15CSE334 -Big Data Analytics

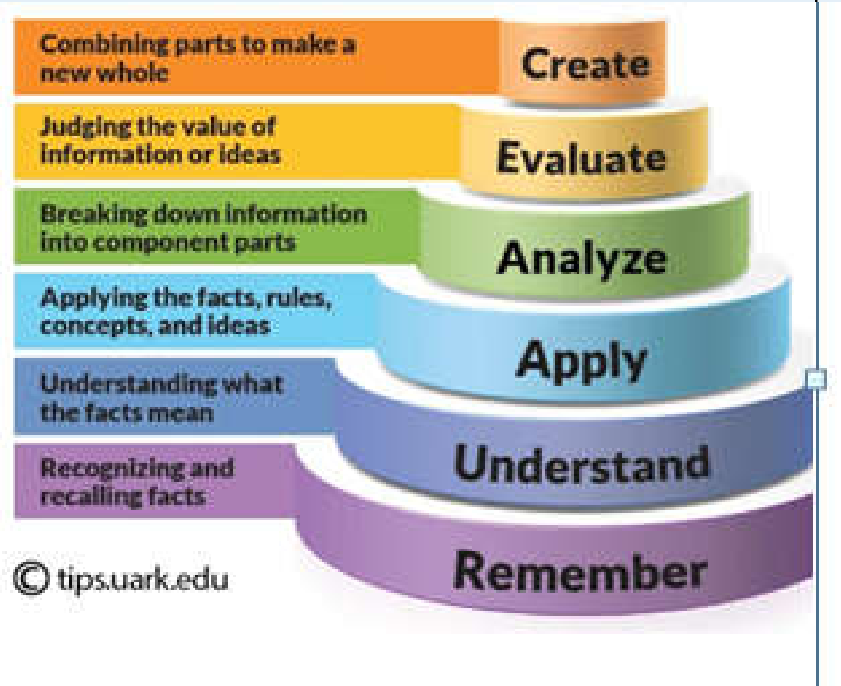
Dr. R. Karthi

Course Outline

- Understand the basics of Big data.
- Understand Hadoop architecture and tools used for big data.
- Learn to implement map-reduce programs for big data.
- Understand and apply Big data tools for various applications.
- Course – Lab Based Course (3 Credits)
  - 3 hrs. theory + 3 hrs. lab per week
  - Laptop with internet connectivity required for Theory and Lab hrs.
  - Enthusiasm to explore and learn new skills.
  - Theory Exams will be in paper based mode.
  - Lab slots are used to demonstrate and have an hand on experience on each platform/tool.

Course Outcomes

	Course Outcome	Bloom's Taxonomy Level
CO 1	**Understand** fundamental concepts of Big Data and its technologies	L2
CO 2	**Apply** concepts of Map Reduce framework for optimization.	L3
CO 3	**Analyze** appropriate No SQL database techniques for storing and processing large volumes of structured and unstructured data.	L4
CO 4	**Apply** data analytics solutions using Hadoop ecosystems	L3
CO 5	**Explore** tools for machine learning and reporting	L4



Evaluation Pattern

Evaluation Component	Weightage
Periodical 1	15
Periodical 2	15
Continuous assessment (Quiz/Tutorial/Assignment)	20
End Semester	50

Course Mapping

- Introduction to Big Data
- Introduction to Hadoop
  - Architecture of hadoop
  - Map Reduce programming
- Hadoop ecosystem –tools
  - Pig
  - Hive
- NOSQL Databases
  - MongoDB
  - Cassandra
- Machine Learning

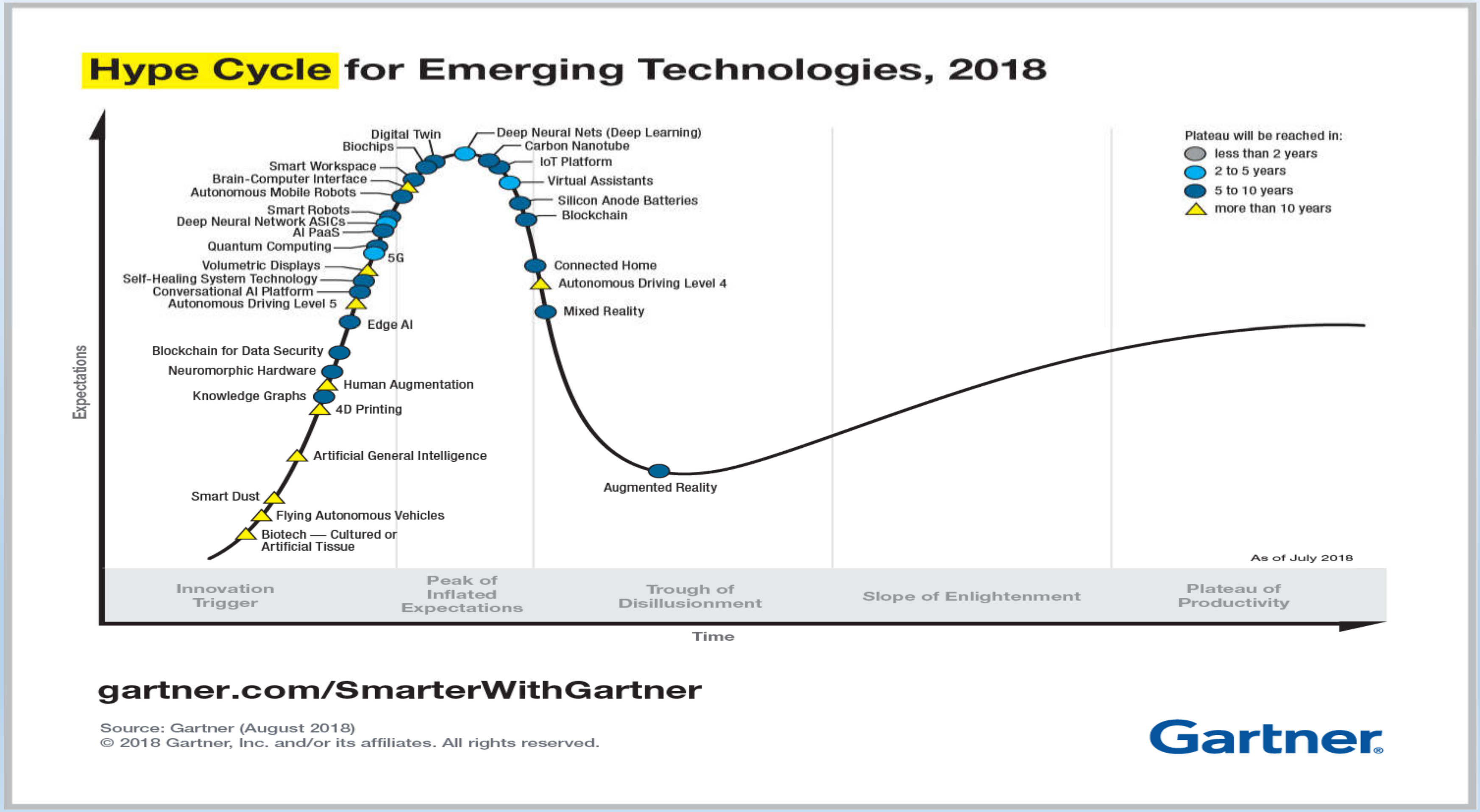
Points to Remember

- Maintain a Observation notebook for lab.
- All exercises and evaluations done in lab and class are to be recorded.
- Keep your laptops updated with sufficient processing speed and memory.
- Amrita AUMS / Email accounts to be active.
- Case study on big data for real world applications.

Introduction to Big Data Analytics

Dr. R. Karthi

Emerge in the Gartner Hype Cycle for Emerging Technologies, 2018



Emerge in the Gartner Hype Cycle for Emerging Technologies, 2019

Big Data – Industry Needs

A 2011 study by the McKinsey Global Institute predicts that by 2018-19the **U.S. alone will face a shortage of 140,000 to 190,000 people with deep analytical skills** as well as ***1.5 million managers and analysts*** to analyze big data and make decisions.

What Is Big Data?

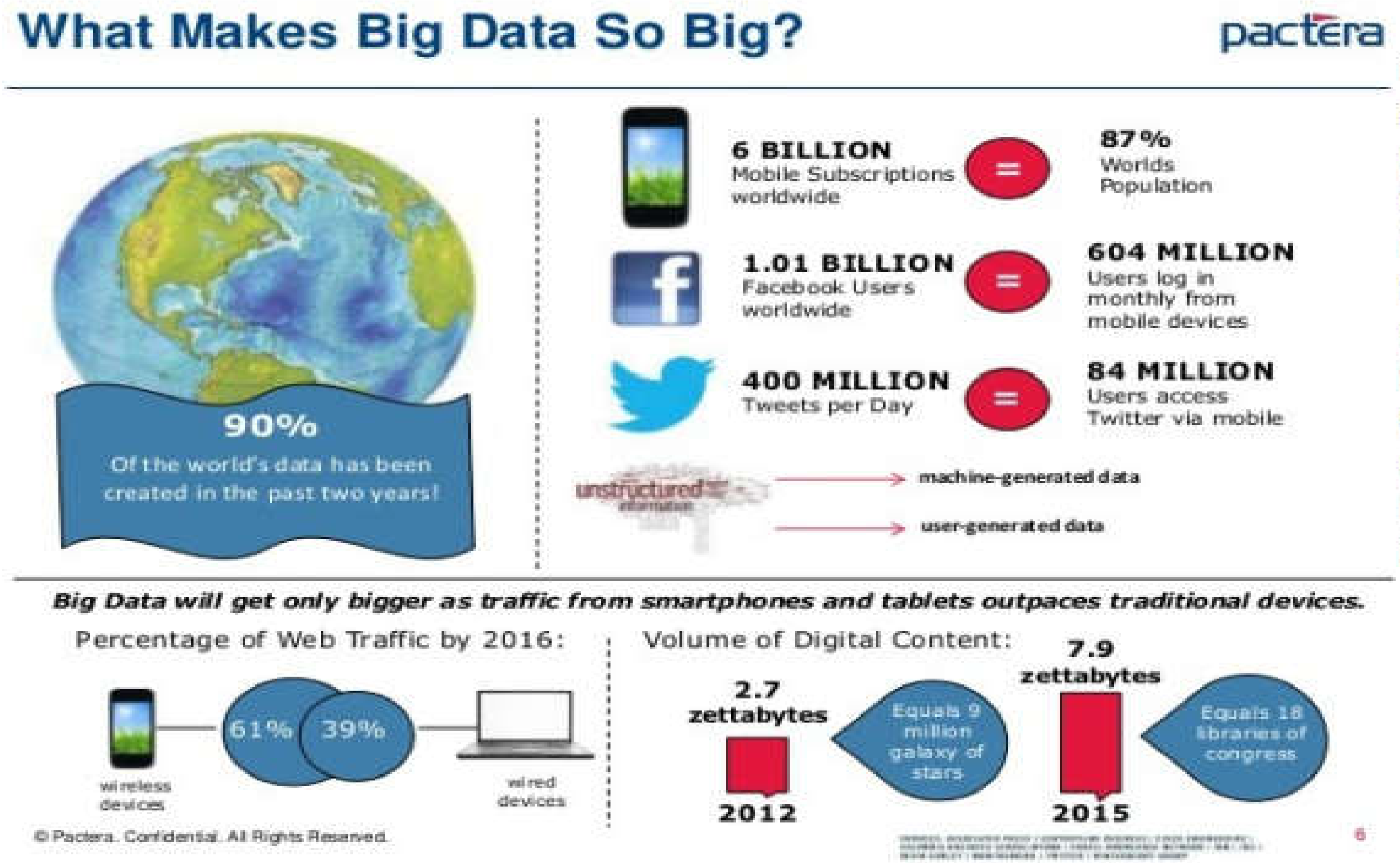
- **No single standard definition…**

- *Extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.*

- *Big data is an evolving term that describes any voluminous amount of structured, semi-structured and unstructured data that has the potential to be mined for information.*

- *"**Big Data**" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…*

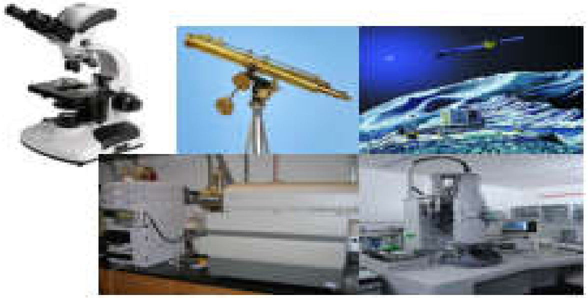
What makes Data "Big"



1 Bit = Binary Digit
8 Bits = 1 Byte
1024 Bytes = 1 Kilobyte
1024 Kilobytes = 1 Megabyte
1024 Megabytes = 1 Gigabyte
1024 Gigabytes = 1 Terabyte
1024 Terabytes = 1 Petabyte
1024 Petabytes = 1 Exabyte
1024 Exabytes = 1 Zettabyte
1024 Zettabytes = 1 Yottabyte
1024 Yottabytes = 1 Brontobyte
1024 Brontobytes = 1 Geopbyte

Who's Generating Big Data

Social media and networks
(all of us are generating data)



Scientific instruments
(collecting all sorts of data)

Mobile devices
(tracking all objects all the time)

Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data

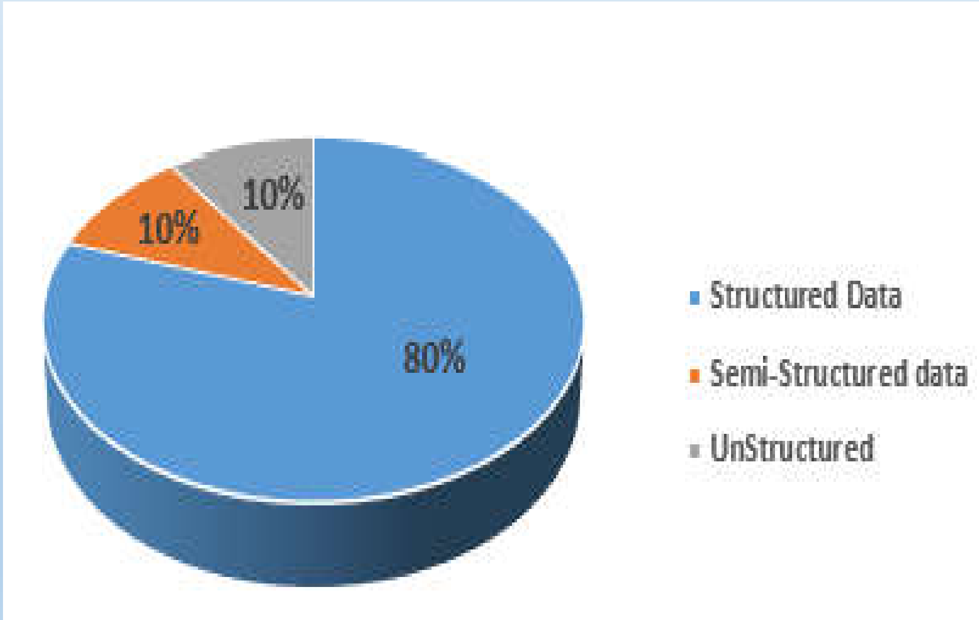
New Model: all of us are generating data, and all of us are consuming data

Classification of Digital Data

Digital data is classified into the following categories:

- Structured data
- Semi-structured data
- Unstructured data

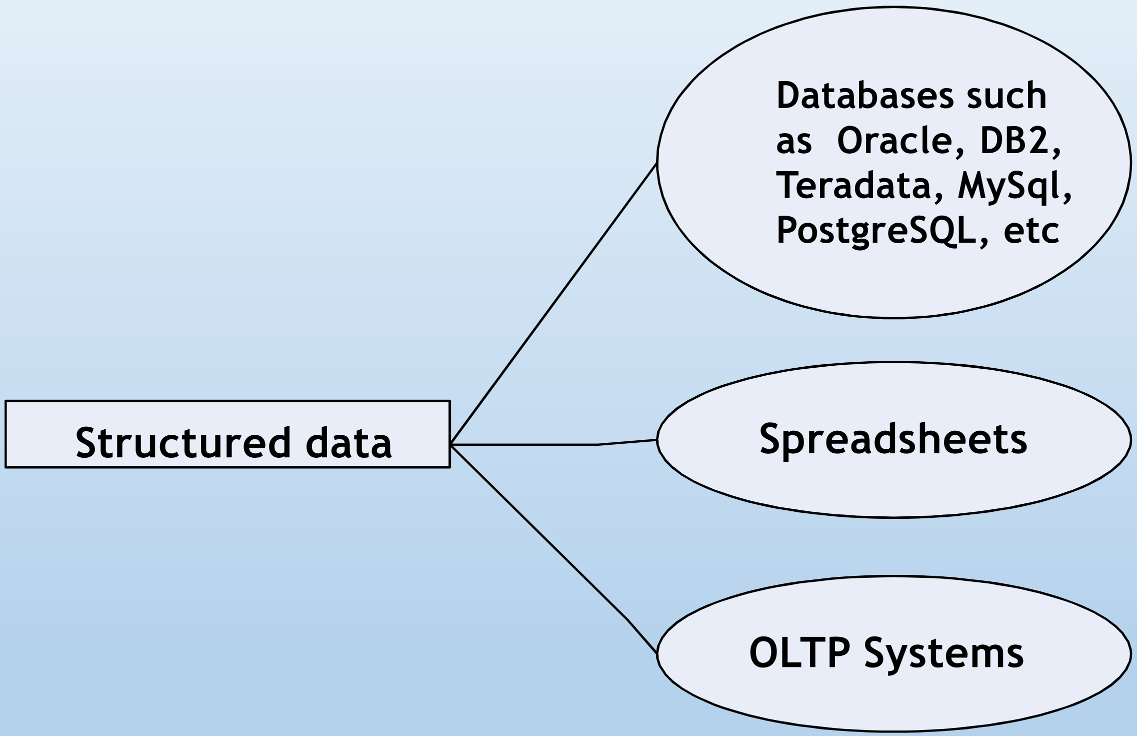
Approximate percentage distribution of digital data



Structured Data

- This is the data which is in an organized form (e.g., in rows and columns) and can be easily used by a computer program - Relational data model
- Specific Data type and Constraints(Unique, Not Null) are defined
- Relationships exist between entities of data, such as classes and their objects.
- Data stored in databases is an example of structured data.
- Example : Employee Data base

Sources of Structured Data



Online Transaction Processing Systems

Ease with Structured Data

Ease with Structured data

- **Input / Update / Delete** — DML operations
- **Security** — Access control (Tokens), Encryption
- **Indexing / Searching** — Speedup select operations with additional write and storage space
- **Scalability** — Scale up by increasing the horse power (Additional memory and processing capacity)
- **Transaction Processing** — ACID properties

Semi-structured Data

- This is the data which does not conform to a data model but has some structure. However, it is not in a form which can be used easily by a computer program.
- It uses tags to separate semantic elements and to enforce hierarchies of records and fields within data.
- No separation between schema and data.
- Metadata for this data is available but is not sufficient.
- Example: emails, XML, markup languages like HTML, etc.

Characteristics of Semi-structured Data

Semi-structured data

- Inconsistent Structure
- Self-describing
  (Label /value pairs)
- Often Schema information is
  blended with data values
- Data objects may have different attributes not known beforehand

Unstructured Data

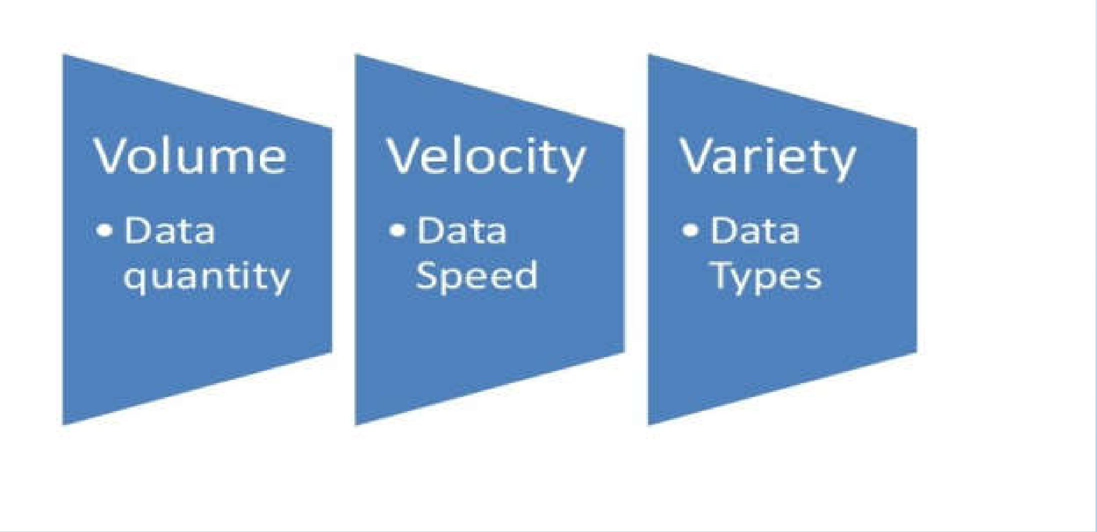
- Data which does not conform to a data model or is not in a form which can be used easily by a computer program.
- About 80-90% data of an organization is in this format.
- Example: memos, chats , PowerPoint presentations, images, videos, letters, researches, white papers, body of an email, etc.

Unstructured data

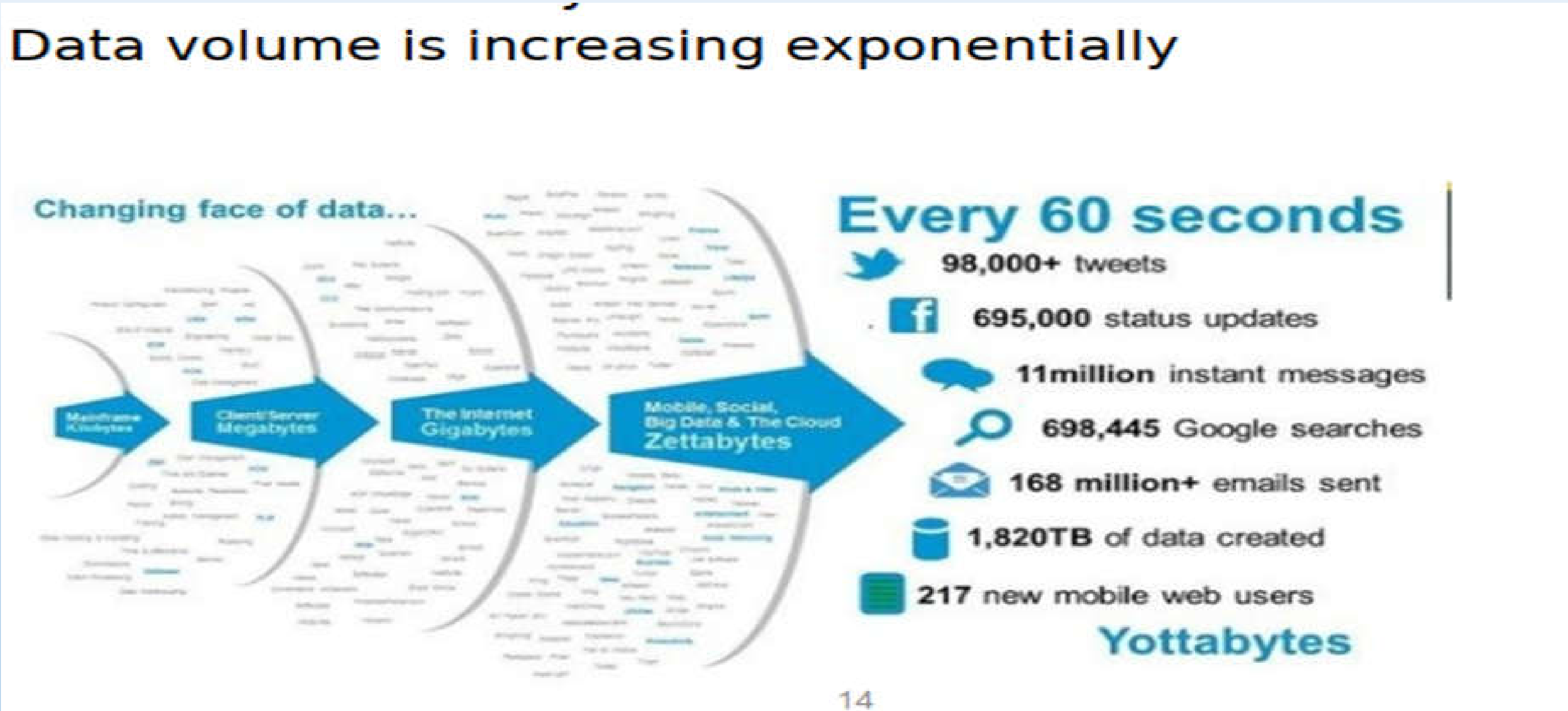
Structure needs to be implied despite not being formerly defined.

Data with some structure may still be label as unstructured if the structure doesn't help with processing

Characteristics of Big Data



Data Volume


Data volume is increasing exponentially
Changing face of data...
Mainframe Kilobytes
Client/Server Megabytes
The Internet Gigabytes
Mobile, Social, Big Data & The Cloud Zettabytes
Every 60 seconds
98,000+ tweets
695,000 status updates
11million instant messages
698,445 Google searches
168 million+ emails sent
1,820TB of data created
217 new mobile web users
Yottabytes
14

Volume

1 Kilobyte (KB) = 1000 bytes

1 Megabyte (MB) = 1,000,000 bytes

1 Gigabyte (GB) = 1,000,000,000 bytes

1 Terabyte (TB) = 1,000,000,000,000 bytes

1 Petabyte (PB) = 1,000,000,000,000,000 bytes

1 Exabyte (EB) = 1,000,000,000,000,000,000 bytes

1 Zettabyte (ZB) = 1,000,000,000,000,000,000,000 bytes

1 Yottabyte (YB) = 1,000,000,000,000,000,000,000,000 bytes

Velocity

Batch → Periodic → Near real time → Real-time processing

Characteristics of Big Data: Velocity

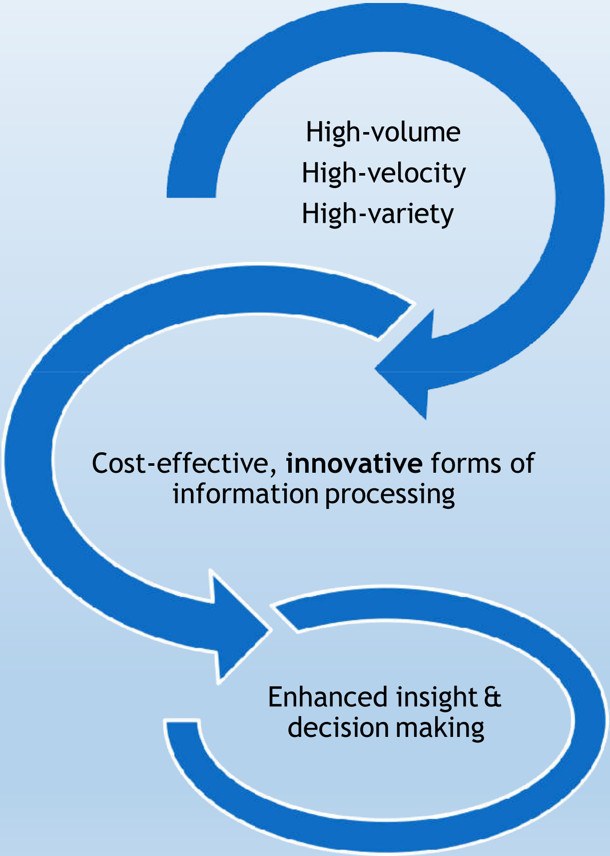
- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities

- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

Variety

- **Structured data:** example: traditional transaction processing systems and RDBMS, etc.
- **Semi-structured data:** example: Hyper Text Markup Language (HTML), eXtensible Markup Language (XML).
- **Unstructured data:** example: unstructured text documents, audio, video, email, photos, PDFs, social media, etc.

Definition of Big Data



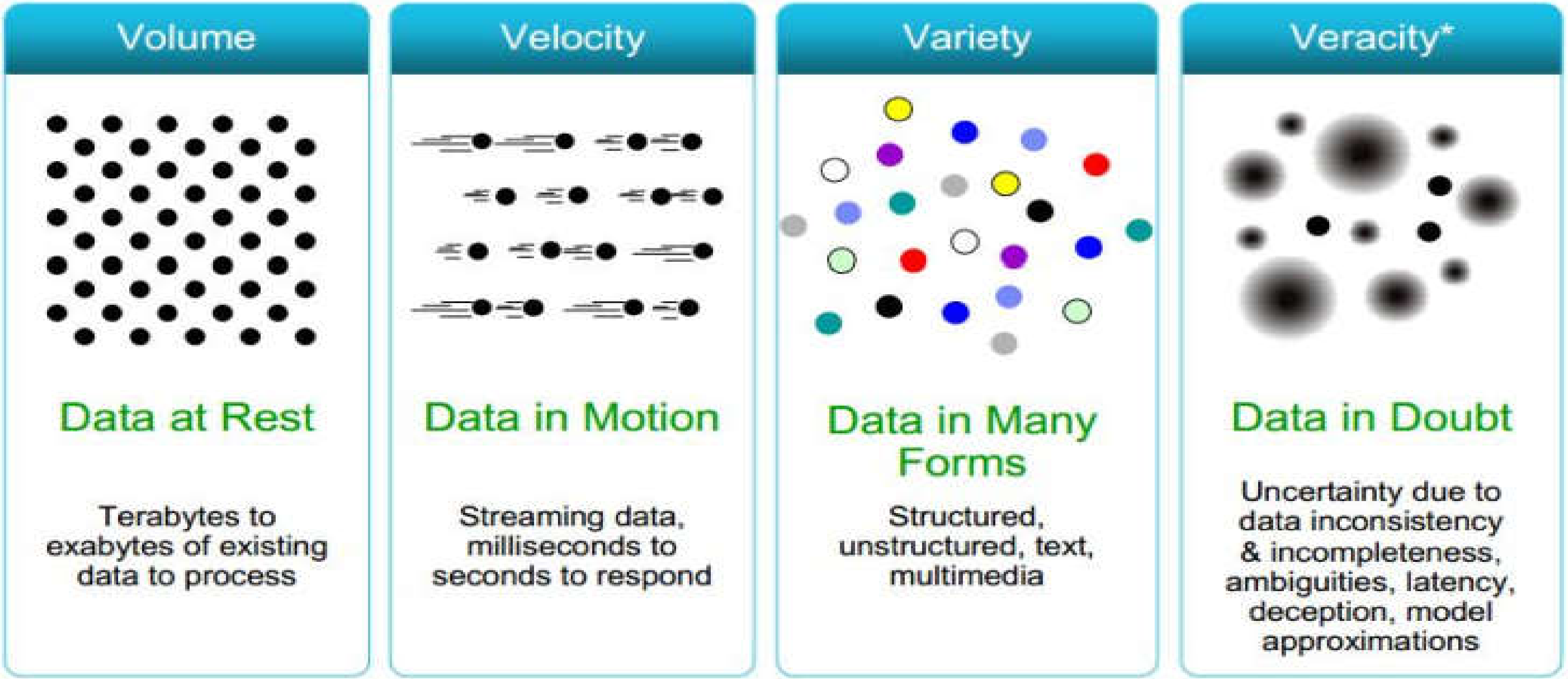
Big Data is high-volume, high-velocity, and high-variety information assets that demand cost effective, innovative forms of information processing for enhanced insight and decision making.

Source: Gartner IT Glossary

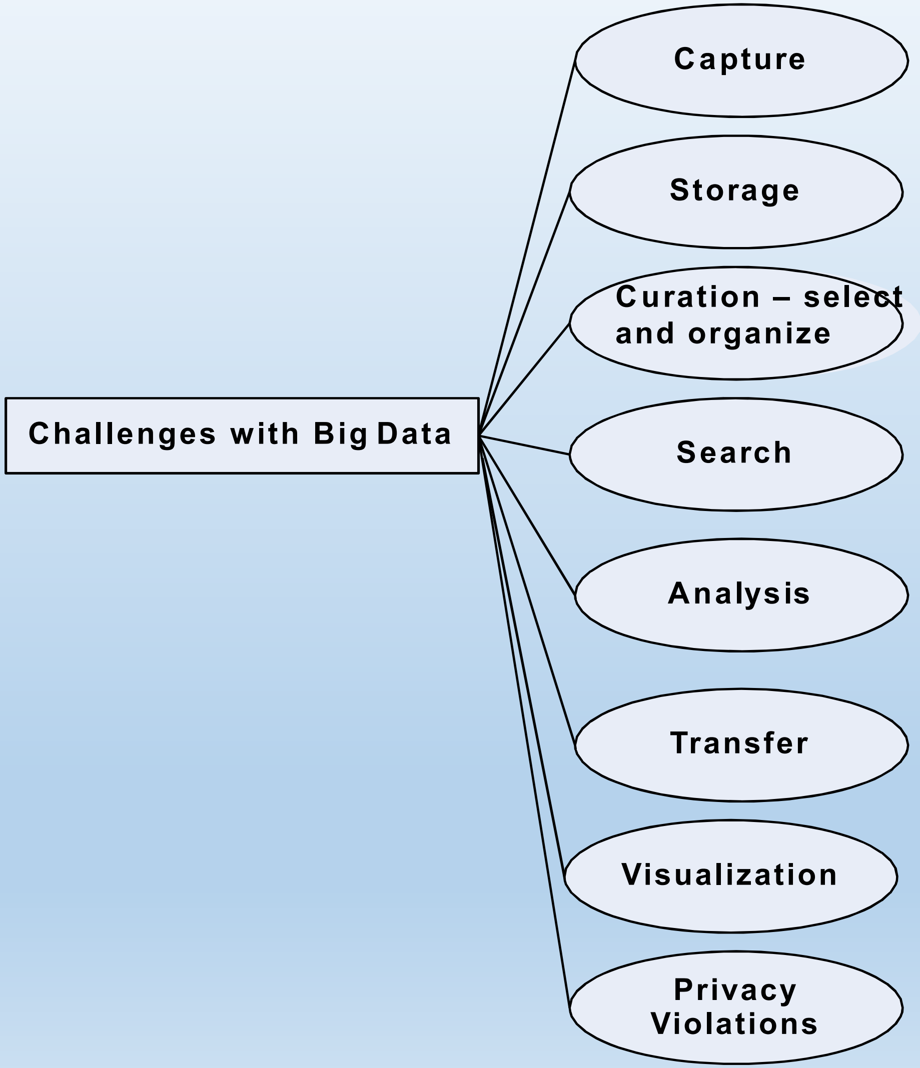
Other Characteristics of Data - Traits of Big Data

- Veracity - the quality of being true or accurate.
- Volatility / Validity - how long is the data valid and how long should it be stored.
- Variability – Data whose meaning is changing , inconsistencies in the data.

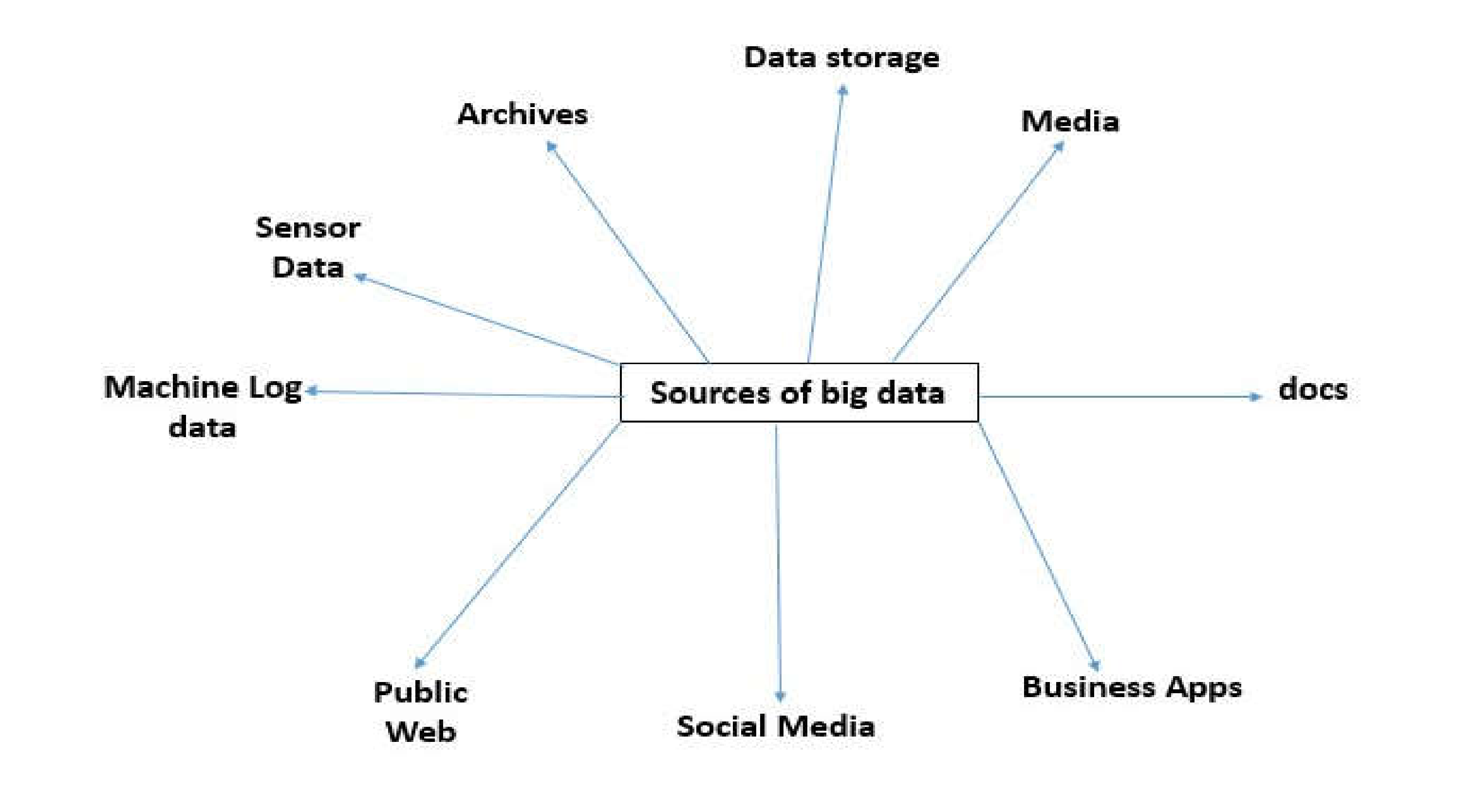
Some Make it 4V's



Challenges with Big Data



Sources of Big Data



Classification of analytics

Two thoughts:
1. Classify analytics into basic, operational, advanced and monetized.
2. Classify analytics into analytics1.0, 2.0 and 3.0

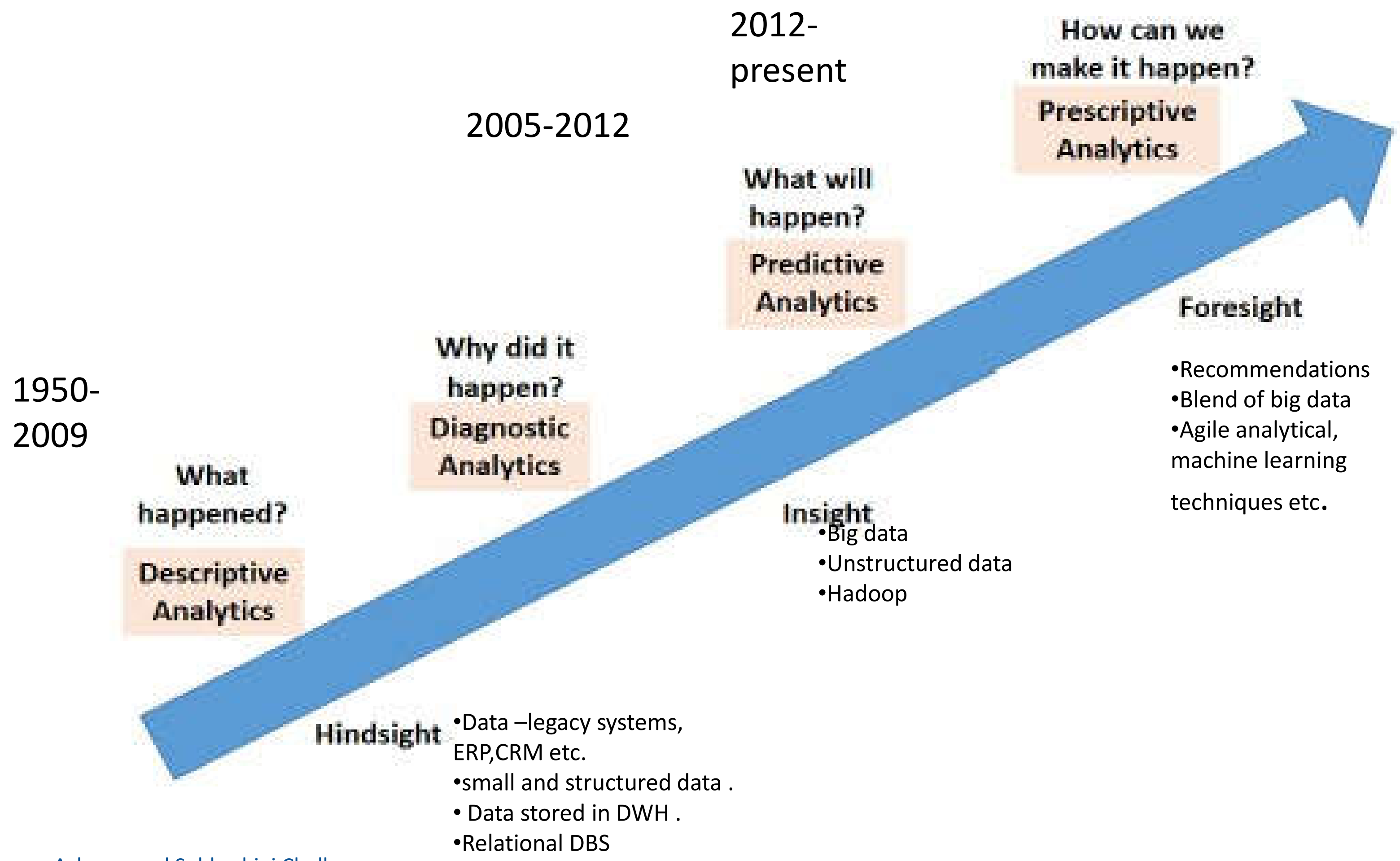
Basic analytics:
- Slicing and dicing data to help with basic business insights
- Based on historical data , visualization etc.

Operational Analytics: Enterprise business analysis (OLTP data)

Advanced Prediction analytics: Forecasting for the future/Prediction

Monetized Analytics: Analysis of data to derive direct business revenue.

Analytics 1.0, 2.0 and 3.0



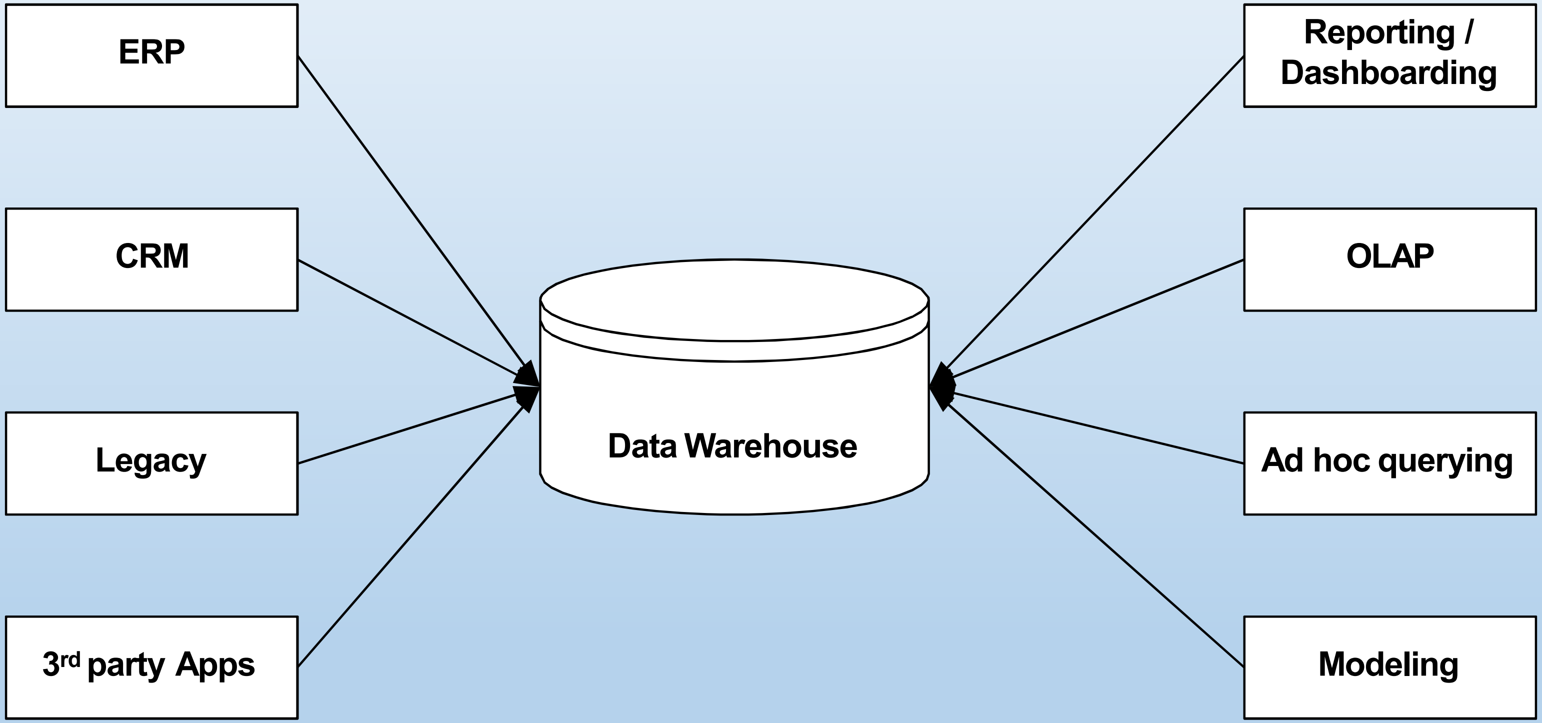
Types of Analytics

- Descriptive analytics:
  - What happened / why it has occurred.
  - Reporting tools, OLAP, dash boards, data visualization.
- Predictive analytics :
  - Model forecast an outcome at some future state or time based upon changes to the model inputs.
  - Algorithms are regression analysis, time series regression, machine learning methods – CART and neural networks.
- Prescriptive analytics:
  - Uses predictive models to suggest actions to take for optimal outcomes.
  - Prescriptive analytics relies on optimization and rules-based techniques

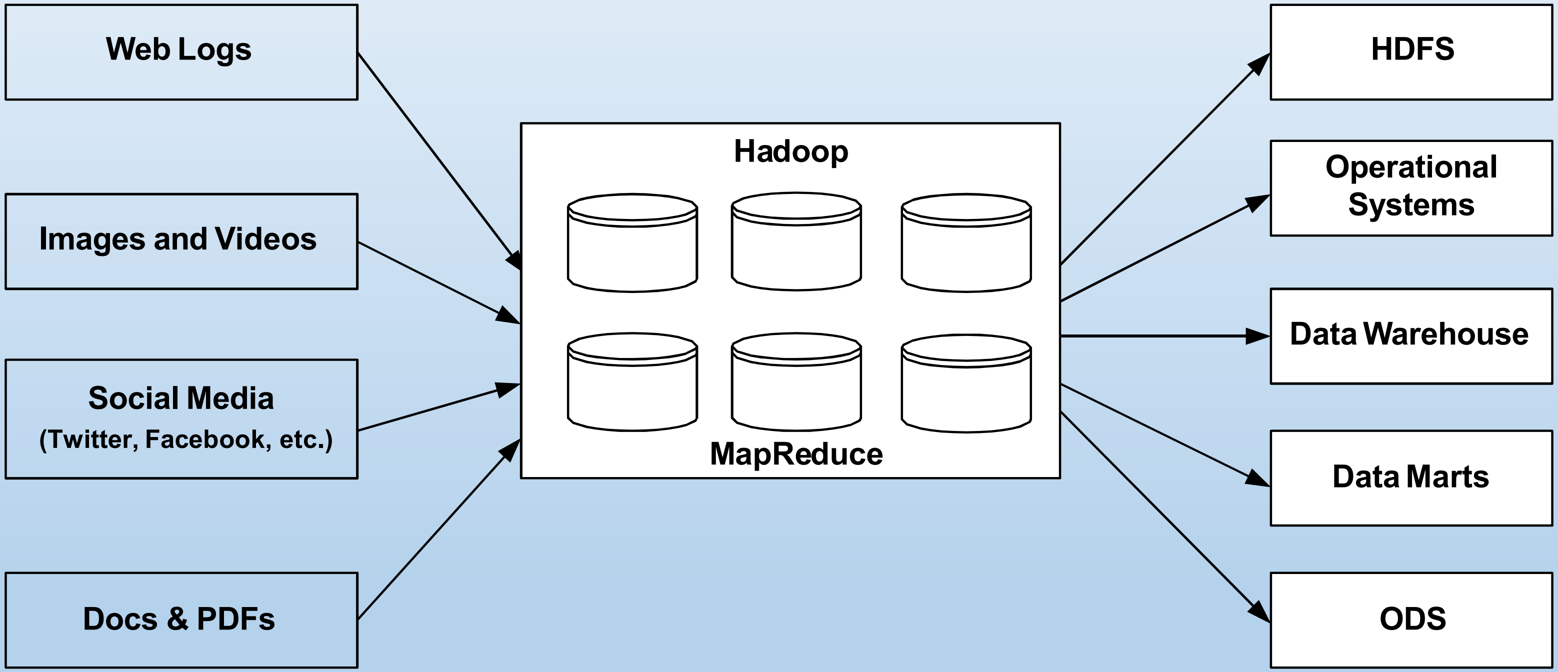
Forecasting the load on the electric grid over the next 24 hours is an example of predictive analytics, whereas deciding how to operate power plant based on this forecast represents prescriptive analytics.

Traditional Business Intelligence (BI) – (Data ware house) versus Big Data

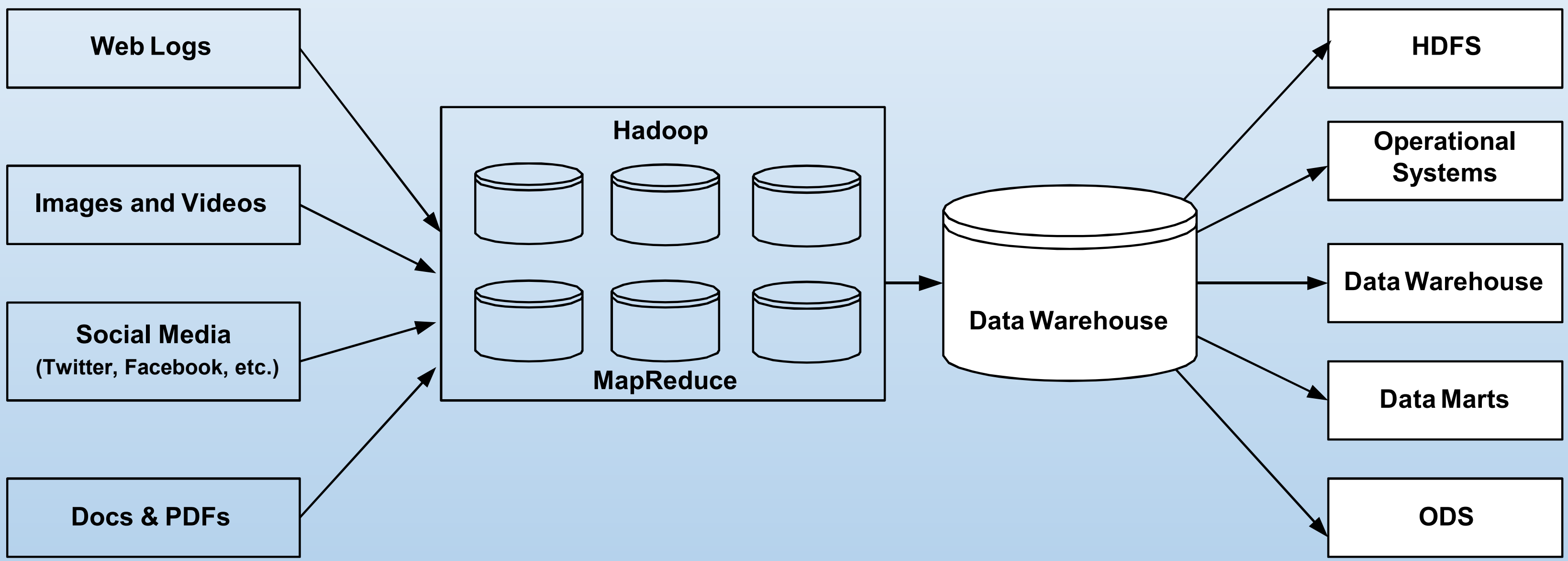
A Typical Data Warehouse Environment



A Typical Hadoop Environment



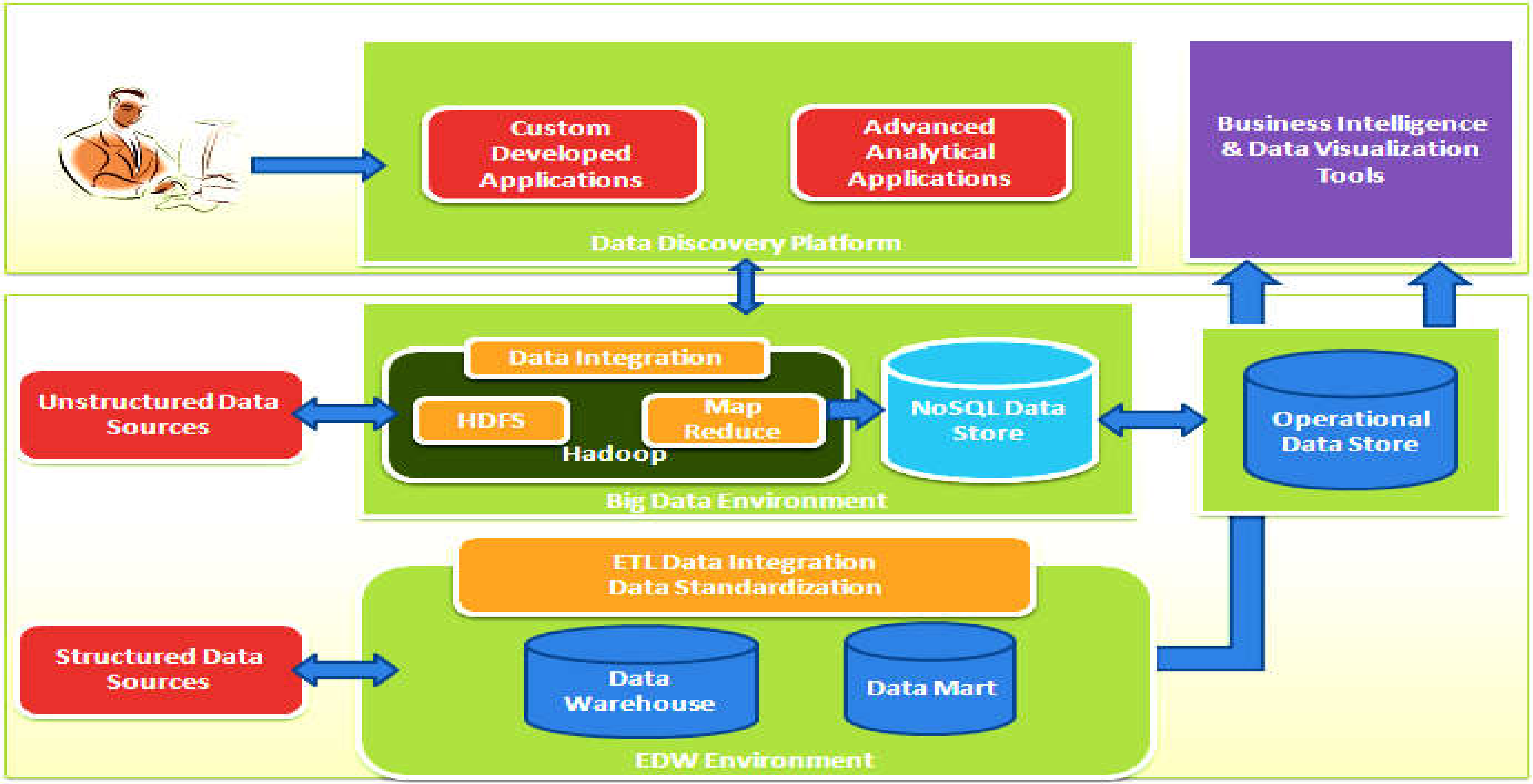
Co-existence of Big Data and Data Warehouse

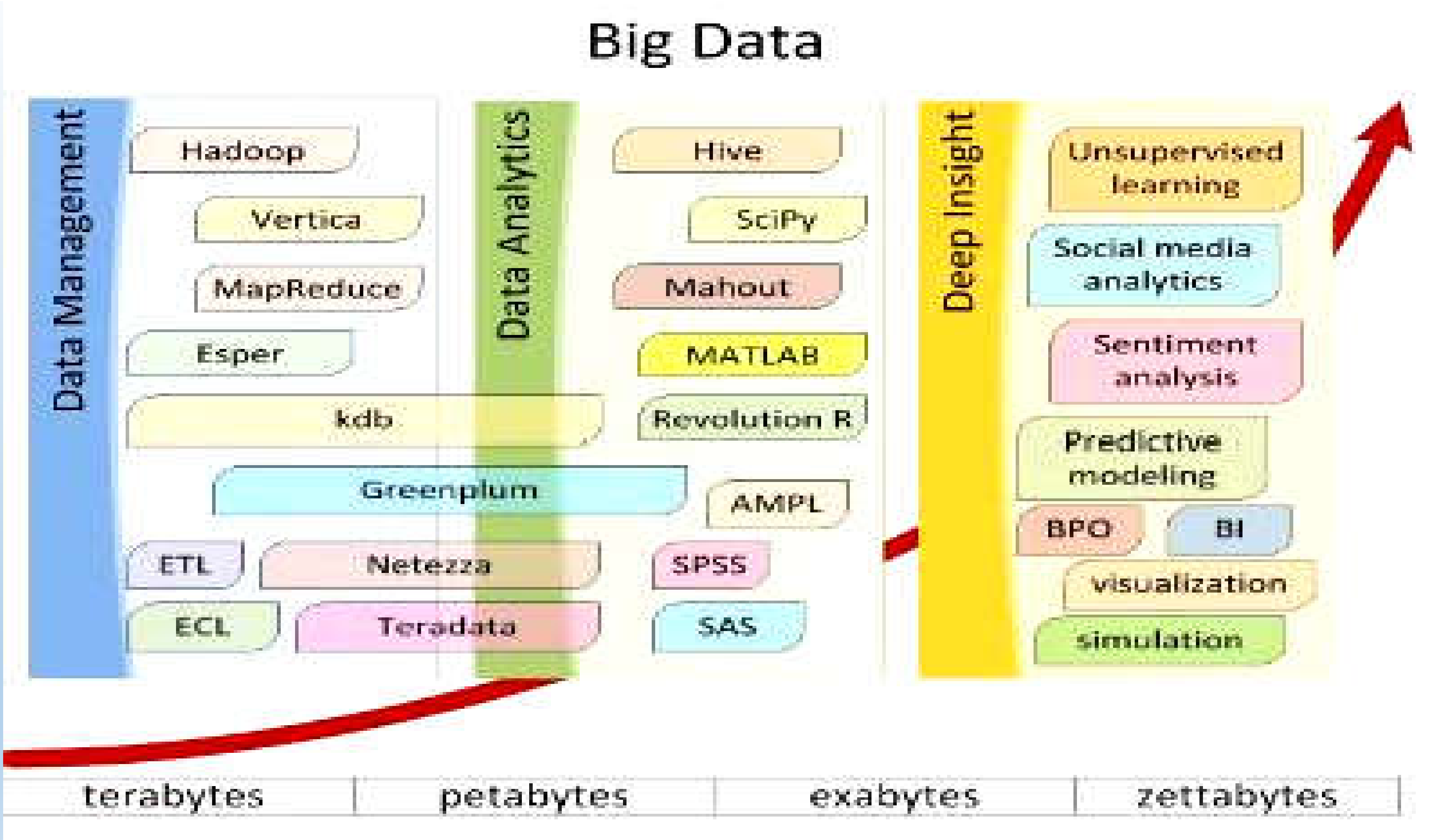


Difference between Traditional BI and Big Data

	Traditional BI	Big Data
Focus on	Descriptive analytics Diagnostic analysis	Predictive analytics
Data Set	Limited dataset Cleaned data simple models	Large scale datasets More Types of data Raw Data Complex data Models
Supports	Causation: what happened and why?	Correlation: New Insights and more accurate answers.

Enterprise data architecture using combined EDW/ Big data environment



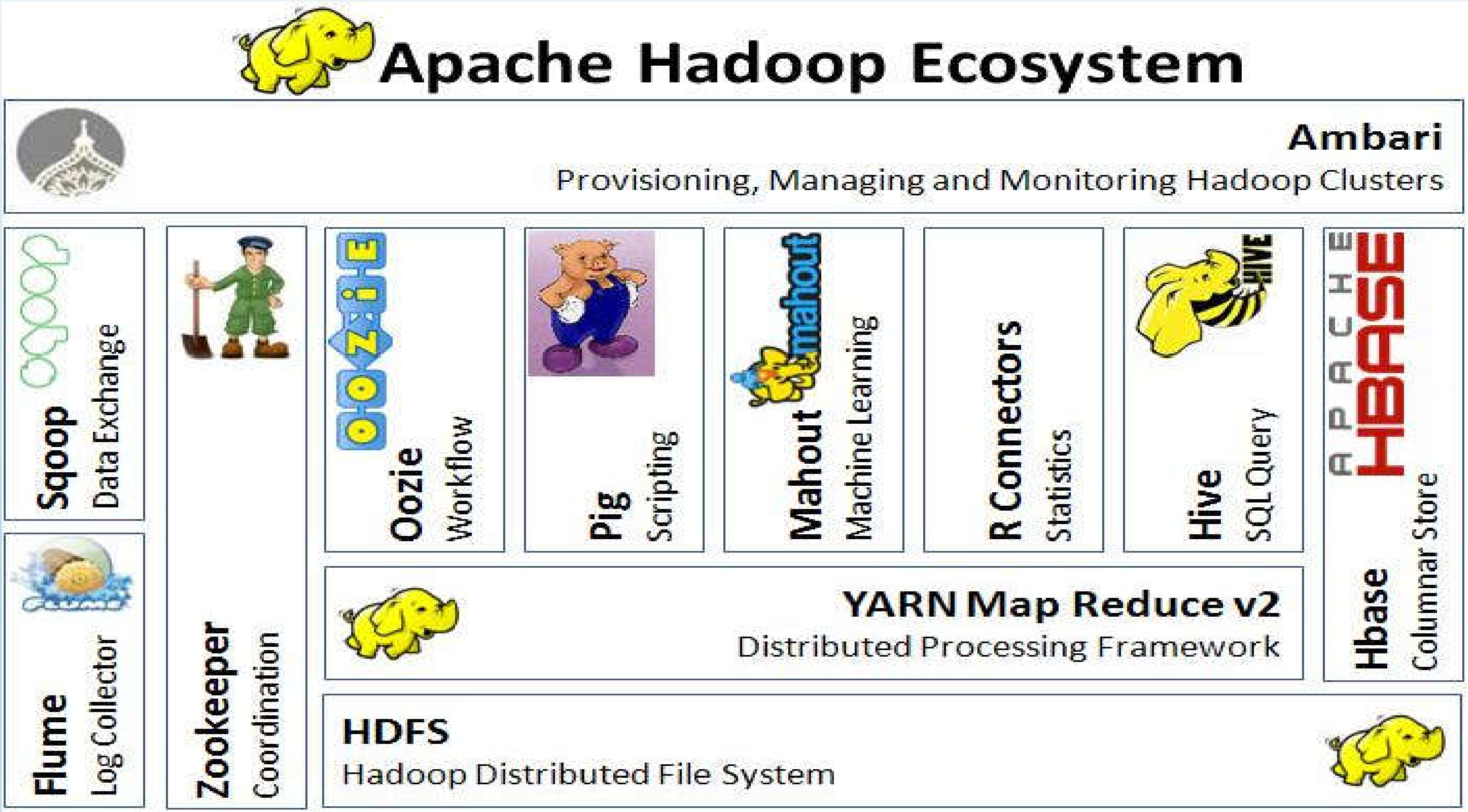

Big Data
Data Management
Hadoop
Vertica
MapReduce
Esper
kdb
Greenplum
ETL
Netezza
ECL
Teradata
Data Analytics
Hive
SciPy
Mahout
MATLAB
Revolution R
AMPL
SPSS
SAS
Deep Insight
Unsupervised learning
Social media analytics
Sentiment analysis
Predictive modeling
BPO
BI
visualization
simulation
terabytes
petabytes
exabytes
zettabytes

Technology for Big data analytics

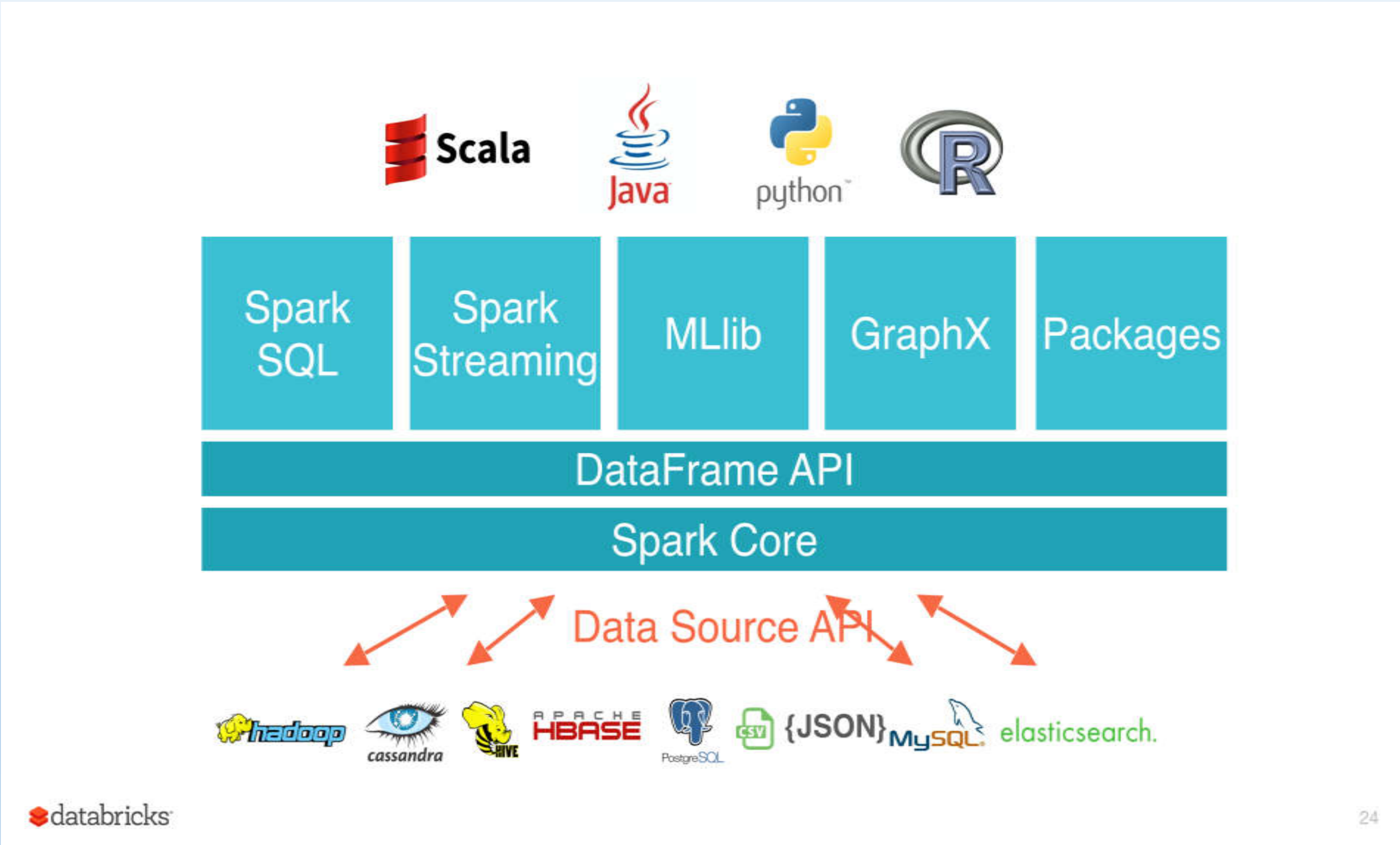
Document databases:
Pair each key with a complex data structure known as a document.
Graph stores :
Used to store information about networks of data, such as social connections.
Key-value stores Every single item in the database is stored as an attribute name (or 'key'), together with its value.
Wide-column stores : Store columns of data together, instead of rows

Type	Example	
Key-Value Store	redis	riak
Wide Column Store	HBASE	cassandra
Document Store	mongoDB	CouchDB relax
Graph Store	Neo4j	InfiniteGraph The Distributed Graph Database

Hadoop Eco systems



Spark Ecosystem



Top challenges facing Big data

- Scalability: Horizontal and vertical scalability
- Security: NOSQL dB
- Schema: Dynamic schemas
- Continuous availability: 24*7
- Consistency: Opt for consistency
- Partition tolerant: Tolerance to both hardware and software failures
- Data quality: Data accuracy , completeness

Kind of technologies looking toward to meet big data challenges

- Cheap and abundant storage
- Faster processors
- Affordable open sources, distributed platforms
- Parallel processing, clustering, Large grid environments
- Cloud computing and other flexible resource allocation arrangements.

Skill set
DBMS and NOSQL Databases Programming languages Open source tools Data warehousing / Data mining Statistics analysis Visualization AI and ML

Terminologies Used in Big data Environments

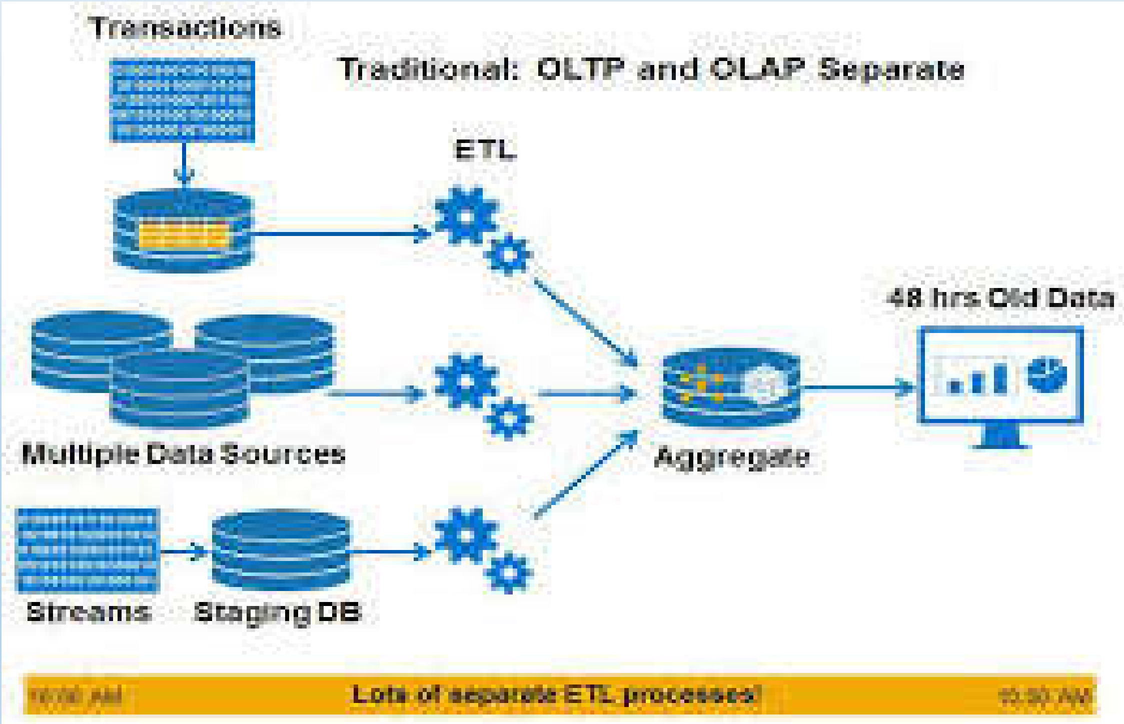
1. In-Memory Analytics
2. In-Database Processing
3. Massively Parallel Processing
4. Parallel System
5. Distributed System
6. Shared Nothing Architecture

In- Memory analytics

- Data access from non- volatile storage such as hard disk and processed in RAM
- Shortened query response times
- Preprocess data and store it as cubes, tables, query sets etc, So it handle small set of records.
- All relevant data stored in RAM

In- Database Processing

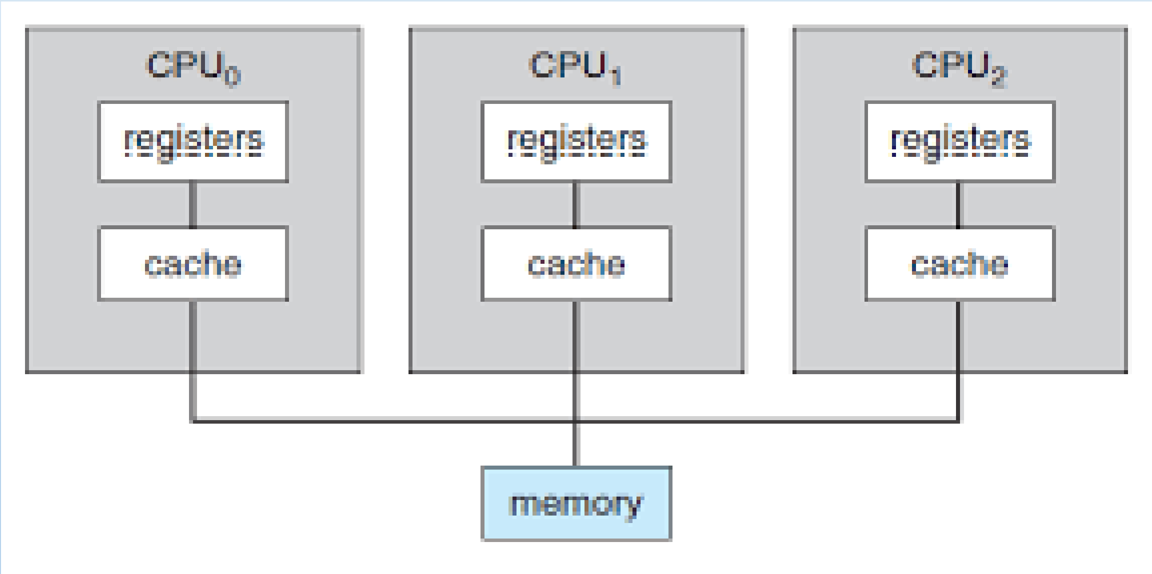
- It is also called as in- database analytics
- It works by fusing from - data warehouse / databases



Symmetric Multiprocessor System (SMP)

A **symmetric multiprocessor system** (SMP) is a **multiprocessor system** with centralized shared memory called main memory (MM) operating under a single operating **system** with two or more homogeneous processors

SMP is also called tightly coupled multiprocessing

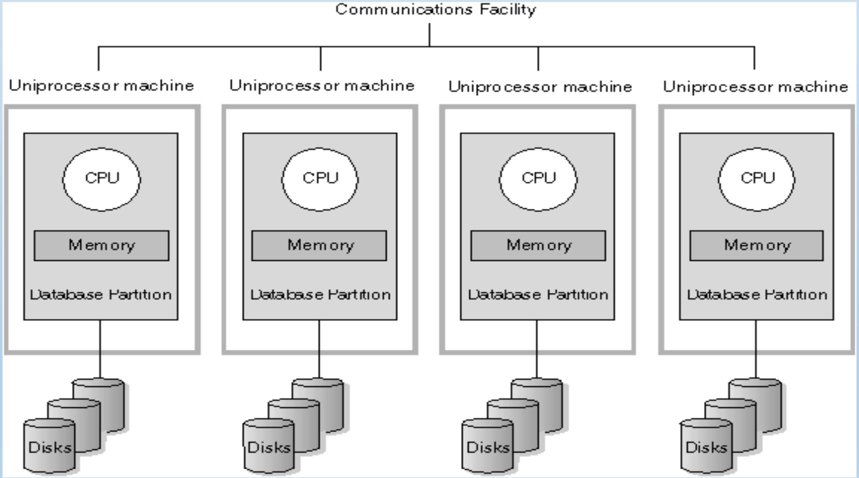


Massively Parallel Processing

- MPP (**massively parallel processing**) is the coordinated **processing** of a program by multiple processors that work on different parts of the program, with each processor using its own operating system and memory .
- Each process works on different part of the program / data.
- Typically, MPP processors communicate using some messaging interface.
- Data and processing is governed solely by **performance considerations.**
- Used for scientific applications.
- May be

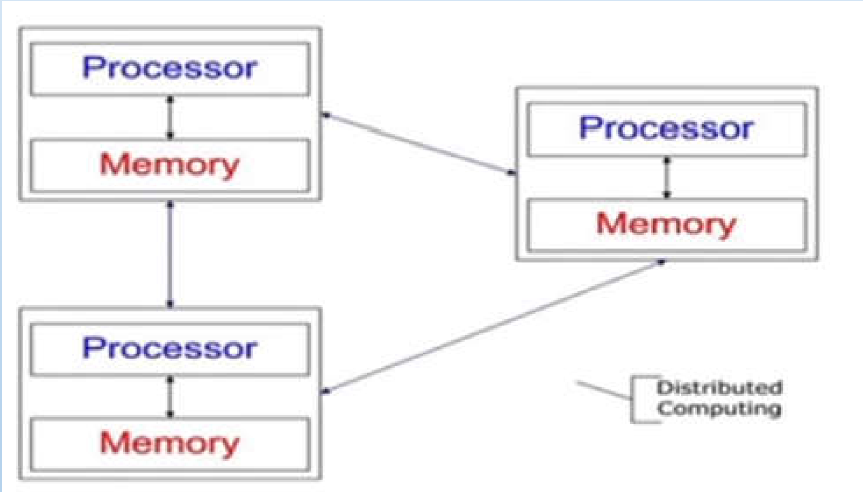
    tightly (Shared memory)

    loosely coupled (no shared memory)



Distributed Computing

- Different autonomous systems that have their own computational capacity and local memory.
- Systems Communicate with each other by message passing.
- Fault tolerant and high availability.
- Resource Sharing capability.
- Data and processing is governed solely by **application considerations.**
- Loosely Coupled System



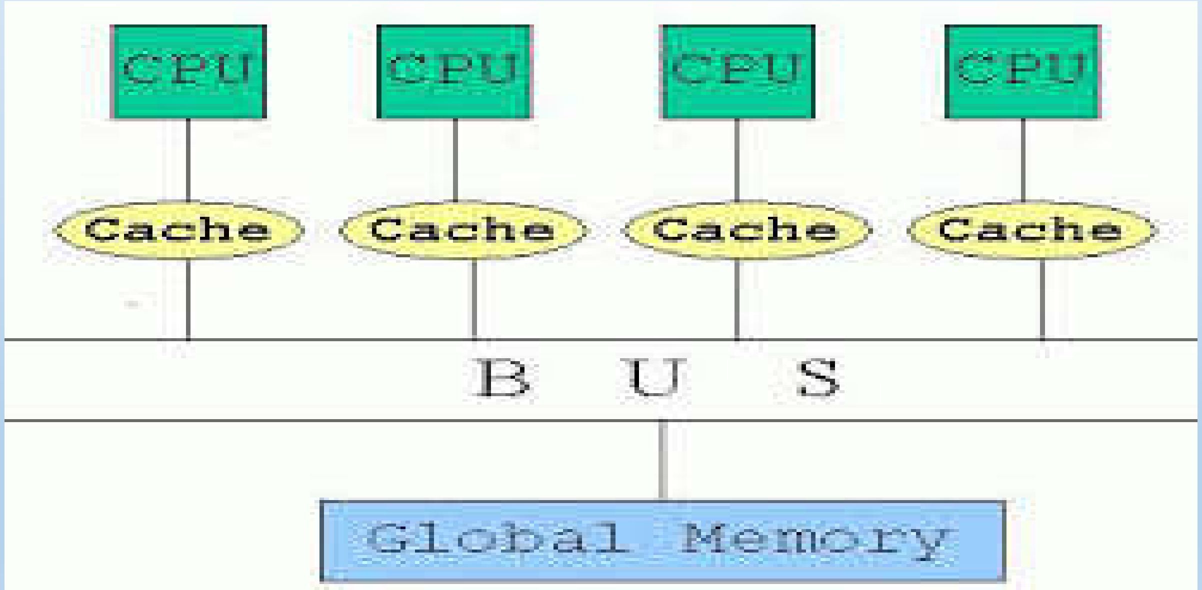
Multiprocessing Architecture

- Shared memory
- Shared disk
- Shared Nothing Architecture

shared memory

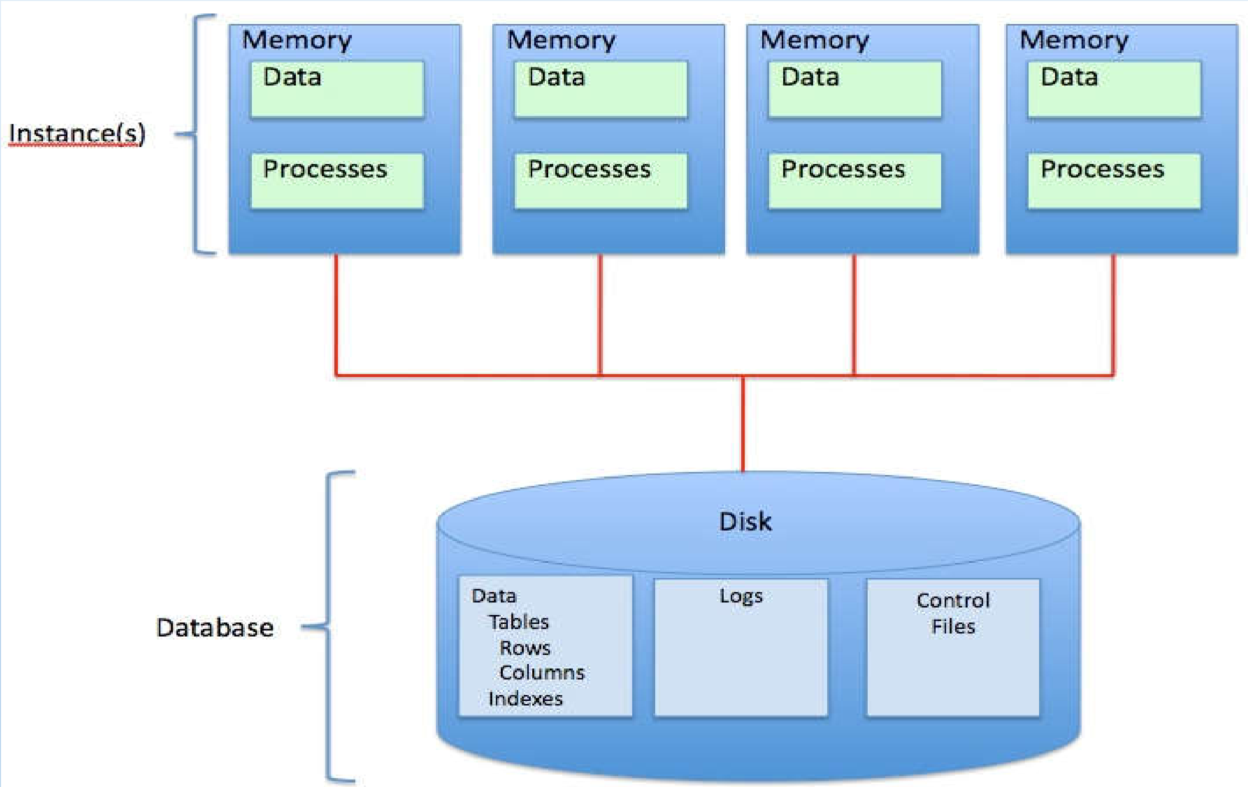
- **Shared memory** is **memory** that may be simultaneously accessed by multiple programs with an intent to provide communication among them or avoid redundant copies.

- **Shared memory** is an efficient means of passing data between programs.



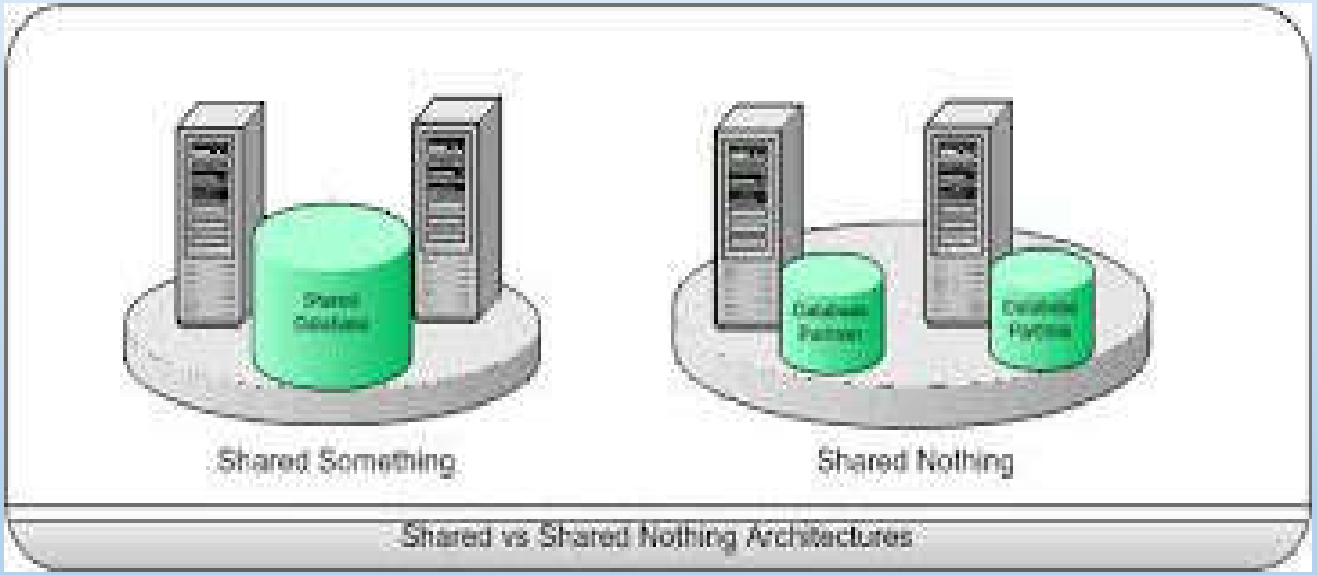
Shared Disk

A **shared disk** architecture (SD) is a distributed computing architecture in which all **disks** are accessible from all cluster nodes



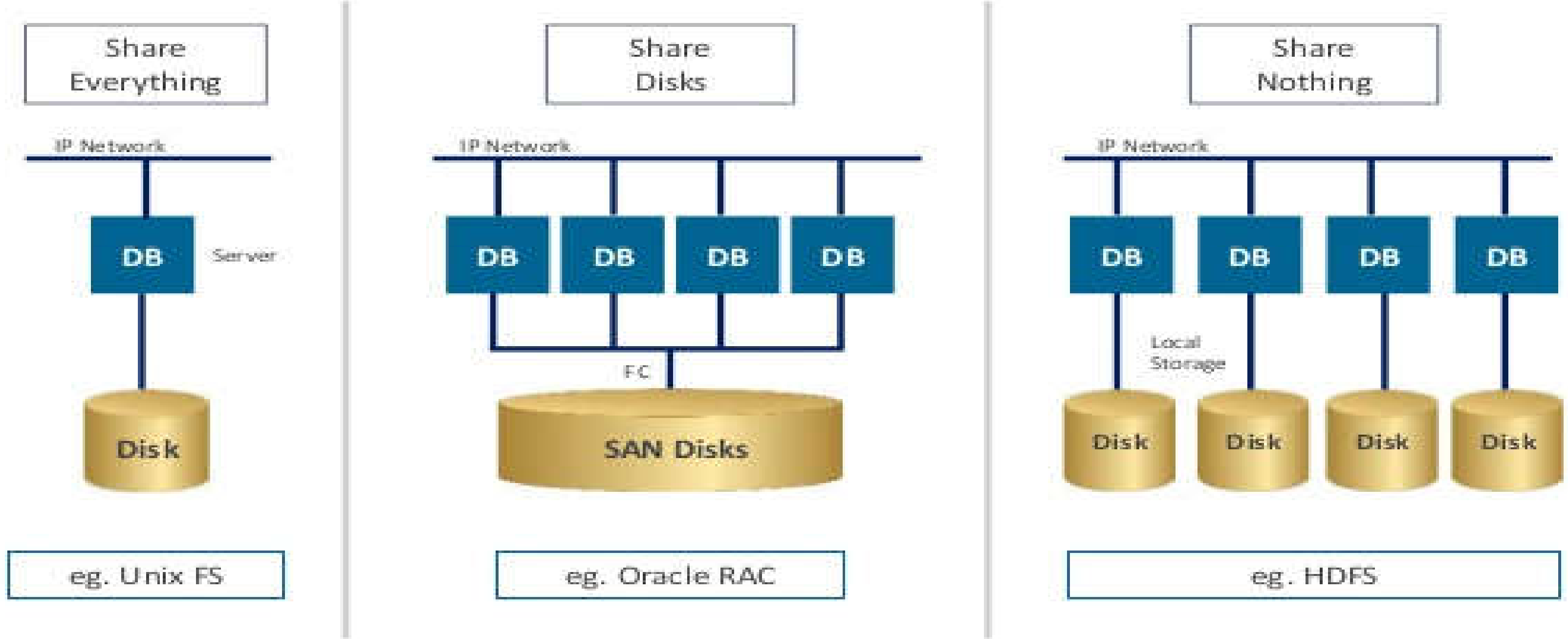
Shared Nothing Architecture

- A **shared nothing architecture** (SN) is a distributed computing **architecture**
- Each node is independent and self-sufficient, and there is no single point of contention across the system.
- More specifically, none of the nodes share memory or disk storage.



Shared vs Shared Nothing Architectures

SHARE NOTHING ARCHITECTURE



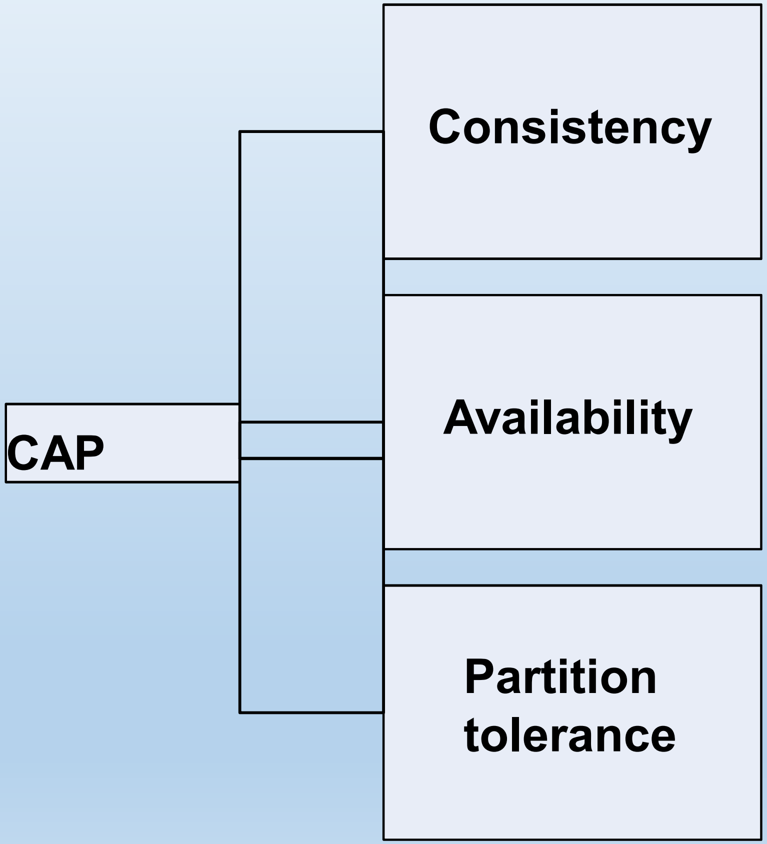
Advantages of shared Nothing architecture

1. Fault Tolerance

2. Scalability

CAP Theorem

Brewer's CAP



CAP Theorem

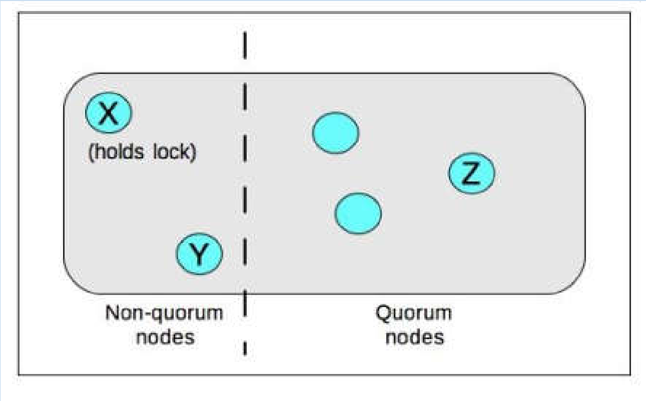
- **C**onsistency:
  All nodes should see the same data at the same time
- **A**vailability:
  Node failures do not prevent survivors from continuing to operate
  This condition states that every request gets a response on success/failure of nodes.
  Every client gets a response, regardless of the state of any individual node in the system.
- **P**artition-tolerance:
  The system continues to operate despite network partitions failures.
  partition-tolerant can sustain any amount of network failure that doesn't result in a failure of the entire network.
  Data records are sufficiently replicated across combinations of nodes and networks to keep the system up through intermittent outages.

A distributed system can satisfy any two of these guarantees at the same time **but not all three**

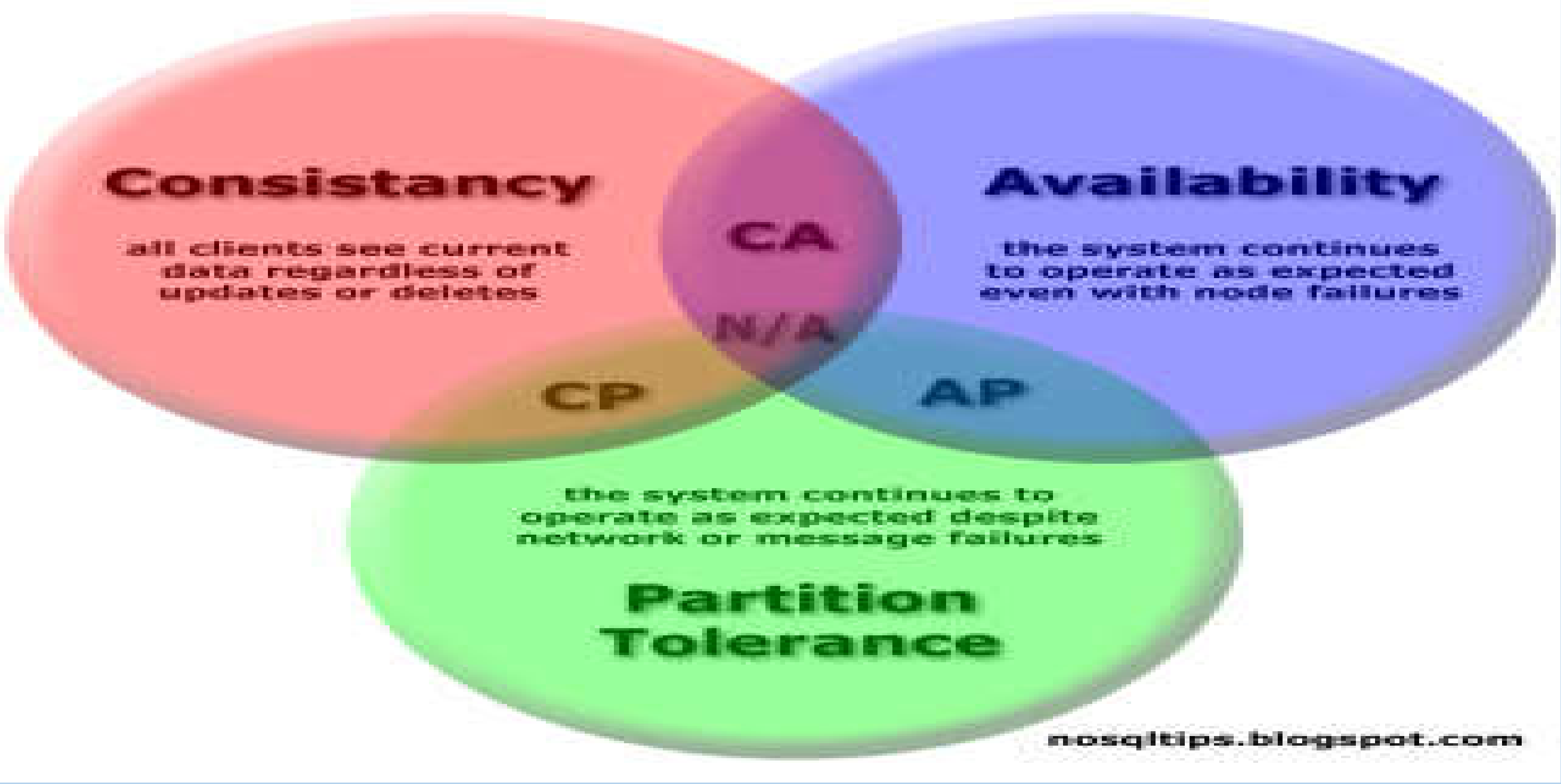
In an available but not partition-tolerant system, Y would be allowed to process its request because it can reach X to obtain the lock. Z's request would be blocked because X is unreachable.

In a partition-tolerant but not available system, Z would be allowed to process its request because it is part of the quorum group (X's lock will be broken). Y's request would be blocked because it is not part of the quorum group.

In a system that is both available and partition-tolerant, both requests would be allowed to progress. Y would return current data as possibly modified by X, while Z would return possibly stale data. Stale data could possibly mean no data in cases where there is no replica available with Quorum nodes. Consistency is obviously sacrificed in this case.

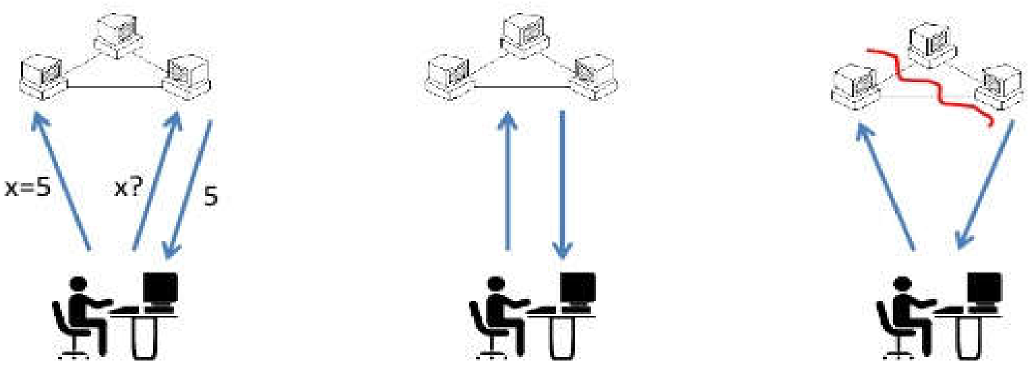


CAP theorem


Consistancy
all clients see current data regardless of updates or deletes
CA
Availability
the system continues to operate as expected even with node failures
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance
nosqltips.blogspot.com

CAP Theorem

Consistency Availability Partition tolerance



Why this is important?

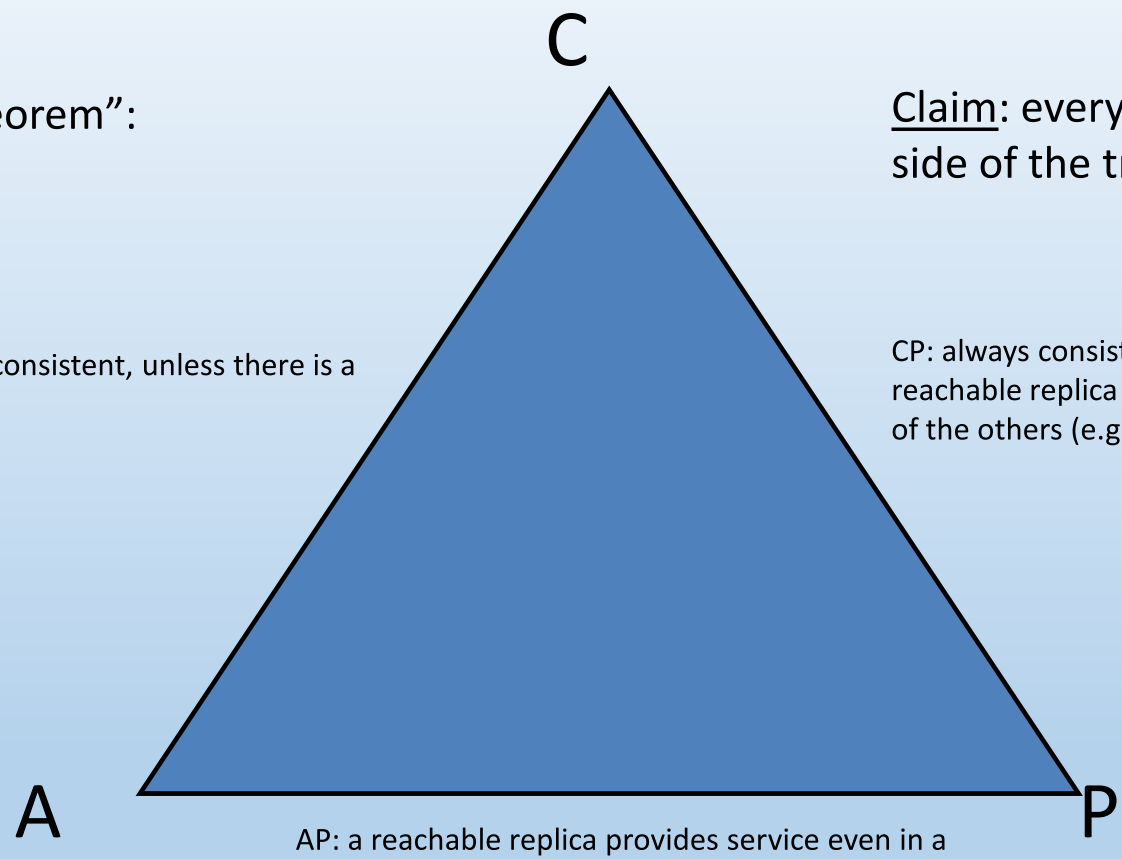
- The future of databases is **distributed** (Big Data Trend, etc.)

- CAP theorem describes the **trade-offs** involved in distributed systems

- A proper understanding of CAP theorem is essential to **making decisions** about the future of distributed database **design**

- Misunderstanding can lead to **erroneous or inappropriate** design choices consistency

Fox&Brewer “CAP Theorem”:
C-A-P: choose two.



<u>Claim</u>: every distributed system is on one side of the triangle.

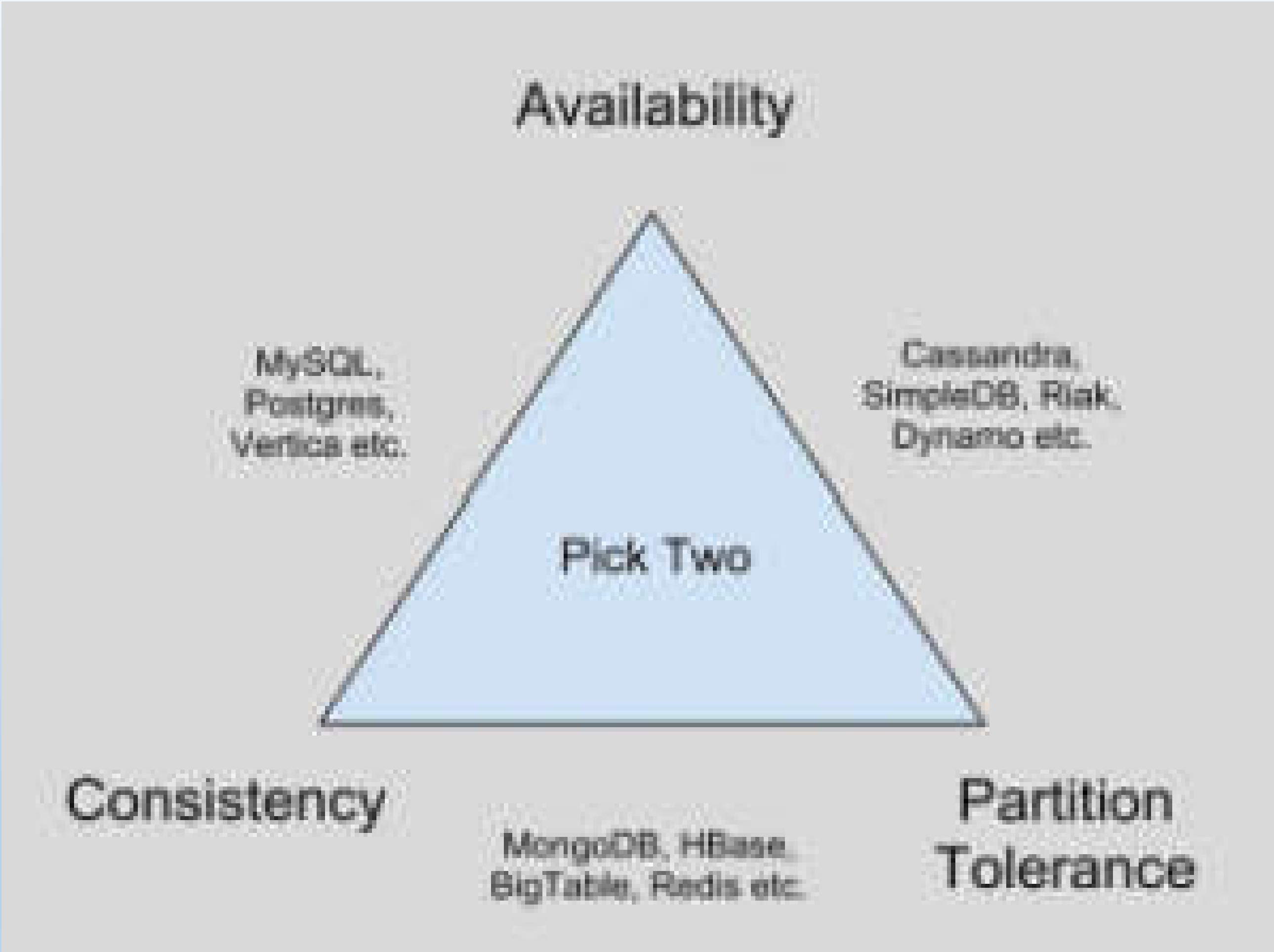
CA: available, and consistent, unless there is a partition.

CP: always consistent, even in a partition, but a reachable replica may deny service without agreement of the others (e.g., quorum).

AP: a reachable replica provides service even in a partition, but may be inconsistent.

Availability

Partition-resilience


Availability
MySQL, Postgres, Vertica etc.
Cassandra, SimpleDB, Riak, Dynamo etc.
Pick Two
Consistency
MongoDB, HBase, BigTable, Redis etc.
Partition Tolerance

When to consider consistency over availability and Vice -versa

1. Choose availability over consistency when your business requirements allow some flexibility around when data in the system synchronizes.

2. Choose consistency over availability when your business requirements demand atomic reads and writes.

Types of Consistency

Strong Consistency
After the update completes, **any subsequent access** will return the **same** updated value.

Weak Consistency
It is **not guaranteed** that subsequent accesses will return the updated value.

Eventual Consistency
Specific form of weak consistency
It is guaranteed that if **no new updates** are made to object, **eventually** all accesses will return the last updated value (e.g., *propagate updates to replicas in a lazy fashion*)

Eventual Consistency Variations

Causal consistency
Processes that have causal relationship will see consistent data

Read-your-write consistency
A process always accesses the data item after it’s update operation and never sees an older value

Session consistency
As long as session exists, system guarantees read-your-write consistency
Guarantees do not overlap sessions

Eventual Consistency Variations

Monotonic read consistency
If a process has seen a particular value of data item, any subsequent processes will never return any previous values

Monotonic write consistency
The system guarantees to serialize the writes by the *same* process

In practice
A number of these properties can be combined
Monotonic reads and read-your-writes are most desirable

Eventual Consistency
- A Facebook Example

Bob finds an interesting story and shares with Alice by posting on her Facebook wall
Bob asks Alice to check it out
Alice logs in her account, checks her Facebook wall but finds:
- **Nothing is there!**

Eventual Consistency
- A Facebook Example

Bob tells Alice to wait a bit and check out later
Alice waits for a minute or so and checks back:

- **She finds the story Bob shared with her!**

Eventual Consistency
- A Facebook Example

Reason: it is possible because Facebook uses an **eventual consistent model**

Why Facebook chooses eventual consistent model over the strong consistent one?

- Facebook has more than 1 billion active users
- It is non-trivial to efficiently and reliably store the huge amount of data generated at any given time
- Eventual consistent model offers the option to **reduce the load and improve availability**

BASE

- Basically Available Soft state Eventual consistency

- Where it is used? – Distributed Computing

- Why?- To achieve high availability

- How it is achieved? - No new updates made to the data for a stipulated period of time , eventually all accesses to this data will return the updated value.

- What is replica convergence? system that has achieved eventual consistency is said to have converged.

- Conflict resolution: solved by
    1. Read repair
    2. Write repair
    3. Asynchronous repair

ACID vs BASE

ACID

- Strong consistency for transactions highest priority
- Availability less important
- Complex mechanisms

BASE

- Availability and scaling highest priorities
- Weak consistency
- Simple and fast

Thank you