
Department of Computer Science and Engineering

Bigdata and Analytics (BCS714D)

Module-01

Data: It refers to raw, unorganized facts or figures.

- It can be numbers, words, measurements, or observations.
- **Example:** "John, 25, Male" — this is data.

Information is processed or organized data that is meaningful and useful.

- It is data given context.
- **Example:** "John is a 25-year-old male customer." — This is information derived from data.

What is Big Data?

- **Big Data** refers to extremely large and complex datasets that traditional tools can't process efficiently.
- It is characterized by **5 V's**:
 - **Volume:** Huge amount of data.
 - **Velocity:** Speed at which data is generated.
 - **Variety:** Different formats (text, video, images, etc.).
 - **Veracity:** Uncertainty of data accuracy.
 - **Value:** Extracting useful insights.

Big Data Analytics: **Big Data Analytics** is the process of examining big data to uncover patterns, trends, and insights.

- It helps businesses make informed decisions.
- Uses tools like **Hadoop, Spark, Hive, Pig**, etc.
- **Example:** Analyzing customer purchase history to recommend products.

2.1 Characteristics of Data

Three key characteristics of data:

1. Composition
 - Structure of data: sources, types, granularity.
 - Static vs. real-time data.
2. Condition
 - State of the data (clean, raw, needs enhancement).
 - Questions: Can it be used as-is? Does it need processing?
3. Context
 - Background of the data: where, when, why was it generated?

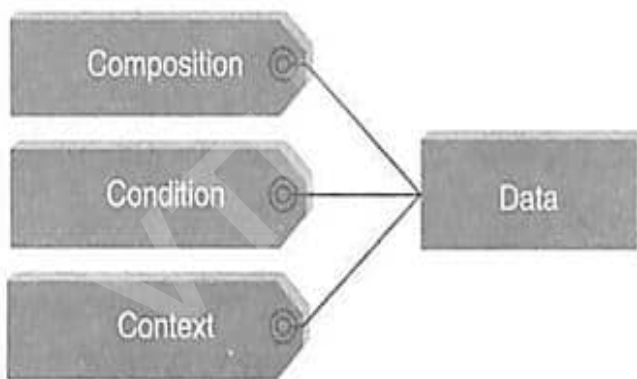


Figure 2.1 Characteristics of data.

2.2 Evolution of Big Data

◆ Overview:

- **Before 1970s:** Data was **primitive and structured**; stored using **mainframes**.
- **1980s–1990s:** Rise of **relational databases**; data became **relational** and used in **data-intensive applications**.

- **2000s & beyond:** Explosion of **complex, unstructured** data due to **Web, IoT**, and **multimedia** technologies.

◆ **Table 2.1 – Evolution of Big Data**

Time Period	Data Generation & Storage	Data Utilization	Data Driven Output
1970s & before	Primitive and Structured	Mainframes for basic data storage	—
1980s & 1990s	Complex and Relational	Relational databases; data-intensive apps	—
2000s and beyond	Complex and Unstructured	—	Structured, unstructured, multimedia data

◆ **Key Contributors to Modern Big Data**

- **World Wide Web (WWW)** – Opened floodgates for global digital content.
- **Internet of Things (IoT)** – Devices constantly generate vast, diverse data streams.

■ 2.3 Definition of Big Data

◆ **Common Understandings of Big Data**

Several responses have been widely accepted to define Big Data:

1. **Beyond Infrastructure:**

Big Data refers to **anything that exceeds traditional storage, processing, and analysis capabilities** of existing human and technical infrastructure.

2. **Relative to Time:**

What is considered "big" today may become **normal or standard tomorrow**, indicating that Big Data is a **moving threshold**.

3. **Large Volume:**

It refers to enormous data sizes – **Terabytes, Petabytes, Zettabytes**, and even **Yottabytes**.

4. **The 3 Vs** – A standard definition framework.

◆ **Standard 3Vs Definition (Gartner's Perspective)**

According to **Gartner IT Glossary** and **Doug Laney (2001)**, Big Data is defined by:

Big Data is high-volume, high-velocity, and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.

- **Volume:** Huge amounts of data.
- **Velocity:** Speed of data generation and processing.
- **Variety:** Different types and formats (structured, unstructured, semi-structured).

◆ **Key Conceptual Flow**

Big Data → Information → Actionable Intelligence → Better Decisions → Enhanced Business Value

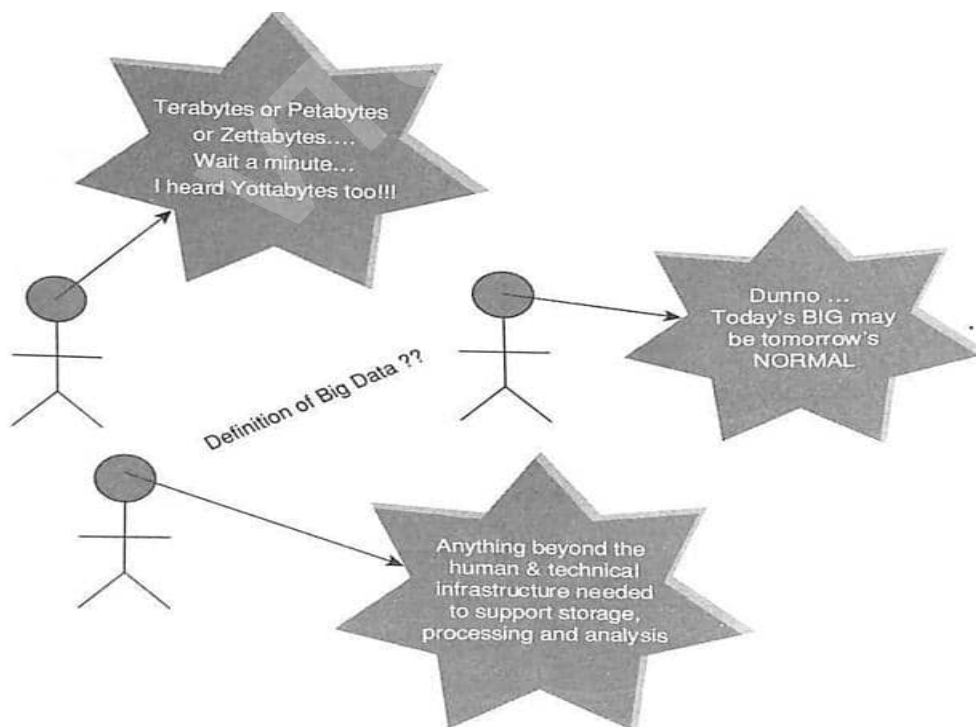


Figure 2.2 Definition of big data.

1. Volume-Based Definition

- Focuses on the **massive size** of data.
- Big Data is often defined by its **scale**, measured in **terabytes, petabytes, zettabytes**, or even **yottabytes**.

★ 2. Infrastructure-Based Definition

- Refers to data that **exceeds the capacity** of current tools or infrastructure.
- Suggests Big Data begins where **traditional systems struggle** to store, process, or analyze data efficiently.

★ 3. Relativity Definition

- Emphasizes the **relative and evolving nature** of Big Data.
- What we call “big” today may become standard in the near future due to **technological advances**.

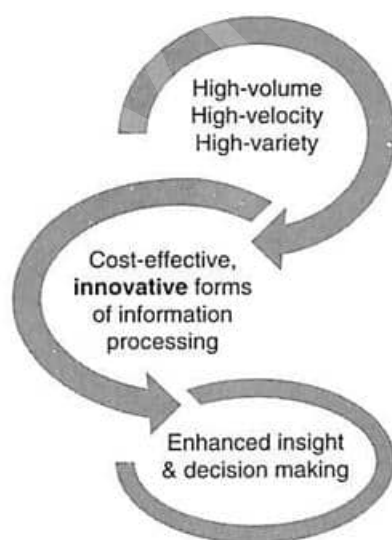


Figure 2.3 Definition of big data – Gartner.

Definition of Big Data (Gartner's Perspective)

Three-Part Definition (3Vs Model):

"Big Data is high-volume, high-velocity, and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."

◆ Part I: High-Volume, High-Velocity, High-Variety

- Refers to large and fast-growing data.
- Includes:
 - **Structured, semi-structured, and unstructured** data.
- Requires:
 - Fast **storage, processing, and analysis** capabilities.

◆ Part II: Cost-Effective & Innovative Processing

- Focuses on:
 - Using **new technologies and methods** to: Ingest, Store, Process, Persist, Integrate, Visualize.
- Goal:
 - Efficient handling of large, fast, and diverse data sets.

◆ Part III: Enhanced Insight & Decision-Making

- Involves:
 - Gaining **deep, rich, and actionable insights**.
 - Using those insights for **better and faster decisions**.
- Leads to:
 - Improved **business value** and **competitive advantage**.

Flow Summary:

Data → Information → Actionable Intelligence → Better Decisions → Enhanced Business Value

Challenges with Big Data

1. Exponential Data Growth

- Data volume is increasing rapidly, mostly generated in the last 2-3 years.
- Raises key questions:
 - Will all this data be useful for analysis?
 - Should we work with all data or a subset?
 - How to separate valuable knowledge from noise?

2. Cloud Computing and Virtualization

- Cloud is essential for cost-efficient, elastic, and scalable infrastructure.
- However, deciding whether to host big data solutions on-premises or in the cloud is complex.

3. Data Retention Period

- Determining how long to keep big data is tricky.
- Some data is valuable for long-term decisions.
- Other data becomes irrelevant or obsolete quickly (sometimes within hours).

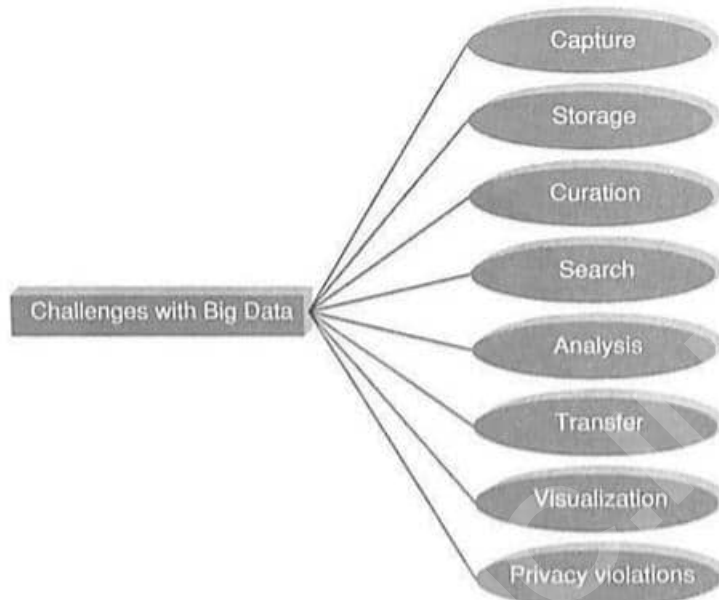


Figure 2.4 Challenges with big data.

2.5 What is Big Data?

- Big data is defined by three main characteristics:
 - Volume (amount of data)
 - Velocity (speed of data generation and processing)
 - Variety (different types and formats of data)

2.5.1 Volume

- Data size has grown exponentially, evolving through units:
 - Bits → Bytes → Kilobytes → Megabytes → Gigabytes → Terabytes → Petabytes → Exabytes → Zettabytes → Yottabytes
- The sources of big data are diverse and can include:
 - XLS, DOC, PDF (unstructured data)
 - Videos (e.g., YouTube)
 - Chat conversations (e.g., Internet Messenger)
 - Customer feedback forms on retail websites

2.5.1.1 Where does Data get generated?

Data Characteristics Illustrated:

- Data Volume (X-axis, moving outward):
 - Data size grows from small (MB, GB) to very large (TB, PB).
 - The more data you have, the further out on the volume axis it is.
- Data Velocity (Y-axis, moving upward):
 - Speed of data generation and processing:
 - From Batch processing (slowest) to Periodic, Near real-time, and Real-time (fastest).
- Data Variety (Z-axis, moving diagonally):
 - Types of data include:
 - Table, Database (structured data)
 - Photo, Social, Web, Video, Audio, Mobile (mostly unstructured or semi-structured data)

Data Growth and Units (Table 2.2)

- **Data Size Units (Binary scale):**
 - Bit = 0 or 1
 - Byte = 8 bits
 - Kilobyte (KB) = 1024 bytes
 - Megabyte (MB) = 1024^2 bytes
 - Gigabyte (GB) = 1024^3 bytes
 - Terabyte (TB) = 1024^4 bytes
 - Petabyte (PB) = 1024^5 bytes
 - Exabyte (EB) = 1024^6 bytes

- Zettabyte (ZB) = 1024^7 bytes

- Yottabyte (YB) = 1024^8 bytes

- **Data Size Units (Decimal scale as per Figure 2.6):**

- 1 KB = 1,000 bytes

- 1 MB = 1,000,000 bytes

- 1 GB = 1,000,000,000 bytes

- 1 TB = 1,000,000,000,000 bytes

- 1 PB = 1,000,000,000,000,000 bytes

- 1 EB = 1,000,000,000,000,000,000 bytes

- 1 ZB = 1,000,000,000,000,000,000,000 bytes

- 1 YB = 1,000,000,000,000,000,000,000,000 bytes

Note: Binary units are based on powers of 2 (1024), while decimal units are powers of 10 (1000).

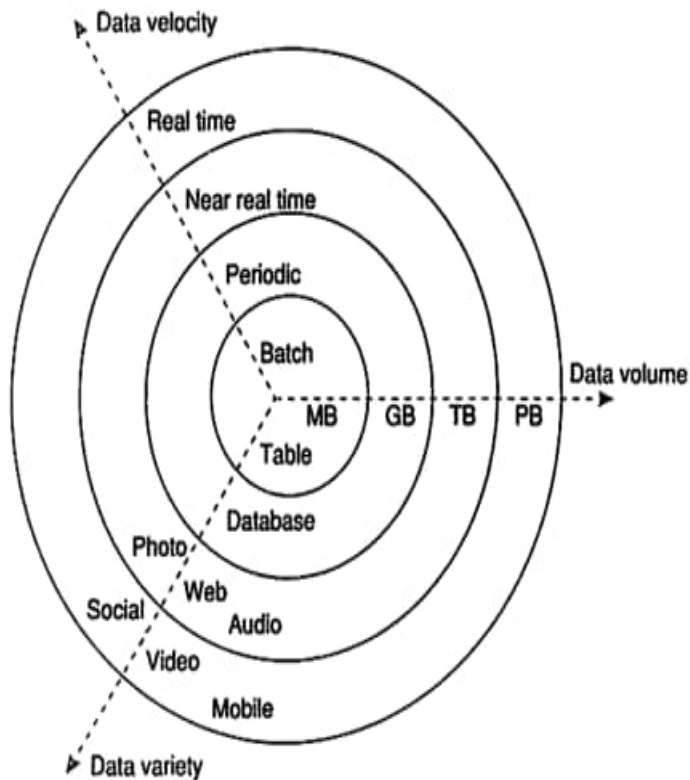


Figure 2.5 Data: Big in volume, variety, and velocity.

Table 2.2 Growth of data

Bits	0 or 1
Bytes	8 bits
Kilobytes	1024 bytes
Megabytes	1024 ² bytes
Gigabytes	1024 ³ bytes
Terabytes	1024 ⁴ bytes
Petabytes	1024 ⁵ bytes
Exabytes	1024 ⁶ bytes
Zettabytes	1024 ⁷ bytes
Yottabytes	1024 ⁸ bytes

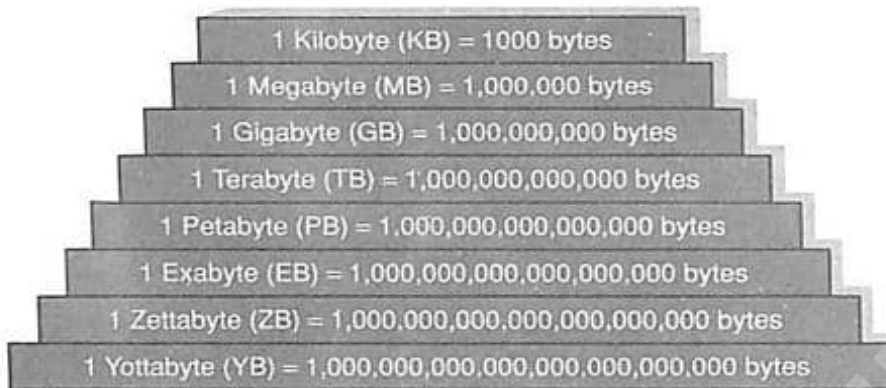


Figure 2.6 A mountain of data.

Sources of Big Data (Figure 2.7)

- **Internal Data Sources:**

- **Data storage:** File systems, SQL databases (Oracle, MySQL, PostgreSQL, MongoDB, Cassandra, etc.)
- **Archives:** Scanned documents, paper archives, customer correspondence, health records, admission records, assessment records, etc.

- **External Data Sources:**

- **Public Web:** Wikipedia, weather, regulatory data, compliance, census, etc.

- **Both Internal + External:**

- **Sensor data:** Car sensors, smart meters, HVAC systems, refrigerators, etc.
- **Machine log data:** Event logs, app logs, business process logs, audit logs, clickstream data.
- **Social media:** Twitter, Facebook, LinkedIn, YouTube, Instagram.
- **Business apps:** ERP, CRM, HR systems, Google Docs.
- **Media:** Audio, video, images, podcasts.
- **Docs:** CSV files, Word documents, PDFs, XLS, PPT.

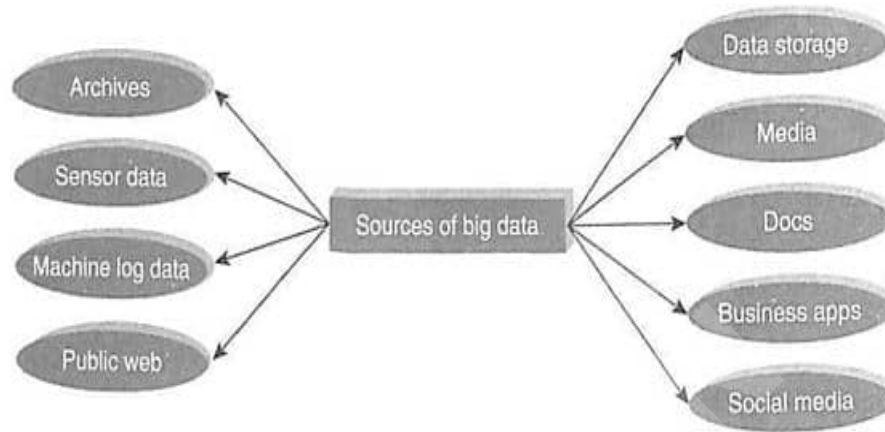


Figure 2.7 Sources of big data.

External Data Sources

- Data that resides **outside** an organization's firewall.
- Examples include:
 - **Public Web:** Wikipedia, weather data, regulatory info, compliance, census data, etc.

Both (Internal + External Data Sources)

- Data that can come from both inside and outside the organization.
- Examples include:
 - **Sensor data:** Car sensors, smart electric meters, office building sensors, air conditioning units, refrigerators, etc.
 - **Machine log data:** Event logs, application logs, business process logs, audit logs, clickstream data, etc.
 - **Social media:** Twitter, blogs, Facebook, LinkedIn, YouTube, Instagram, etc.
 - **Business apps:** ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), HR systems, Google Docs, etc.
 - **Media:** Audio, video, images, podcasts, etc.

-
- **Docs:** CSV (Comma Separated Values), Word documents, PDF, XLS, PPT, and similar files.

2.5.2 Velocity

- Velocity refers to the speed at which data is processed.
- There's been a shift from **batch processing** (e.g., traditional payroll applications) to **real-time processing**.
- The progression in data processing speed is:
Batch → **Periodic** → **Near real-time** → **Real-time processing**

2.5.3 Variety

- Variety deals with the different types and formats of data.
- It classifies data into three categories:
 1. **Structured data:** Comes from traditional systems like transaction processing systems and RDBMS (Relational Database Management Systems).
 2. **Semi-structured data:** Examples include HTML, XML (Hyper Text Markup Language and eXtensible Markup Language).
 3. **Unstructured data:** Includes unstructured text documents, audio, video, emails, photos, PDFs, social media content, etc.

3.2 WHAT IS BIG DATA ANALYTICS?

Big Data Analytics is...

1. **Technology-enabled analytics:**
Uses various data analytics and visualization tools from vendors like IBM, Tableau, SAS, R Analytics, etc., to process and analyze big data.
2. **Gaining meaningful insights:**
Helps businesses gain deeper, richer insights to better understand customers (demographics, preferences) for cross-selling, up-selling, and better vendor/supplier management.
Example: Online retailers recommend products based on stored purchase and preference data.
3. **Competitive edge:**
Enables quicker and better decision-making, giving an advantage over competitors.

4. **Collaboration:**

Involves close cooperation between IT teams, business users, and data scientists.

5. **Handling large datasets:**

Deals with datasets whose volume and variety surpass current storage and processing capabilities of enterprises.

6. **Moving code to data:**

Instead of moving data around, programs (small in size) are moved close to where data resides (especially large datasets in terabytes or petabytes), improving efficiency. This will become more important as data grows to exabytes or zettabytes.

3.5 CLASSIFICATION OF ANALYTICS:

1. **Classification by stages of analytics:**

- Basic
- Operationalized
- Advanced
- Monetized

2. **Classification by versions of analytics:**

- Analytics 1.0
- Analytics 2.0
- Analytics 3.0

Figure 3.5 illustrates what big data entails, showing a cycle:

- More data produced →
- More data stored →
- More data analyzed →
- Better predictions →
- Steady growth of analysis → (loops back to more data produced)

This cycle highlights how increasing data leads to improved insights and predictions, fueling further data generation and analysis.

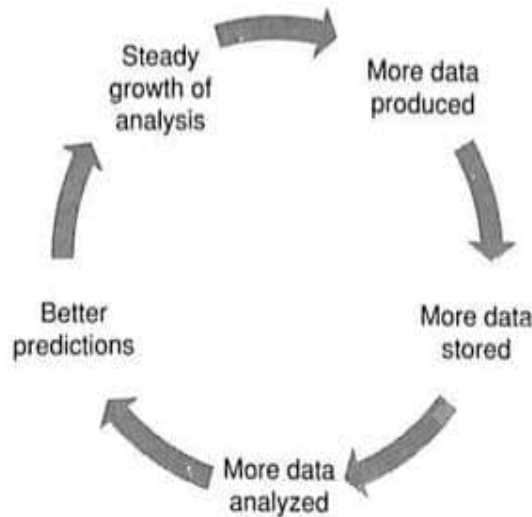


Figure 3.5 What big data entails?

3.5.1 First School of Thought

1. **Basic analytics:** This primarily is slicing and dicing of data to help with basic business insights. This is about reporting on historical data, basic visualization, etc.
2. **Operationalized analytics:** It is operationalized analytics if it gets woven into the enterprise's business processes.
3. **Advanced analytics:** This largely is about forecasting for the future by way of predictive and prescriptive modeling.
4. **Monetized analytics:** This is analytics in use to derive direct business revenue.

3.5.2 Second School of Thought

Table 3.1 Analytics 1.0, 2.0, and 3.0

	Analytics 1.0	Analytics 2.0	Analytics 3.0
Era:	Mid 1950s to 2009	2005 to 2012	2012 to present
Stats used:	Descriptive statistics (report on events, occurrences, etc. of the past)	Descriptive statistics + predictive statistics (use data from the past to make predictions for the future)	Descriptive + predictive + prescriptive statistics (use data from the past to make prophecies for the future and at the same time make recommendations to leverage the situation to one's advantage)

	Analytics 1.0	Analytics 2.0	Analytics 3.0
Key questions asked:	What happened? Why did it happen?	What will happen? Why will it happen?	What will happen? When will it happen? Why will it happen? What should be the action taken to take advantage of what will happen?
Data sources:	Data from legacy systems, ERP, CRM, and 3rd party applications. Small and structured data sources. Data stored in enterprise data warehouses or data marts.	Big data is being taken up seriously. Data is mainly unstructured, arriving at a much higher pace. This fast flow of data entailed that the influx of big volume data had to be stored and processed rapidly, often on massive parallel servers running Hadoop.	A blend of big data and data from legacy systems, ERP, CRM, and 3rd party applications. A blend of big data and traditional analytics to yield insights and offerings with speed and impact.
Sourcing:	Data was internally sourced.	Data was often externally sourced.	Data is both being internally and externally sourced.
Technology:	Relational databases	Database appliances, Hadoop clusters, SQL to Hadoop environments, etc.	In memory analytics, in database processing, agile analytical methods, machine learning techniques, etc.

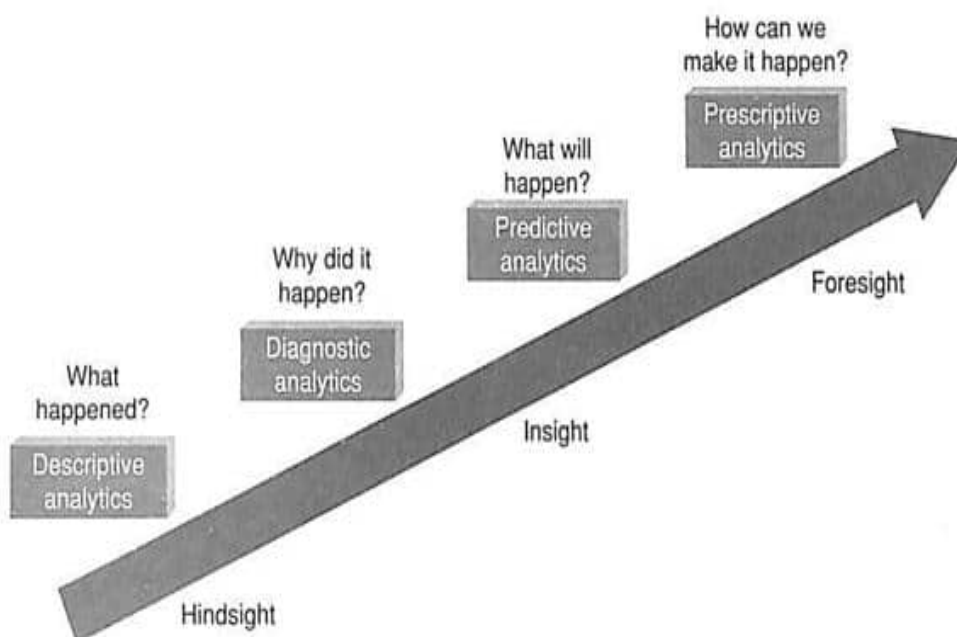


Figure 3.6 Analytics 1.0, 2.0, and 3.0.

3.8 WHY IS BIG DATA ANALYTICS IMPORTANT?

1. **Reactive – Business Intelligence:**

What does Business Intelligence (BI) help us with? It allows the businesses to make faster and better decisions by providing the right information to the right person at the right time in the right format. It is about analysis of the past or historical data and then displaying the findings of the analysis or reports in the form of enterprise dashboards, alerts, notifications, etc. It has support for both pre-specified reports as well as ad hoc querying.

2. **Reactive – Big Data Analytics:**

Here the analysis is done on huge datasets but the approach is still reactive as it is still based on static data.

3. **Proactive – Analytics:**

This is to support futuristic decision making by the use of data mining, predictive modeling, text mining, and statistical analysis. This analysis is not on big data as it still uses the traditional database management practices on big data and therefore has severe limitations on the storage capacity and the processing capability.

4. **Proactive – Big Data Analytics:**

This is sieving through terabytes, petabytes, exabytes of information to filter out the relevant data to analyze. This also includes high performance analytics to gain rapid insights from big data and the ability to solve complex problems using more data.

3.12 TERMINOLOGIES USED IN BIG DATA ENVIRONMENTS

1. **In-Memory Analytics**

- Data is fetched from RAM (Random Access Memory) instead of slow hard disk storage.
- Pre-processed and frequently-used data (e.g., cubes, aggregates) is stored in-memory.
- Enables faster access, quicker deployment, better insights, and minimal IT involvement.
- Eliminates repeated disk reads for analysis.

2. **In-Database Processing**

- Also known as in-database analytics.
- Combines data warehouses with analytical systems.
- Analytical computations are done within the database itself.

-
- Avoids time-consuming export of data.
 - Saves time and supports complex/extensive computations.

3. Symmetric Multiprocessor System (SMP)

- Uses a single main memory shared by two or more identical processors.
- All processors:
 - Have access to I/O devices.
 - Are controlled by a single OS instance.
- Processors are tightly coupled.
- Each processor has its own high-speed cache memory.
- Communication happens via a system bus.

4. Massively Parallel Processing (MPP)

- Involves multiple processors working in parallel on different parts of the same program.
- Each processor:
 - Has its own OS and dedicated memory.
 - Communicates through messaging interfaces.
- Suitable for large-scale data analytics.
- More complex to program than SMP.
- Referred to as loosely coupled systems (unlike tightly-coupled SMP).

5. Difference Between Parallel and Distributed Systems

- Parallel systems:
 - Are tightly coupled.
 - Use shared memory.
 - User is unaware of how tasks are parallelized.
 - Common in parallel databases.
- Distributed systems (explained later in text):
 - Are loosely coupled.
 - Use multiple systems possibly at different locations.

- Require coordination over a network.

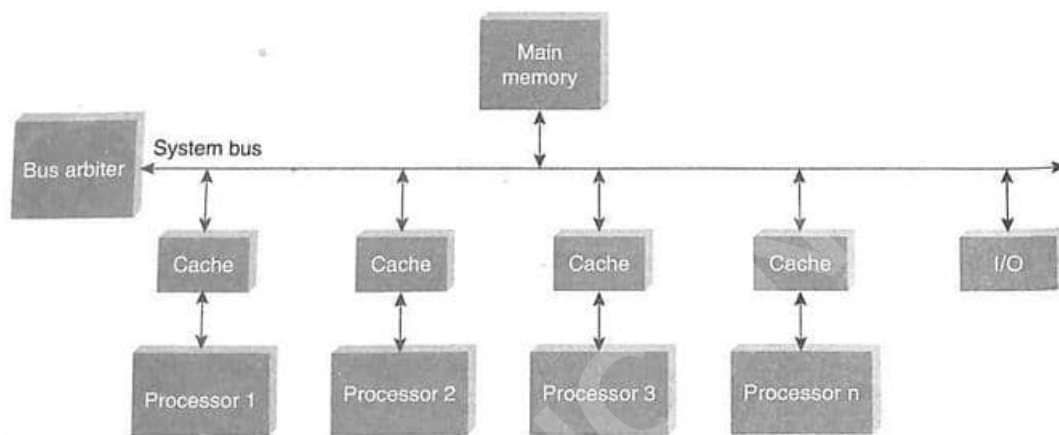


Figure 3.9 Symmetric Multiprocessor System.

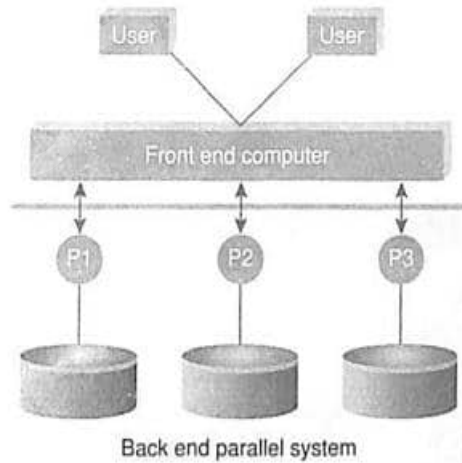


Figure 3.10 Parallel system.

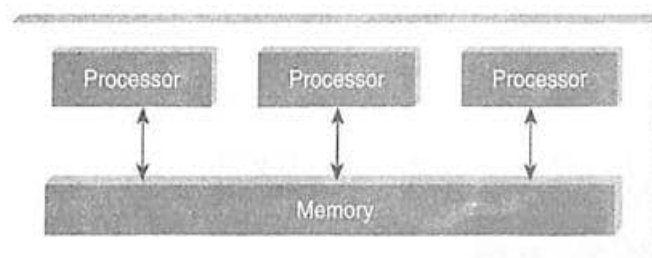


Figure 3.11 Parallel system.

3.12.6 Shared Nothing Architecture

There are three common types of multiprocessor architecture used in high transaction rate systems:

1. Shared Memory (SM)

- All processors share a common central memory.
- Suitable for systems needing tight coordination.
- Memory access is faster but can cause contention (conflicts).

2. Shared Disk (SD)

- Each processor has its own private memory.
- Disks are shared among multiple processors.
- Coordination required to manage access to shared disks.

3. Shared Nothing (SN)

- No sharing of memory or disk among processors.
- Each processor has its own private memory and disk.
- Ideal for scalability – processors operate independently.
- Most distributed systems (like Hadoop) follow this model.

Diagram – Figure 3.12: Distributed System

- Shows multiple processors (P1, P2, P3), each serving its own set of users.
- Each processor has its own storage.
- Processors communicate over a network.
- Represents a Shared Nothing Architecture, where processors **work independently and manage their own data.**

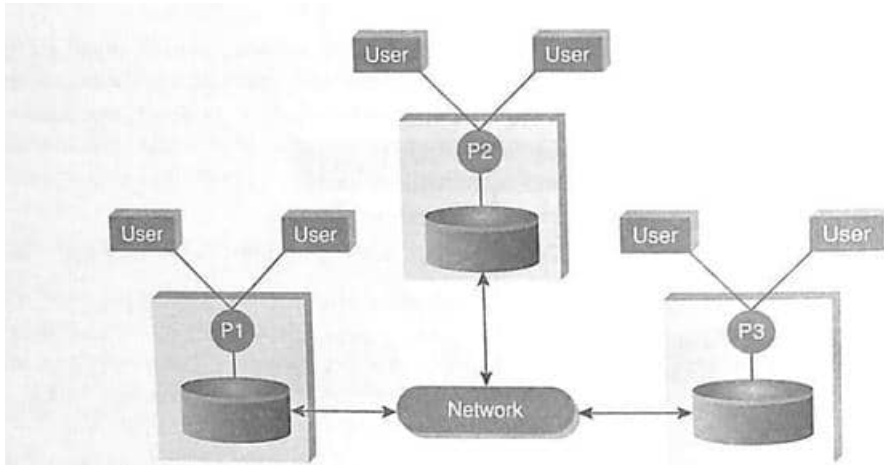


Figure 3.12 Distributed system.

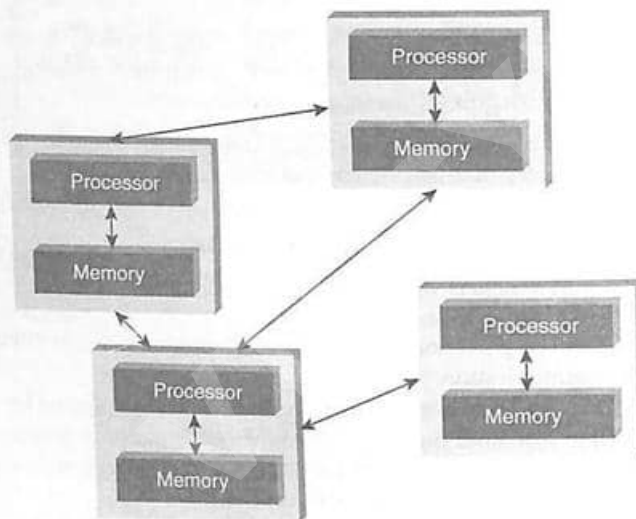


Figure 3.13 Distributed system.

3.12.6.1 Advantages of a “Shared Nothing Architecture”

□ Fault Isolation

- A "Shared Nothing Architecture" helps isolate faults effectively.
- If a fault occurs in a node, it is contained within that specific node.
- Other nodes remain unaffected since communication is done only through messages (or the absence of them).

□ Scalability

- In shared systems, disk bandwidth and controller become bottlenecks due to resource sharing.
- In a shared nothing setup, each node has its own resources, eliminating contention.
- No synchronization is needed for shared data access, so nodes can scale easily without performance degradation.

3.12.7 CAP Theorem

The CAP Theorem (also known as Brewer's Theorem) states:

In a distributed computing environment, it is impossible for a system to simultaneously guarantee all three of the following:

1. Consistency
2. Availability
3. Partition Tolerance

At best, only two of the three can be fully achieved.

3.12.7.1 CAP Theorem – Term Explanations

1. Consistency
 - Every read receives the most recent (last) write.
 - No stale or outdated data is returned.
2. Availability
 - Every request (read or write) receives a successful response (not necessarily the most recent).
 - The system is always available to serve requests.
3. Partition Tolerance
 - The system continues to function even if communication between nodes is lost (i.e., network failure).
 - No single point of failure crashes the entire system.

Diagram Summary (Figure 3.14)

A triangle with each corner representing one aspect of the CAP Theorem:

- Consistency
- Availability
- Partition Tolerance

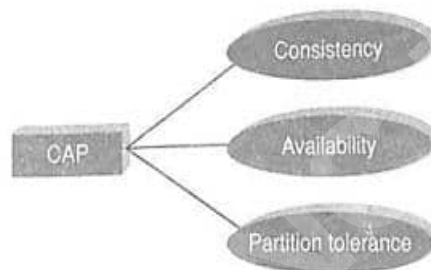


Figure 3.14 Brewer's CAP.

Training Institute Example

You visit Amey (Office Admin) to check your schedule:

1. You ask Amey if you're scheduled to train at 3:00 PM.
2. Amey checks his records and finds no such training scheduled.
3. You recall that the training coordinator informed yesterday about this session.
4. Amey suspects it could have been Joey (the second admin) who got the update.
5. Joey confirms: You are indeed scheduled for training at 3:00 PM.

Problem Identified:

⚠ "A clear case of inconsistent system!!!"

- The update was shared only with Joey, but you were checking with Amey.
- Both records were not synchronized, leading to data inconsistency.

Resolution Plan by Training Coordinator:

To ensure data consistency, the coordinator proposes:

1. Dual Update Rule:

- If either Amey or Joey gets an update, they must update their own file and inform the other to update theirs too.

2. Email Backup Rule:

- If the other admin is unavailable, the first one must send all updates via email.

3. Update on Resumption:

- When the unavailable admin returns to duty, he must update his file using the email updates.

Key Learnings:

- The inconsistency happened due to a lack of synchronization between two systems (admins).
- The solution ensures that availability may be slightly compromised (delays), but consistency is preserved.
- This directly reflects the CAP Theorem:
 - You cannot guarantee Consistency, Availability, and Partition Tolerance simultaneously.
 - This solution chooses to sacrifice some availability to preserve consistency.

Summary: CAP Theorem Trade-Offs

You can only choose two out of three:

1. Consistency (C):

- All users always get the latest, updated information.

2. Availability (A):

- The system always responds (e.g., admins always provide schedule info).

3. Partition Tolerance (P):

- System functions even if parts (admins/servers) can't communicate.

When to Choose What?

- Choose Availability over Consistency when:
 - Business can tolerate some delay in data sync.

- Choose Consistency over Availability when:
 - Business requires accurate, up-to-date reads/writes.

Examples of Database Systems Based on CAP:

Combination	Description	Example Databases
AP (Availability + Partition Tolerance)	System is always responsive and continues despite partitions, but may have inconsistent data.	Riak, Cassandra, CouchDB, DynamoDB
CP (Consistency + Partition Tolerance)	Strongly consistent but may become unavailable during network issues.	HBase, MongoDB, Redis, MemcacheDB, BigTable
CA (Consistency + Availability)	Consistent and always available, but can't tolerate partitions.	Traditional RDBMS, PostgreSQL, MySQL, etc.

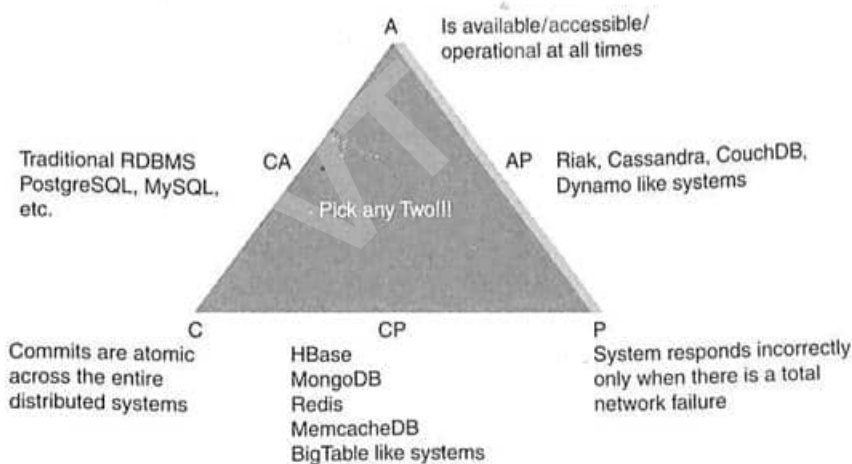


Figure 3.15 Databases and CAP.

Data Growth and Units (Table 2.2)

- **Data Size Units (Binary scale):**
 - Bit = 0 or 1
 - Byte = 8 bits
 - Kilobyte (KB) = 1024 bytes

-
- Megabyte (MB) = 1024^2 bytes
 - Gigabyte (GB) = 1024^3 bytes
 - Terabyte (TB) = 1024^4 bytes
 - Petabyte (PB) = 1024^5 bytes
 - Exabyte (EB) = 1024^6 bytes
 - Zettabyte (ZB) = 1024^7 bytes
 - Yottabyte (YB) = 1024^8 bytes

- **Data Size Units (Decimal scale as per Figure 2.6):**

- 1 KB = 1,000 bytes
- 1 MB = 1,000,000 bytes
- 1 GB = 1,000,000,000 bytes
- 1 TB = 1,000,000,000,000 bytes
- 1 PB = 1,000,000,000,000,000 bytes
- 1 EB = 1,000,000,000,000,000,000 bytes
- 1 ZB = 1,000,000,000,000,000,000,000 bytes
- 1 YB = 1,000,000,000,000,000,000,000,000 bytes

Note: Binary units are based on powers of 2 (1024), while decimal units are powers of 10 (1000).

Figure 3.15 – CAP Triangle

- Illustrates that you can pick only two of the three: C, A, or P.
- Each DB system aligns with one of the triangle's edges:
 - CA: Traditional RDBMS (e.g., PostgreSQL)
 - CP: Systems like HBase, Redis
 - AP: Systems like Cassandra, DynamoDB

4.1 NoSQL (NOT ONLY SQL)

- The term NoSQL was first coined by Carlo Strozzi in 1998 for his lightweight, open-source relational database that did not expose the standard SQL interface.
- In 2009, Johan Oskarsson reintroduced the term at an event focused on open-source distributed networks.
- The hashtag #NoSQL was coined by Eric Evans and others to describe non-relational databases.

Key Features of NoSQL Databases:

1. They are open source.
2. They are non-relational.
3. They are distributed.
4. They are schema-less.
5. They are cluster-friendly.
6. They emerged from 21st-century web applications.

4.1.1 Where is it Used?

- Widely used in big data and real-time web applications.
- Used to stock large volumes of data for analysis.
- Ideal for storing social media data and other data types that are difficult to store and analyze in traditional RDBMS.

4.1.2 What is it?

- NoSQL stands for Not Only SQL.
- These databases are:
 - Non-relational
 - Open source
 - Distributed
- Popular due to:
 - Ability to scale out or scale horizontally.
 - Aptitude for handling structured, semi-structured, and unstructured data.

Additional Feature of NoSQL Databases:

- Non-relational: They do not follow a relational data model.
- Types of NoSQL databases include:
 - Key-value pairs
 - Document-oriented
 - Column-oriented
 - Graph-based databases.

Characteristics of NoSQL Databases:

1. Are non-relational

- NoSQL databases do **not adhere to the relational data model**.
- They can be **key-value pairs, document-oriented, column-oriented, or graph-based** databases.

Figure 4.1: Where to use NoSQL?

- Log analysis
- Social networking feeds
- Time-based data (not easily analyzed in traditional RDBMS)

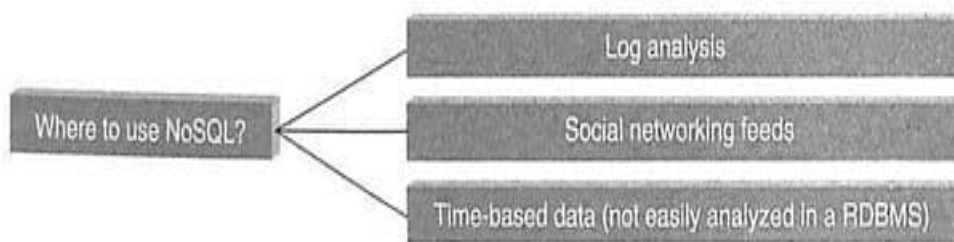


Figure 4.1 Where to use NoSQL?

Figure 4.2: What is NoSQL?

- Non-relational data storage systems
- No fixed table schema
- No joins

-
- No multi-document transactions
 - Relaxes one or more ACID properties

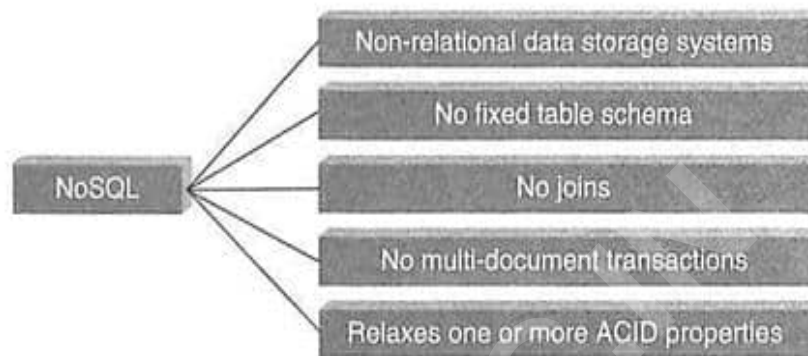


Figure 4.2 What is NoSQL?

Additional Features:

2. Are distributed:

- Data is distributed across several nodes in a cluster, which is often made up of low-cost commodity hardware.

3. Offer no support for ACID properties (Atomicity, Consistency, Isolation, Durability):

- They generally do **not** support traditional ACID transaction properties.
- Instead, they adhere to **Brewer's CAP theorem** (Consistency, Availability, and Partition tolerance), often **compromising on consistency** in favor of **availability and partition tolerance**.

4. Provide no fixed schema:

- NoSQL databases allow flexible schema design.
- They do **not require data to strictly follow any schema**, making them suitable for evolving or semi-structured data.

4.1.3 Types of NoSQL Databases

NoSQL databases are non-relational and broadly classified into two types:

1. Key-Value (the big hash table)

- Maintains a large hash table of keys and values.

- Examples: Dynamo, Redis, Riak, Amazon S3, Scalaris.
- **Sample Key-Value Pair:**

Key	Value
First Name	Simmonds
Last Name	David

2. Schema-less (includes Document, Column, and Graph-based)

- These databases do not have a fixed schema.
- Examples:

Document: MongoDB, Apache CouchDB, Couchbase, MarkLogic.

Store data as collections of documents.

Sample Document:

```
{
  "Book Name": "Fundamentals of Business Analytics",
  "Publisher": "Wiley India",
  "Year of Publication": "2011"
}
```

Column: Cassandra, HBase.

Each storage block contains data from only one column.

Graph-based: Neo4j.

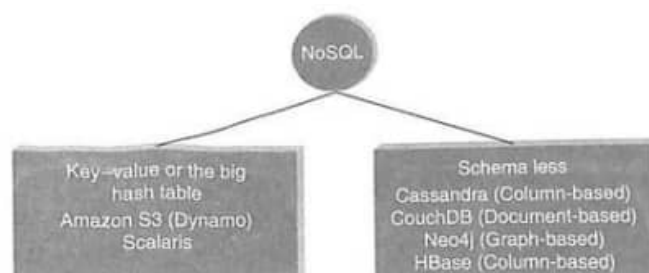


Figure 4.3 Types of NoSQL databases.

Graph Databases (also called network databases)

- Store data in **nodes**.
- Examples: Neo4j, HyperGraphDB.
- Sample graph structure example:
 - Nodes:
 - ID: 1001, Name: John, Age: 28
 - ID: 1002, Name: Joe, Age: 32
 - ID: 1003, Name: Group, Age: AAA
 - Relationships:
 - John knows Joe since 2002
 - John is a member of Group since 2003
 - Joe is a member of Group since 2002

4.1.4 Why NoSQL?

1. **Scale-out architecture:** Instead of monolithic relational databases.
2. **Can house large volumes** of structured, semi-structured, and unstructured data.
3. **Dynamic schema:** Allows insertion of data without a predefined schema. Supports faster development and easier code integration.
4. **Auto-sharding:** Automatically spreads data across multiple servers, balancing load and providing quick recovery if a server goes down.
5. **Replication:** Supports replication for high availability, fault tolerance, and disaster recovery.

4.1.5 Advantages of NoSQL

- **Can easily scale up and down:** Supports rapid, elastic scaling including scaling to the cloud.

Table 4.1: Popular Schema-less Databases

Key-Value Data Store	Column-Oriented Data Store	Document Data Store	Graph Data Store
Riak	Cassandra	MongoDB	Infinite Graph
Redis	HBase	CouchDB	Neo4j
Membase	Hyper Table	Raven DB	Allegro Graph

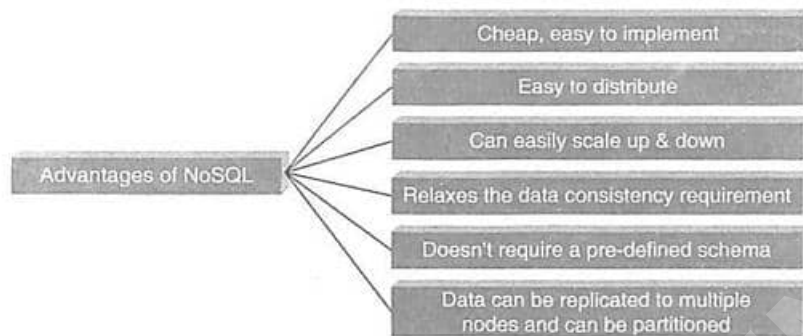


Figure 4.4 Advantages of NoSQL.

More Detailed Advantages of NoSQL

1. **Cluster scale:**

- Supports distribution across 100+ nodes (often in multiple data centers).

2. **Performance scale:**

- Handles over 100,000+ database reads/writes per second.

3. **Data scale:**

- Can store over 1 billion+ documents.

2. **Doesn't require a pre-defined schema**

- NoSQL (e.g., MongoDB) allows records with **different sets of key-value pairs**.
- Example (from MongoDB):

```
{
  "_id": 101,
  "BookName": "Fundamentals of Business Analytics",
  "AuthorName": "Seema Acharya",
  "Publisher": "Wiley India"
},
{
  "_id": 102,
  "BookName": "Big Data and Analytics"
}
```

3. Cheap, easy to implement

- Allows benefits of **scale, fault tolerance, high availability** at low cost.

4. Relaxes the data consistency requirement

- Adheres to the **CAP theorem** (Compromises consistency, but ensures availability and partition tolerance).
- Supports **eventual consistency**.

5. Data can be replicated and partitioned

(a) Sharding

- Distributes data automatically across multiple servers.
- Handles server addition/removal **without application downtime**.
- Balances data and query load.

(b) Replication

- Stores **multiple copies** of data across clusters/data centers.
- Ensures **high availability and fault tolerance**.

4.1.6 What We Miss With NoSQL?

- NoSQL solves **scalability** and **schema flexibility** issues.
- But **some RDBMS features are still superior** – details in next figure (Figure 4.5).

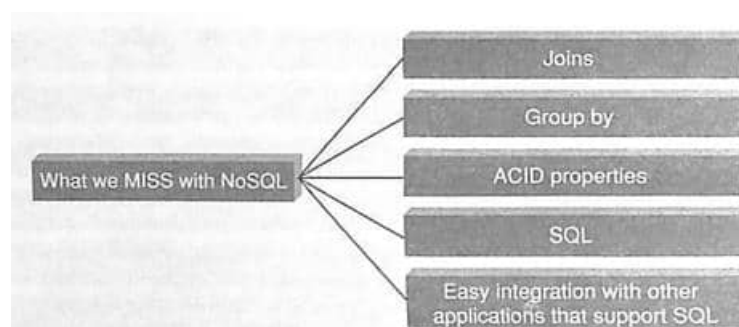


Figure 4.5 What we miss with NoSQL?

1. **Joins** – NoSQL databases generally **do not support JOIN operations** like SQL databases.
2. **Group By** – Advanced aggregations like GROUP BY can be complex or unsupported.
3. **ACID properties** – No built-in support for **Atomicity, Consistency, Isolation, Durability**.

-
4. **SQL** – No standard query language (although MongoDB uses its own query syntax, and Cassandra uses CQL).
 5. **Easy integration with other SQL-based applications** – Since it lacks standard SQL, it is **harder to integrate** with tools designed for SQL-based databases.

However, **MongoDB and Cassandra** mitigate this to some extent with:

- **MongoDB Query Language**
- **CQL (Cassandra Query Language)**

4.1.7 Use of NoSQL in Industry

NoSQL is used across various industries, particularly for big data and real-time applications.

Breakdown of how different NoSQL models are used:

1. **Key-Value Pairs**
 - Use: Shopping carts, user data analysis
 - Examples: Amazon, LinkedIn
2. **Column-Oriented**
 - Use: Analyze large user actions, sensor feeds
 - Examples: Facebook, Twitter, eBay, Netflix
3. **Document Based**
 - Use: Real-time analytics, logging, document management
 - Examples: MongoDB, CouchDB use cases
4. **Graph-Based**
 - Use: Network modeling, recommendations, upsell and cross-sell analytics
 - Examples: Walmart, social networks

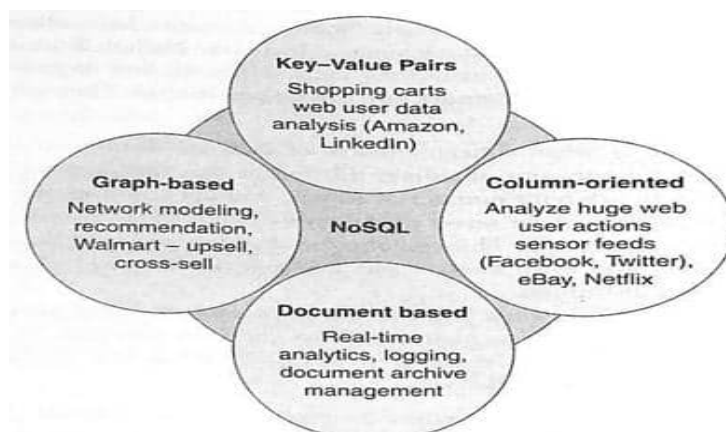


Figure 4.6 Use of NoSQL in industry.

4.1.8 NoSQL Vendors

A few popular NoSQL vendors and their products:

Company	Product	Most Widely Used by
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netflix, Twitter, eBay
Google	BigTable	Adobe Photoshop

4.1.9 SQL vs NoSQL

Table 4.3 summarizes the key differences:

SQL	NoSQL
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table-based databases	Document, graph, wide-column, or key-value store
Vertically scalable	Horizontally scalable (scale-out using clusters)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not ideal for large datasets	Preferred for large datasets
Not a good fit for hierarchical data	Ideal for hierarchical/JSON-like data
Emphasis on ACID properties	Follows CAP theorem (sacrifices ACID)
Excellent vendor support	Heavily community supported
Supports complex queries	Weak at complex querying

SQL	NoSQL
Can be configured for strong consistency	Some support eventual consistency (e.g., Cassandra)
Examples: Oracle, MySQL, PostgreSQL, etc.	Examples: MongoDB, Cassandra, HBase, Redis, Neo4j, CouchDB

4.1.10 NewSQL

- NewSQL is a modern RDBMS that combines features of both SQL and NoSQL.
- It offers the **scalability of NoSQL** systems used for **Online Transaction Processing (OLTP)**.
- At the same time, it maintains the **ACID guarantees** of traditional databases.
- NewSQL supports the **relational data model** and uses **SQL as the primary interface**.

4.1.10.1 Characteristics of NewSQL

- Based on the **shared-nothing architecture**.
- Uses a **SQL interface** for application interaction.

4.1.11 Comparison of SQL, NoSQL, and NewSQL

Feature	SQL	NoSQL	NewSQL
Adherence to ACID properties	Yes	No	Yes
OLTP/OLAP	Yes	No	Yes
Schema rigidity	Yes	No	Maybe
Adherence to data model	Adherence to relational model	No	Yes
Data Format Flexibility	No	Yes	Maybe
Scalability	Scale up (Vertical Scaling)	Scale out (Horizontal Scaling)	Scale out
Distributed Computing	Yes	Yes	Yes
Community Support	Huge	Growing	Slowly growing

4.2 HADOOP

- **Hadoop** is an open-source project by the **Apache Foundation**.
- It is a framework written in **Java**, originally developed by **Doug Cutting** in 2005.
- Named after Doug Cutting's son's **toy elephant**.
- Initially developed to support distribution for **Nutch** (a text search engine).
- Inspired by **Google MapReduce** and **Google File System**.
- Now a core part of computing infrastructure for companies like **Yahoo, Facebook, LinkedIn, Twitter**, etc.

4.2.1 Features of Hadoop

- Optimized to handle massive amounts of structured, semi-structured, and unstructured data using inexpensive, commodity hardware.
- Based on a shared-nothing architecture.
- Data replication across multiple computers ensures fault tolerance and availability.
- Designed for high throughput (not low latency). It's batch-oriented.
- Supports OLTP and OLAP, but not a replacement for traditional RDBMS.
- Not ideal when work cannot be parallelized or when data dependencies exist.
- Not suitable for small files—performs best with huge datasets.

4.2.2 Key Advantages of Hadoop

❑ **Stores data in its native format:**

- HDFS allows storing data without enforcing structure at input.
- Structure is applied only during processing.

❑ **Scalable:**

- Can store and distribute data across thousands of low-cost servers.
- Proven scalability (e.g., used by Facebook & Yahoo).

❑ **Cost-effective:**

- Reduced cost per terabyte due to scale-out architecture.

❑ **Resilient to failure:**

- Ensures data replication across nodes for fault tolerance.
- Automatically recovers from node failures.

□ **Flexible:**

- Handles all types of data (structured, semi-structured, unstructured).
- Supports various applications: log analysis, data mining, recommendation systems, market campaigns, etc.

□ **Fast:**

- High-speed data processing due to “move code to data” paradigm.

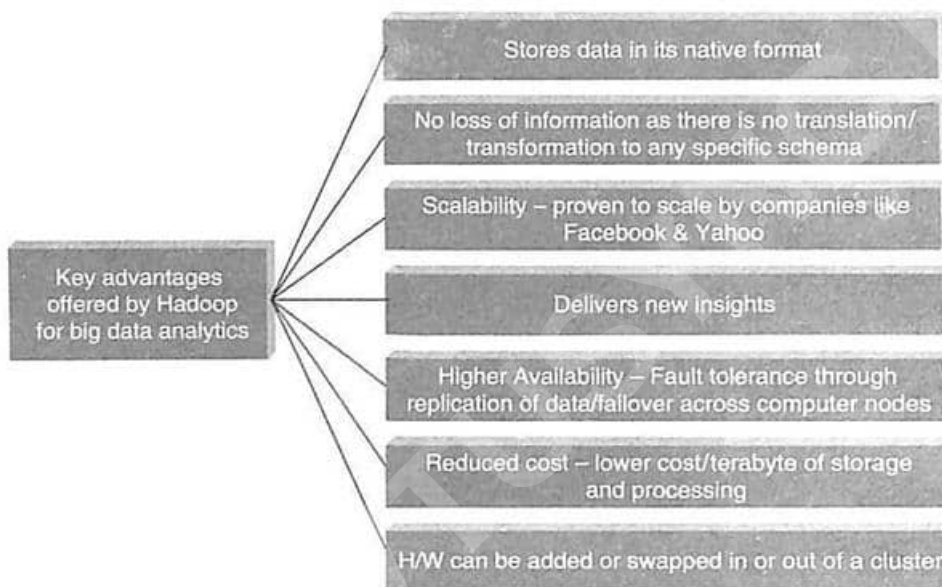


Figure 4.9 Key advantages of Hadoop.

4.2.3 Versions of Hadoop

There are two versions of Hadoop available:

1. **Hadoop 1.0**
2. **Hadoop 2.0**

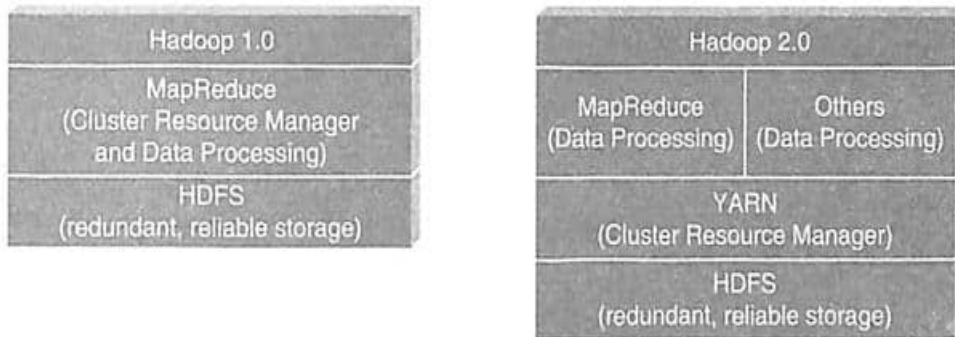


Figure 4.10 Versions of Hadoop.

- **4.2.3.1 Hadoop 1.0**

- It has two main parts:

1. **Data storage framework:**

- A general-purpose file system called **Hadoop Distributed File System (HDFS)**.
- HDFS is **schema-less**.
- It simply stores data files.
- These data files can be in just about any format.

2. **Data processing framework:**

Based on a simple functional programming model called **MapReduce** (popularized by Google).

- Uses two key functions:
 - **Map:** Takes in a set of key-value pairs and generates intermediate data (another list of key-value pairs).
 - **Reduce:** Acts on the intermediate data to produce the output data.
- Map and Reduce work in **isolation** from each other.
- Processing is **highly distributed, fault-tolerant**, and **scalable**.

Limitations of Hadoop 1.0

1. **Requirement for MapReduce expertise:**

- Developers needed to be proficient in MapReduce and programming languages like **Java**.

2. **Batch processing only:**

-
- Hadoop 1.0 supported only **batch processing**.
 - Suitable for:
 - Log analysis
 - Large-scale data mining
 - Not suitable for:
 - Other kinds of projects requiring real-time or interactive processing.

3. **Tight coupling with MapReduce:**

- Hadoop 1.0 was **tightly coupled** with MapReduce.
- Vendors had two poor options:
 - Rewrite their tools to work with MapReduce.
 - Extract data from HDFS and process it outside Hadoop.
- Both options led to inefficiencies due to data movement in and out of the Hadoop cluster.
-

4.2.3.2 Hadoop 2.0

- HDFS remains the data storage framework in Hadoop 2.0.
- Introduced a new resource management framework called:
 - YARN (Yet Another Resource Negotiator):
 - Supports dividing applications into parallel tasks.
 - Enhances flexibility, scalability, and efficiency.
 - Replaces the old JobTracker with ApplicationMaster.
 - Replaces TaskTracker with NodeManager.
 - Allows any application (not just MapReduce) to run on Hadoop.
- **Key Benefits:**
 - MapReduce expertise is no longer required.
 - Supports both:
 - Batch processing
 - Real-time processing
 - MapReduce is no longer the only option:
 - Alternative data processing tools are now supported.
 - Native features like data standardization and master data management can be performed in HDFS.

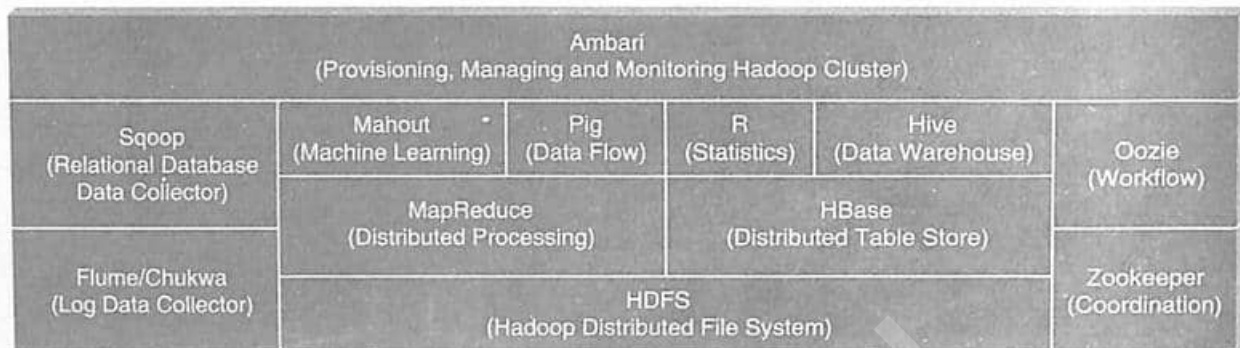


Figure 4.11 Hadoop ecosystem.

HDFS (Hadoop Distributed File System)

- It is the **distributed storage unit** of Hadoop.
- Provides **streaming access** to file system data.
- Supports **file permissions** and **authentication**.
- Based on **GFS (Google File System)**.
- Scales a single cluster node to **hundreds or thousands of nodes**.
- Handles **large datasets** on **commodity hardware**.
- HDFS is **highly fault-tolerant**:
 - Stores files across multiple machines.
 - Files are stored in a **redundant fashion** to allow **data recovery in case of failure**.

(Example Scenario)

- An **e-commerce website** stores **millions of customers' data** in a distributed manner.
- Data has been collected over **4–5 years**.
- Batch analytics is run on **archived data** to analyze:
 - Customer **behavior**
 - **Buying patterns**
 - **Preferences**

-
- **Requirements**
 - Helps identify:
 - Which **products** are purchased
 - In which **months**
 - By which **types of customers**

HBase

- **Stores data in HDFS.**
- It is the **first non-batch component** of the Hadoop ecosystem.
- Works as a **database on top of HDFS.**
- Offers **quick random access** to stored data.
- Has **very low latency** compared to HDFS.
- It is a:
 - **NoSQL database**
 - **Non-relational**
 - **Column-oriented** database
- **Data Structure:**
 - A **table** can have **thousands of columns.**
 - A **row** can have several **column families.**
 - Each **column family** can contain **multiple columns.**
 - Each **column** can have several **key-value pairs.**
- **Based on: Google BigTable**
- **Used by:** Facebook, Twitter, Yahoo, etc.

(Example Scenario)

- The same **e-commerce website** also stores **millions of product data.**
- To **search** among millions of products and get **real-time results**, optimization is needed.
- **HBase supports real-time analytics.**
- Due to high data velocity, they **chose HBase over HDFS**, as HDFS does **not support real-time writes.**
- **Impact:**
 - **Query time reduced** from **3 days to 3 minutes.**

Difference Between HBase and Hadoop/HDFS

Aspect	HDFS (Hadoop Distributed File System)	HBase (Hadoop Database)
Type	File System	NoSQL Database
Analogy	Like NTFS	Like MySQL
Write/Read Model	WORM (Write Once Read Many)	Real-time random read and write
Underlying System	Based on Google File System (GFS)	Based on Google BigTable
Data Access Pattern	Full table scan or partition scan	Random access to small ranges
Performance with Hive	Very good	4–5 times slower than HDFS
Access Methods	Via MapReduce jobs only	Via Java APIs, REST, Avro, Thrift
Storage Type	Static and rigid	Dynamic and flexible
Latency	High latency	Low latency
Best Suited For	Batch processing and analytics	Real-time analytics

Hadoop Ecosystem Components for Data Processing

1. MapReduce

- A **programming model** for processing large datasets in parallel.
- Developed by **Google** in 2004.
- Works in **two phases**:
 - **Map**: Converts input data into key-value pairs.
 - **Reduce**: Combines and processes these pairs to generate output.

-
- Data flows from and back to **HDFS**.

2. Spark

- A **programming and computing model** developed at **UC Berkeley** in 2009.
- Open-source and written in **Scala**.
- Performs **in-memory processing** (much faster than disk-based).
- Can use **HDFS** as a data source but **does not require MapReduce**.
- Can run with or without Hadoop.

Spark Libraries:

- **Spark SQL** – Supports querying using SQL.
- **Spark Streaming** – For real-time data analysis.
- **MLlib** – For machine learning tasks.
- **GraphX** – For graph computations.

Key Notes

- **Hadoop** is used for **batch processing** of unstructured data.
- **Spark** is widely used for **fast, real-time, in-memory processing**.

Hadoop Ecosystem Components for Data Analysis

1. Pig

- A **high-level scripting language** used with Hadoop.
- Acts as an **alternative to MapReduce**.
- Developed by **Yahoo**.

Pig has two main parts:

- **Pig Latin:**
 - A scripting language that gets translated into **MapReduce jobs**.
 - Used for **ETL (Extract, Transform, Load)** and data analysis.

- Supports operations like **grouping, filtering, sorting, joining**, etc.
- Loads data from HDFS, processes it, and sends it back or displays it.

- **Pig runtime:**

- The **execution environment** where Pig scripts are run.

2. Hive

- A **data warehouse software** on top of Hadoop.
- Used for **summarization, querying, and analysis**.
- Uses **HQL (Hive Query Language)** or **HiveQL**, similar to SQL.
- Converts SQL-like queries into **MapReduce jobs** for execution on Hadoop.

Difference Between Hive and RDBMS

Aspect	Hive	RDBMS (MySQL, SQL Server, etc.)
Schema Enforcement	Enforced on read (schema-on-read)	Enforced on write (schema-on-write)
Data Load	Fast, as schema is not enforced during insert	Slower due to schema enforcement during load
Query Performance	Better read performance after faster load	Good performance after schema validation
Data Access Model	Write once, read many times	Read and write many times
Processing Type	Batch-oriented , not ideal for OLTP	Designed for OLTP and day-to-day transactions
Best Suited For	Analytical workloads (OLAP-like use cases)	Transactional workloads (OLTP)
Query Execution	Queries are converted into MapReduce jobs	Queries are executed directly in the database engine

4.2.5 Hadoop Distributions

- **Hadoop** is an **open-source Apache project**.
- Core components of Hadoop:
 1. **Hadoop Common**
 2. **HDFS (Hadoop Distributed File System)**
 3. **YARN (Yet Another Resource Negotiator)**
 4. **MapReduce**
- Several companies have built **custom distributions** of Hadoop for easier use:
 - **IBM**
 - **Amazon Web Services**
 - **Microsoft**
 - **Teradata**
 - **Hortonworks**
 - **Cloudera**
- Despite different strategies, all distributions aim to:
 - Distribute data and workloads across many servers.
 - Make big data **manageable** and **scalable**.

4.2.5 Hadoop Distributions

- **Hadoop** is an open-source project from Apache, freely available for download.
- Core components of Hadoop include:
 1. **Hadoop Common**
 2. **Hadoop Distributed File System (HDFS)**
 3. **Hadoop YARN** (Yet Another Resource Negotiator)
 4. **Hadoop MapReduce**

- Various companies have created their own **distributions** (packaged versions) of Hadoop to make it more user-friendly and consumable. These include: **IBM, Amazon, Web Services, Microsoft, Teradata, Hortonworks, Cloudera.**
- Though their strategies differ slightly, the main goal is to distribute data and workloads across many servers for scalable and manageable big data processing.

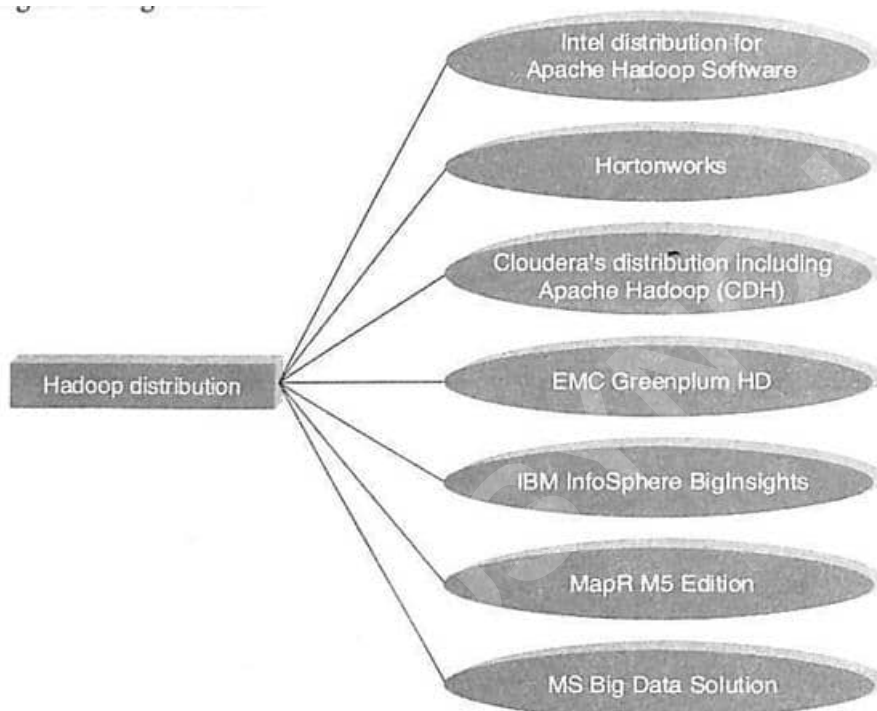


Figure 4.12 Hadoop distributions.

4.2.8 Hadoop versus SQL

Hadoop	SQL
Scale out	Scale up
Key-Value pairs	Relational table
Functional Programming	Declarative Queries
Offline batch processing	Online transaction processing

-
- Scale out (Hadoop) means adding more machines to handle increased workload, while scale up (SQL) means increasing the power of a single machine.
 - Hadoop uses a key-value pair data model, whereas SQL uses relational tables.
 - Hadoop programming is functional in nature, while SQL uses declarative queries.
 - Hadoop is optimized for offline batch processing of large datasets, while SQL is designed for online transaction processing (OLTP) with quick response times.

4.2.7 Integrated Hadoop Systems Offered by Leading Market Vendors

These vendors package and offer Hadoop-based big data platforms that are ready for enterprise use.

The major vendors mentioned are:

- **EMC Greenplum**
- **Oracle Big Data Appliance**
- **Microsoft Big Data Solution**
- **IBM InfoSphere**
- **HP Big Data Solutions**
-

These integrated systems typically combine hardware, software, and support services optimized for Hadoop workloads, helping organizations deploy big data solutions more easily and efficiently.

4.2.8 Cloud-Based Hadoop Solutions

Amazon Web Services (AWS): Provides a comprehensive portfolio of cloud services aimed at managing big data efficiently. AWS emphasizes reducing costs, scaling on demand, and accelerating innovation speed.

Google Cloud Storage connector for Hadoop: Allows direct execution of MapReduce jobs on data stored in Google Cloud Storage without copying it locally. This simplifies Hadoop deployment, reduces costs, and offers performance comparable to Hadoop's HDFS. It also enhances reliability by removing the single point of failure (name node).

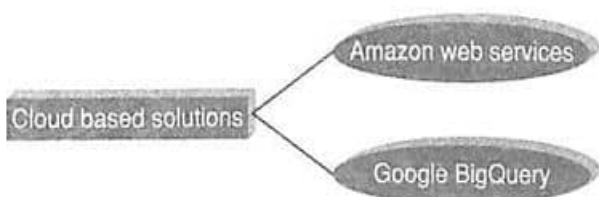


Figure 4.14 Cloud-based solutions.
