
- Hive Replication
- Hive Beeline Shell
- Hive LLAP
- Security & Observability

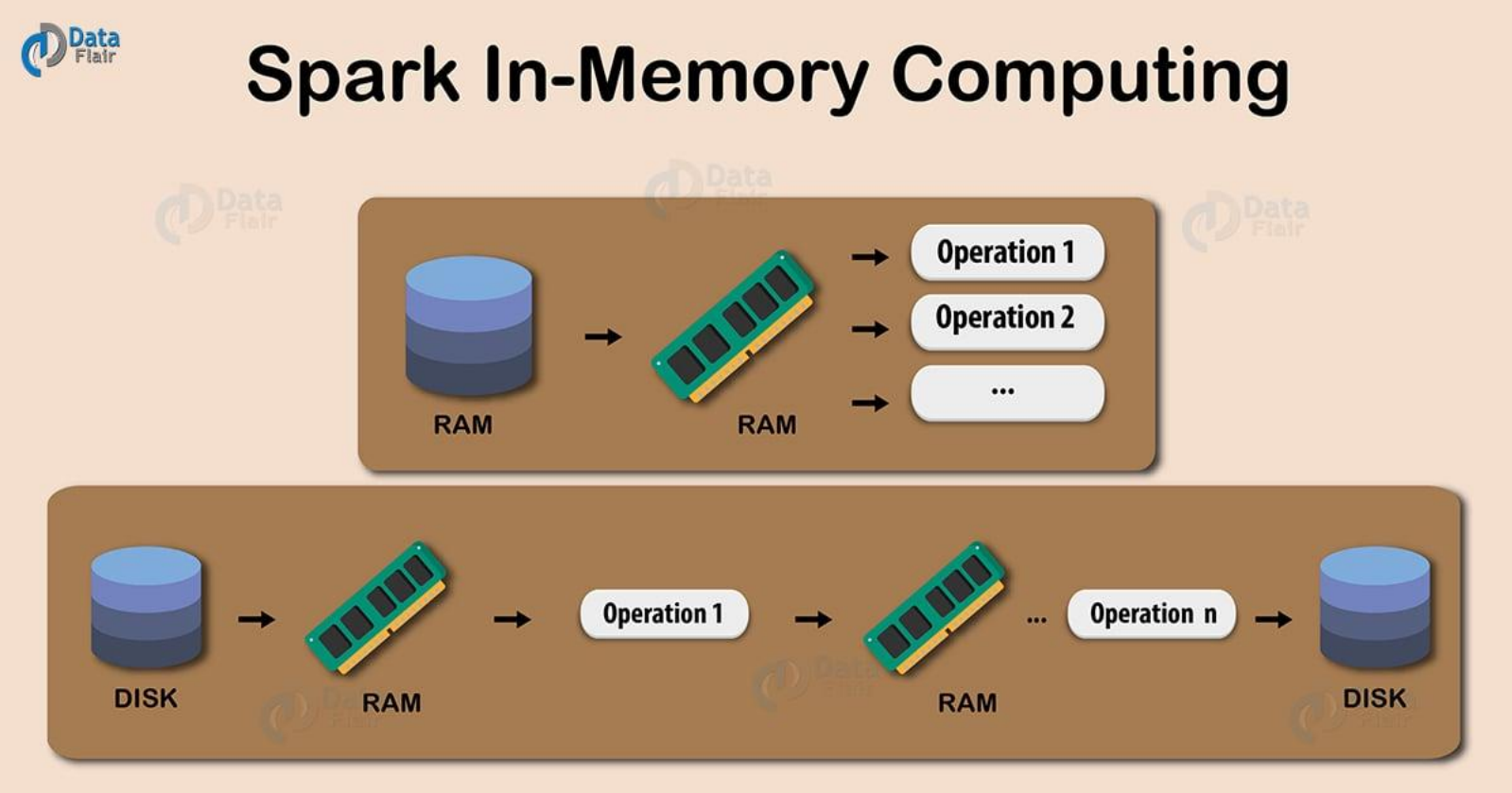
WHAT IS DATA WAREHOUSE?

A data warehouse is a system designed to support data analysis and reporting.

Example

- https://www.simplilearn.com/tutorials/hadoop-tutorial/hive

Spark: In-Memory Computing



Core Innovation

- Core Innovation 1: In-Memory Computing
  - Spark stores intermediate data and results in memory as much as possible, spilling them to disk only when memory is insufficient or persistence is required.
  - Analogy: MapReduce is like a mathematician who records every step of a calculation with pen and paper (on disk), while Spark is a master of mental arithmetic (in-memory).
- Core Innovation 2: Unified Engine
  - Spark provides a powerful general-purpose execution engine that supports multiple computing models, such as SQL queries, stream processing, machine learning, and graph computing.
  - Value: Users only need to learn a single framework to solve a variety of problems, reducing learning costs and system complexity.