Big Data Technology Stack

Why are specialized big data tools required?

Attributes	Traditional	Big Data	Note
Volume	GBs to TBs	PBs to Zeta Bytes	The amount of data is shifted from TBs to PBs.
Organization	Centralised	Distributed	The data is stored in distributed systems instead of a single system.
Data Type	Structured	Semi structured and unstructured	Today's data consists of structured, semi-structured and unstructured data.
Data Model	Fixed Schema	Flat Schema	Structured data has a fixed schema while big data has flat schema

- Earlier Approach – When this problem came to existence, Google tried to solve it by introducing GFS and Map Reduce process. (https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf)

- These two are based on distributed file systems and parallel processing. The framework was very successful.

- Apache Hadoop is an open source implementation of the Map-Reduce framework.

Why Open Source Tools or Technologies…

- Big companies like Google, Facebook, Twitter etc. all are now contributing to big data open source projects along with thousands of volunteers.
- There are lots of advantages to using open source tools such as flexibility, agility, speed, information security, shared maintenance cost and they also attract better talent.
- Since open source tools are cost effective as compared to proprietary solutions, they provide the ability to start small and scale up in the future.
- Anyone can pick up from a lot of alternatives and if the fit is right then they can scale up with a commercial solution.
- Most mobile, web, and cloud solutions use open source platforms and the trend will only rise upwards, so it is potentially going to be the future of IT.

Parameters for selecting Big Data Enabling Technologies

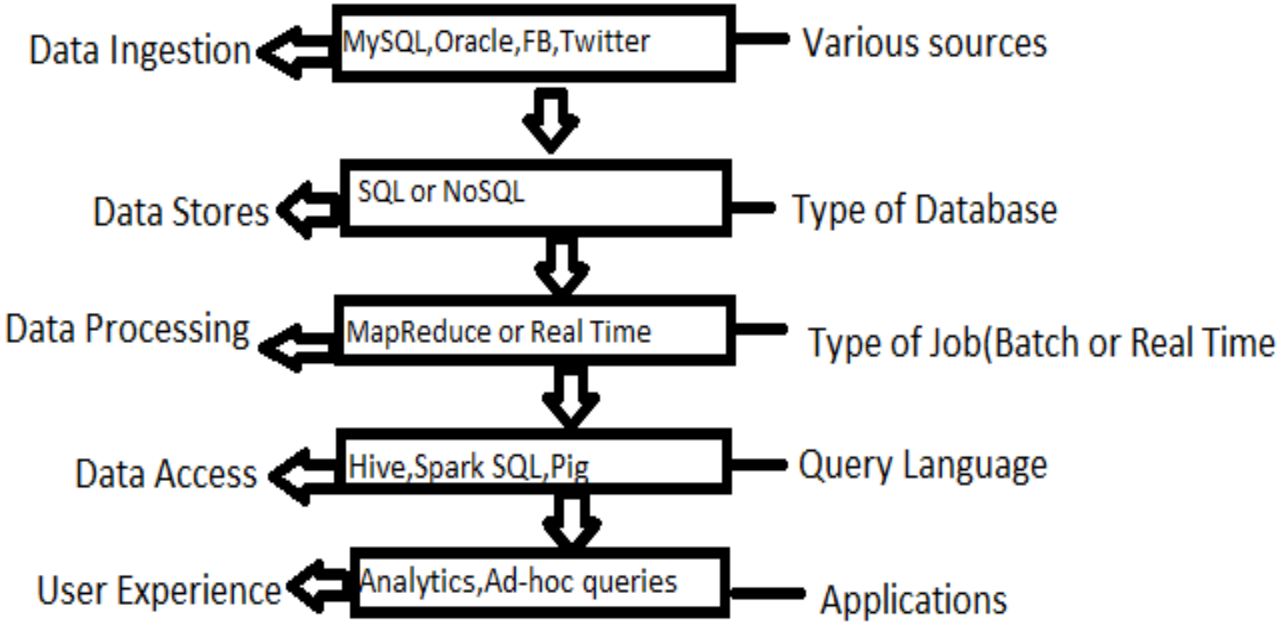
- **Reputation** – What is the general consensus about tools and reviews from in production users?
- **Popularity** – How popular and active is the open source community behind the technology?
- **Ongoing efforts** – What is the technology roadmap for the next 3-5 years?
- **Standards** – Which technical specifications does the technology qualify and which industry implementation standards does it adhere to?
- **Interoperability** – Following standards does ensure interoperability, but there are many interoperability standards too. Big data as a service and with cloud will demand interoperability features.
- **Support (Community and Commercial)** – Open source tools suffer when dedicated resources/volunteers are not keeping technologies up to date and commercial offerings become vital. Many a times, latest required features take years to become available.
- **Skill Set** – Is the tool easy to use and extend?
- **Project Model** – Open source technologies tend to cease with lesser popularity and become commercial with greater popularity. Some open source projects start off as free and many features are offered as paid or do it yourself. It is important to choose technologies that will remain open source.
- **Documentation** – Open source tools suffer from ease of use for the lack of better documentation. But that is mitigated by an active large community.
- **License** – Open source is free but sometimes not entirely free. Choose a tool that will continue to grow with the community.

Big Data Stack for Big Data Analytics :

- The first step in the process is getting the data. It is needed to ingest big data and then store it in data stores (SQL or No SQL).
- Once data has been ingested, after noise reduction and cleansing, big data is stored for processing.
- There are two types of data processing, Map Reduce (batch processing) and Real Time.
- Scripting languages are needed to access data or to start the processing of data.
- After processing, the data can be used in various fields. It may be used for analysis, machine learning, and can be presented in graphs and charts (Data visualization tools)….

Need for Big Data Stack….providing all above functionalities.

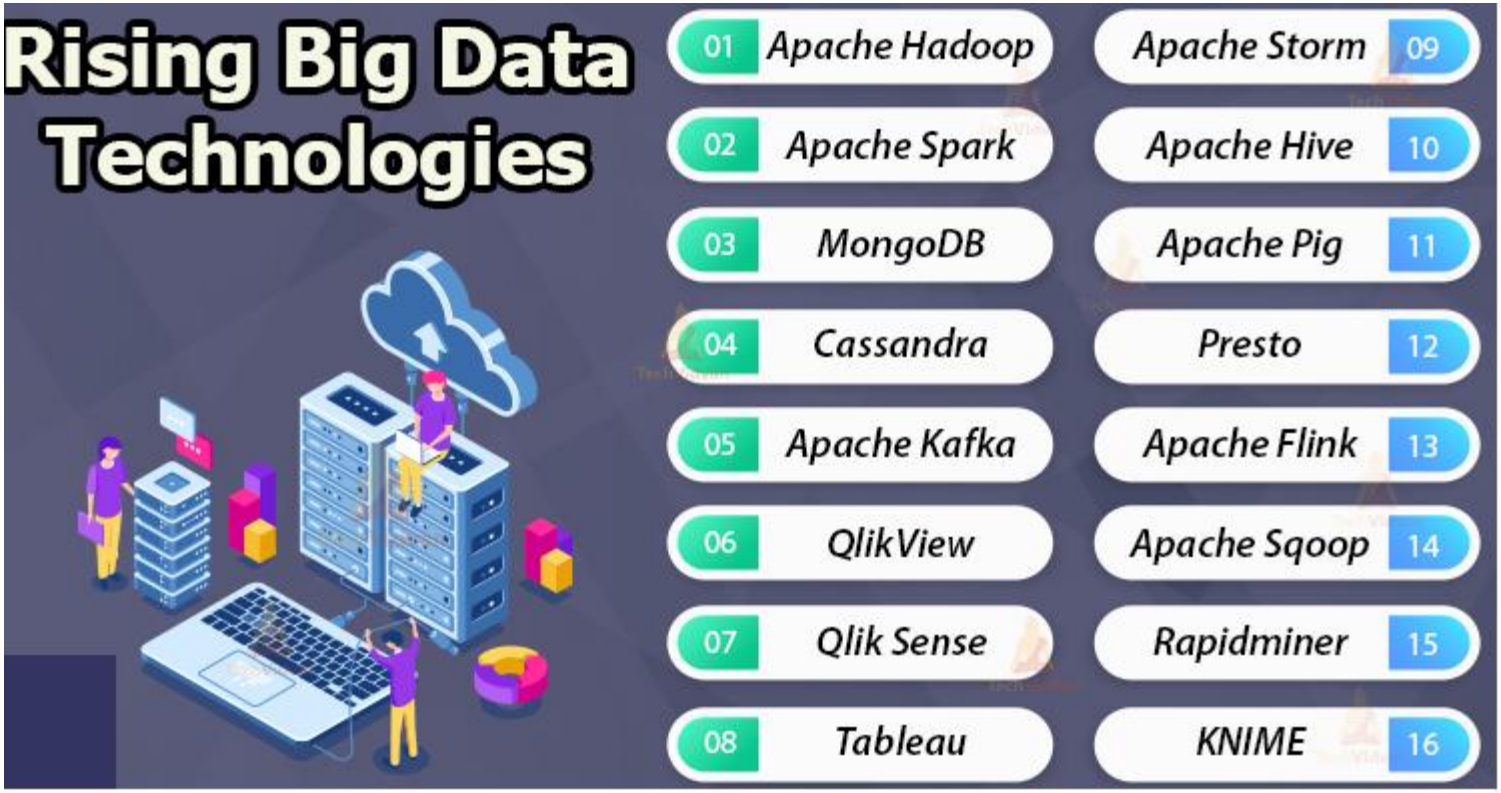
Big Data Stack



• Each big data stack provides many open source alternatives. It is the deployment environment that dictates the choice of technologies to adopt.

• Each tool is good at solving one problem and together big data provides billions of data points to gather business and operational intelligence.

Big Data Technologies



Apache Eco-system

- Scoop : It is a tool designed for efficiently transferring bulk data between Apache Hadoop and **structured data-stores such as relational databases**.

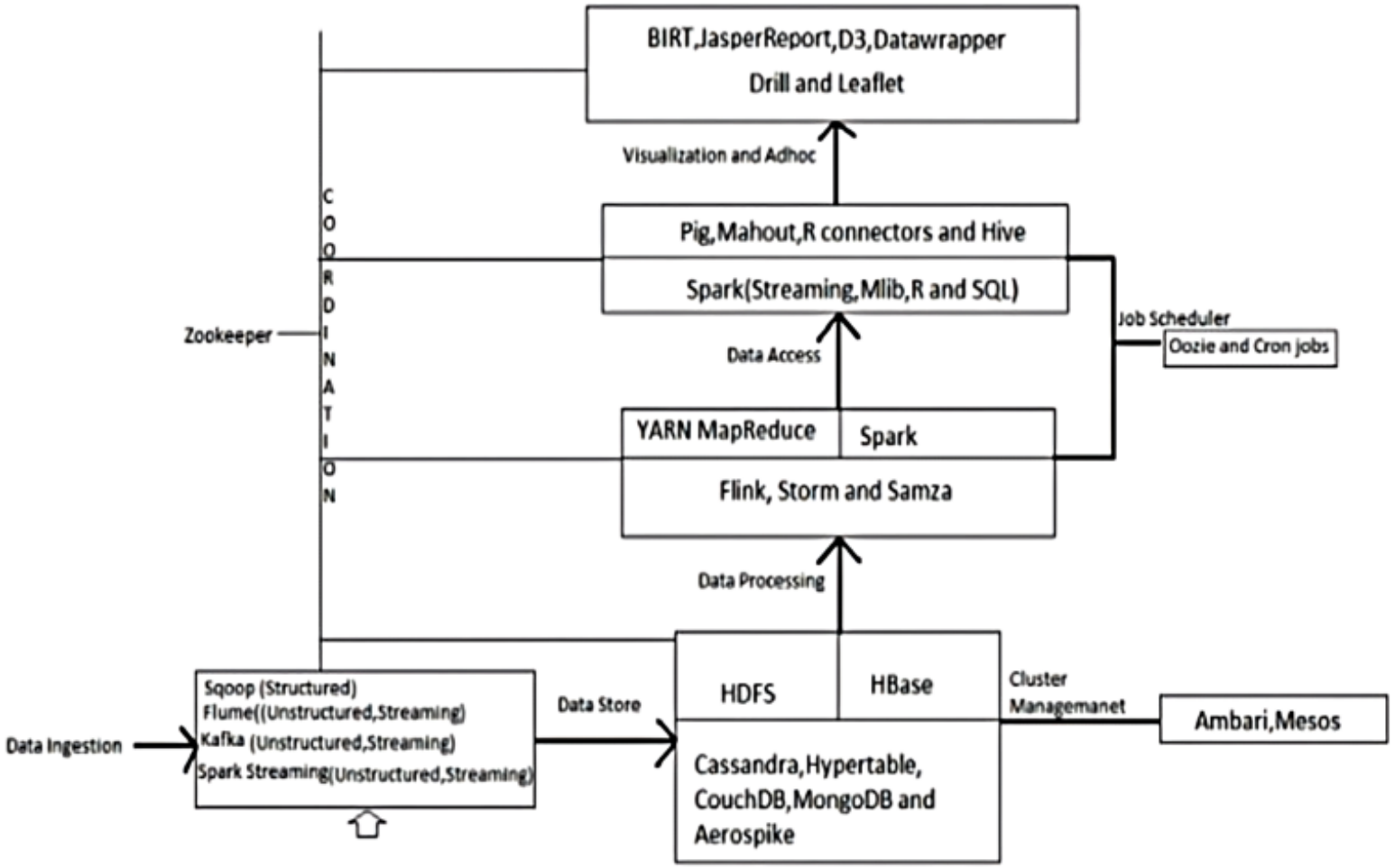
- Flume : It is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

- Ambari : Hadoop management tool which offers provisioning, managing, and monitoring Apache Hadoop clusters.

- Cassandra : It is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers support for clusters spanning multiple datacenters, with asynchronous master-less replication allowing low latency operations for all clients. Cassandra was designed to implement a combination of Amazon's Dynamo distributed storage and replication techniques combined with Google's Big table data and storage engine model.

- Kafka : is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

Big Data Open Source Strategies



Big data distribution packages…

- Big data processing and distribution systems offer a way to collect, distribute, store, and manage massive, unstructured data sets in real time.
- These solutions provide a simple way to process and distribute data amongst parallel computing clusters in an organized fashion.
- Built for scale, these products are created to run on hundreds or thousands of machines simultaneously, each providing local computation and storage capabilities.
- Big data processing and distribution systems provide a level of simplicity to the common business problem of data collection at a massive scale and are most often used by companies that need to organize an exorbitant amount of data.
- Many of these products offer a distribution that runs on top of the open-source big data clustering tool Hadoop.
- Companies commonly have a dedicated administrator for managing big data clusters. The role requires in-depth knowledge of database administration, data extraction, and writing host system scripting languages.

Big data distribution packages…

- Administrator responsibilities often include implementation of data storage, performance upkeep, maintenance, security, and pulling the data sets.
- Businesses often use big data analytics tools to then prepare, manipulate, and model the data collected by these systems.
- To qualify for inclusion in the Big Data Processing And Distribution Systems category, a product must:

✓ Collect and process big data sets in real-time

✓ Distribute data across parallel computing clusters

✓ Organize the data in such a manner that it can be managed by system administrators and pulled for analysis

✓ Allow businesses to scale machines to the number necessary to store its data

Big data Analytics

- Big data analytics software provides insights into large data sets that are collected from big data clusters. These tools help business users digest data trends, patterns, and anomalies and synthesize the information into understandable data visualizations, reports, and dashboards.
- Because of the unstructured nature of big data clusters, these analytics solutions often require a query language to pull the data out of the file system.
- Some solutions may offer self-service features so that non-technical employees can assemble their own charts and graphs from big data sets.
- Some big data analytics solutions offer features powered by machine learning, such as natural language processing, allowing the user to query company data in a natural manner.
- Big data analytics software is commonly used at companies running Hadoop in conjunction with big data processing and distribution software to collect and store data. In addition, these products typically integrate with data warehouse software, the central storage hub for a company's integrated data.

Big data distribution packages…

- Google Big Query
- Snowflake
- Databricks Lakehouse Platform
- Confluent
- Azure Data Lake Store
- Qubole
- Dremio
- Amazon EMR
- IBM BigInsights
- Hadoop HDFS
- TIMi Suite
- Vertica
- Google Cloud Dataflow
- Azure HDInsight

Big data Analytics and Analytics Platforms

- Big data analytics software differs from analytics platforms in as much as the former are solely focused on the manipulation of complex and large scale big data clusters into understandable visualizations, while the latter are geared toward a wide range of data sources and connectors.
- The two categories are mutually exclusive, and those products which are solely focused on big data use cases are only categorized in the big data analytics category.
- To qualify for inclusion in the Big Data Analytics category, a product must:

✓ Consume data, query file systems, and connect directly to big data clusters

✓ Allow users to prepare complex big data sets into helpful and understandable data visualizations

✓ Create business-applicable reports, visualizations, and dashboards based on discoveries inside the data sets

Big Data Platform

- Big data platform is a type of IT solution that combines the features and capabilities of several big data application and utilities within a single solution.
- It is an enterprise class IT platform that enables organization in developing, deploying, operating and managing a big data infrastructure /environment.
- Big data platform generally consists of big data storage, servers, database, big data management, business intelligence and other big data management utilities.
- It also supports custom development, querying and integration with other systems. The primary benefit behind a big data platform is to reduce the complexity of multiple vendors/ solutions into a one cohesive solution.
- Big data platform are also delivered through cloud where the provider provides an all inclusive big data solutions and services.

Big Data Platform Examples

- **GOOGLE CLOUD**

  Google Cloud offers lots of big data management tools, each with its own specialty. BigQuery warehouses petabytes of data in an easily queried format. Dataflow analyzes ongoing data streams and batches of historical data side by side. With Google Data Studio, clients can turn varied data into custom graphics.

- **MICROSOFT AZURE**

  Users can analyze data stored on Microsoft's Cloud platform, Azure, with a broad spectrum of open-source Apache technologies, including Hadoop and Spark. Azure also features a native analytics tool, HDInsight, that streamlines data cluster analysis and integrates seamlessly with Azure's other data tools.

Big Data Platform Examples

- **AMAZON WEB SERVICES**
- Best known as AWS, Amazon's cloud-based platform comes with analytics tools that are designed for everything from data prep and warehousing to SQL queries and data lake design. All the resources scale with your data as it grows in a secure cloud-based environment. Features include customizable encryption and the option of a virtual private cloud.
-
- **CLOUDERA**
- Rooted in Apache's Hadoop, Cloudera can handle massive amounts of data. Clients routinely store more than 50 petabytes in Cloudera's Data Warehouse, which can manage data including machine logs, text, and more. Meanwhile, Cloudera's DataFlow — previously Hortonworks' DataFlow — analyzes and prioritizes data in real time.

Big Data Platform Examples

- **SUMO LOGIC**

  The cloud-native Sumo Logic platform offers apps — including Airbnb and Pokémon GO — three different types of support. It troubleshoots, tracks business analytics and catches security breaches, drawing on machine learning for maximum efficiency. It's also flexible and able to manage sudden influxes of data.

- **SISENSE**

  Sisense's data analytics platform processes data swiftly thanks to its signature In-Chip Technology. The interface also lets clients build, use and embed custom dashboards and analytics apps. And with its AI technology and built-in machine learning models, Sisense enables clients to identify future business opportunities.

Big Data Platform Examples

- **TABLEAU**

  The Tableau platform — available on-premises or in the cloud — allows users to find correlations, trends and unexpected interdependences between data sets. The Data Management add-on further enhances the platform, allowing for more granular data cataloging and the tracking of data lineage.

- **COLLIBRA**

  Designed to accommodate the needs of banking, healthcare and other data-heavy fields, Collibra lets employees company wide find quality, relevant data. The versatile platform features semantic search, which can find more relevant results by unraveling contextual meanings and pronoun referents in search phrases.

- **ALTERYX**

  Alteryx's designers built the company's eponymous platform with simplicity and interdepartmental collaboration in mind. Its interlocking tools allow users to create repeatable data workflows — stripping busywork from the data prep and analysis process — and deploy R and Python code within the platform for quicker predictive analytics.

- Many more…

CAP Theorem

- Conjectured by Prof. Eric Brewer at PODC (Principle of Distributed Computing) 2000 keynote talk

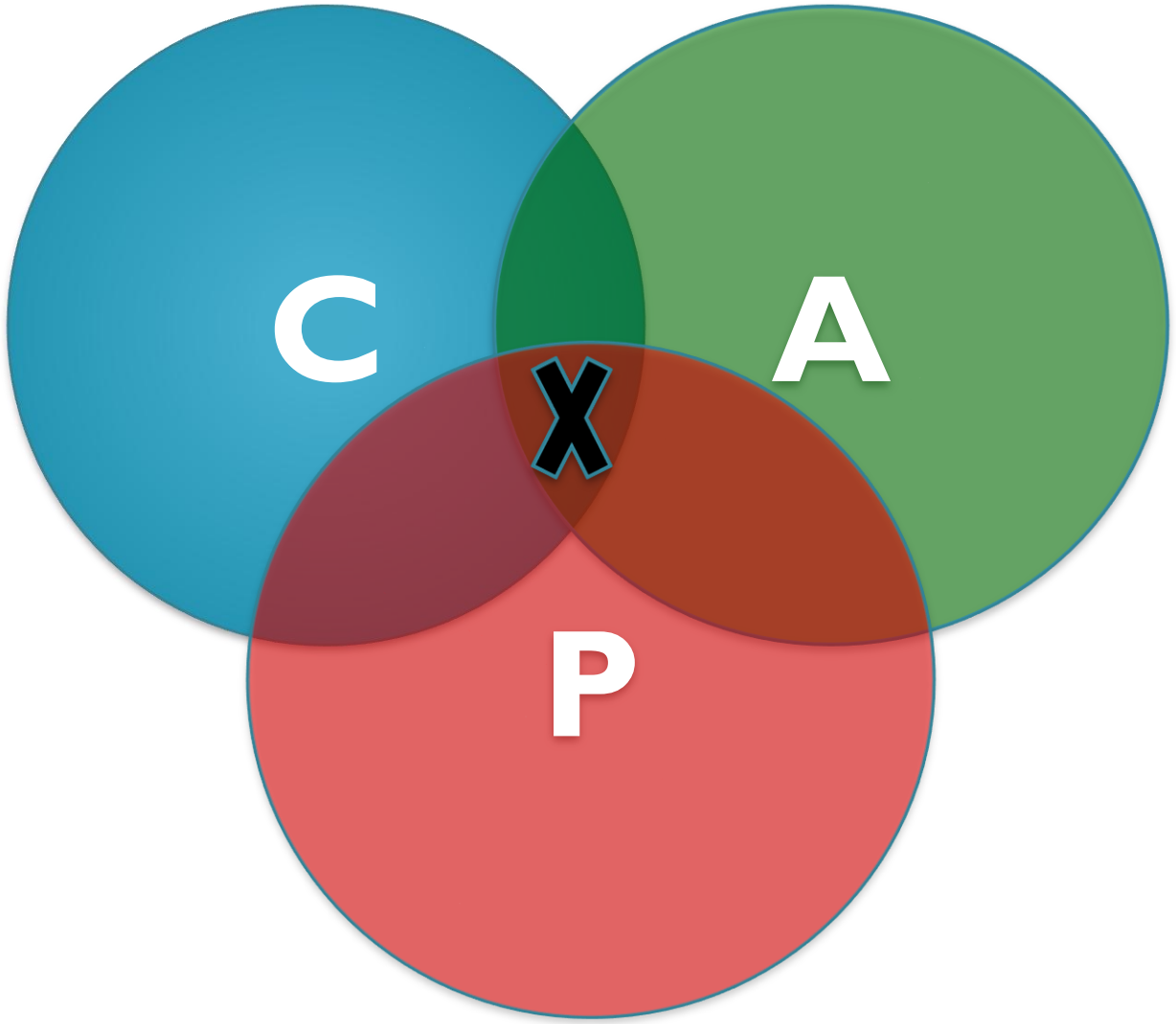
- Described the *trade-offs involved in distributed system*

- It is impossible for a web service to provide following *three guarantees at the same time*:
  - **Consistency**
  - **Availability**
  - **Partition-tolerance**

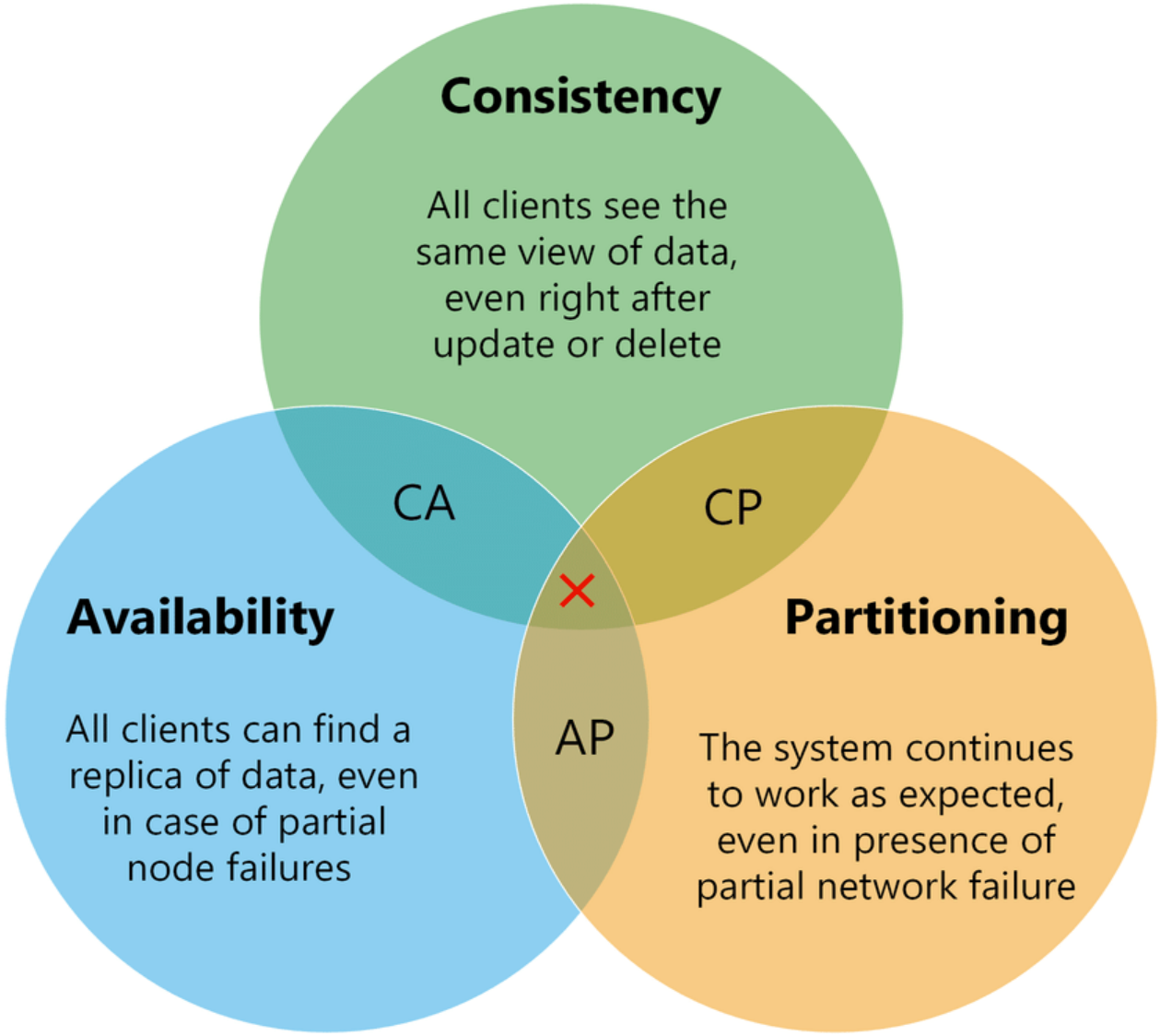
CAP Theorem

- In theoretical computer science, the **CAP theorem**, states that any distributed data store can only provide two of the following three guarantees –
- Consistency : Every read receives the most recent write or an error.
- Availability : Every request receives a (non-error) response, without the guarantee that it contains the most recent write.
- Partition tolerance : The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.
- When a network partition failure happens, it must be decided whether to -
  - cancel the operation and thus decrease the availability but ensure consistency or to
  - proceed with the operation and thus provide availability but risk inconsistency.

CAP Theorem

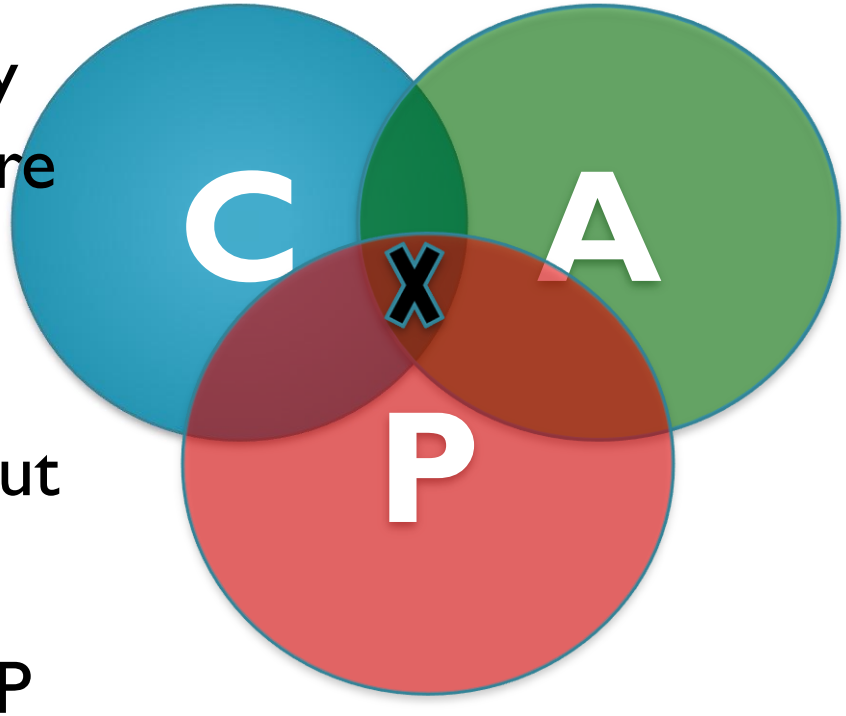


CAP Theorem

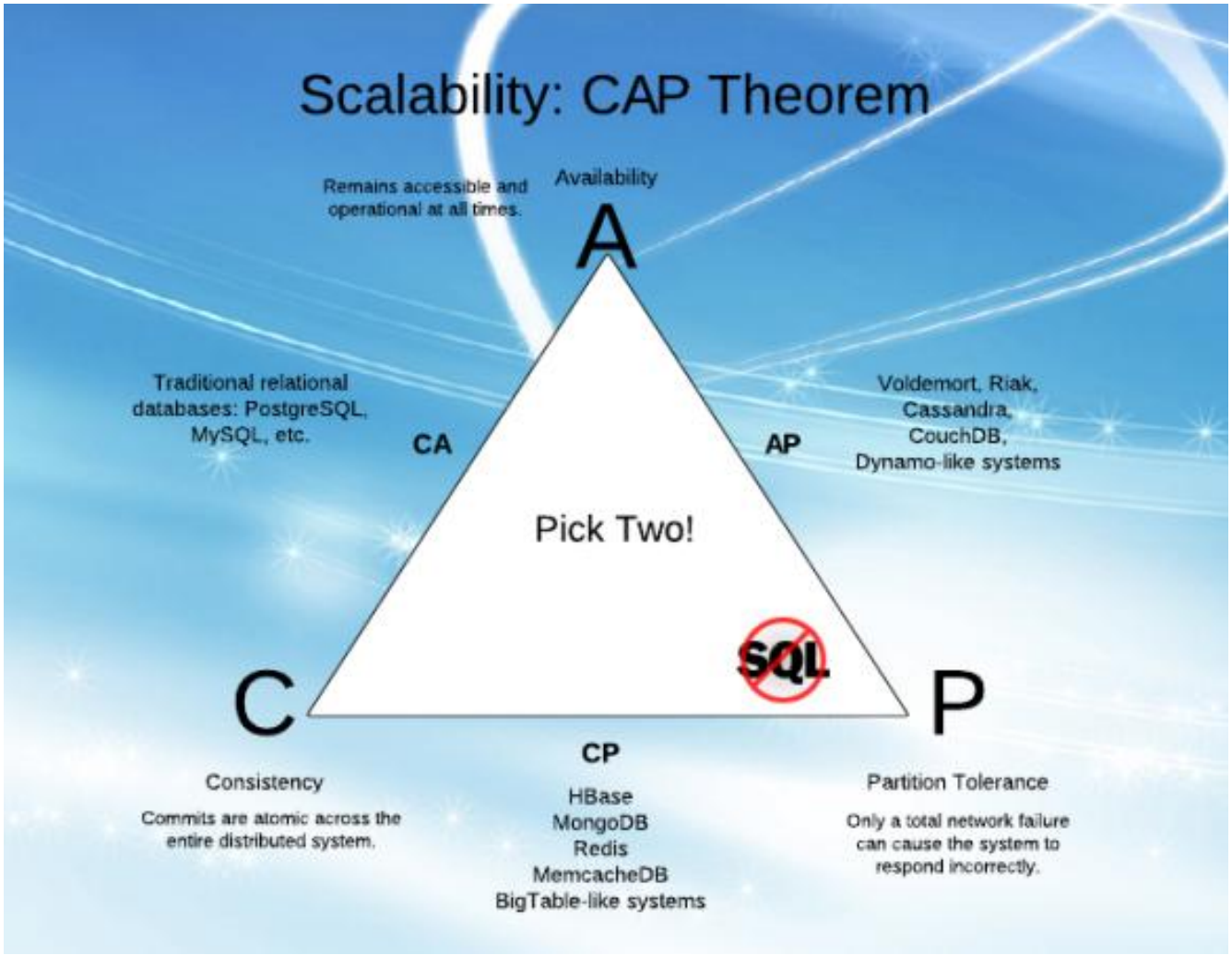

Consistency
All clients see the same view of data, even right after update or delete
CA
CP
Availability
All clients can find a replica of data, even in case of partial node failures
AP
Partitioning
The system continues to work as expected, even in presence of partial network failure

Consistency or Availability

- Consistency and Availability is not "binary" decision
- AP systems relax consistency in favor of availability – but are not inconsistent
- CP systems sacrifice availability for consistency- but are not unavailable
- This suggests both AP and CP systems can offer a degree of consistency, and availability, as well as partition tolerance



CAP Theorem


Scalability: CAP Theorem
Availability
Remains accessible and operational at all times.
A
Traditional relational databases: PostgreSQL, MySQL, etc.
CA
AP
Voldemort, Riak, Cassandra, CouchDB, Dynamo-like systems
Pick Two!
SQL
C
P
CP
Consistency
Commits are atomic across the entire distributed system.
HBase
MongoDB
Redis
MemcacheDB
BigTable-like systems
Partition Tolerance
Only a total network failure can cause the system to respond incorrectly.

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** (Optimistic / Lazy Replication)
  - 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

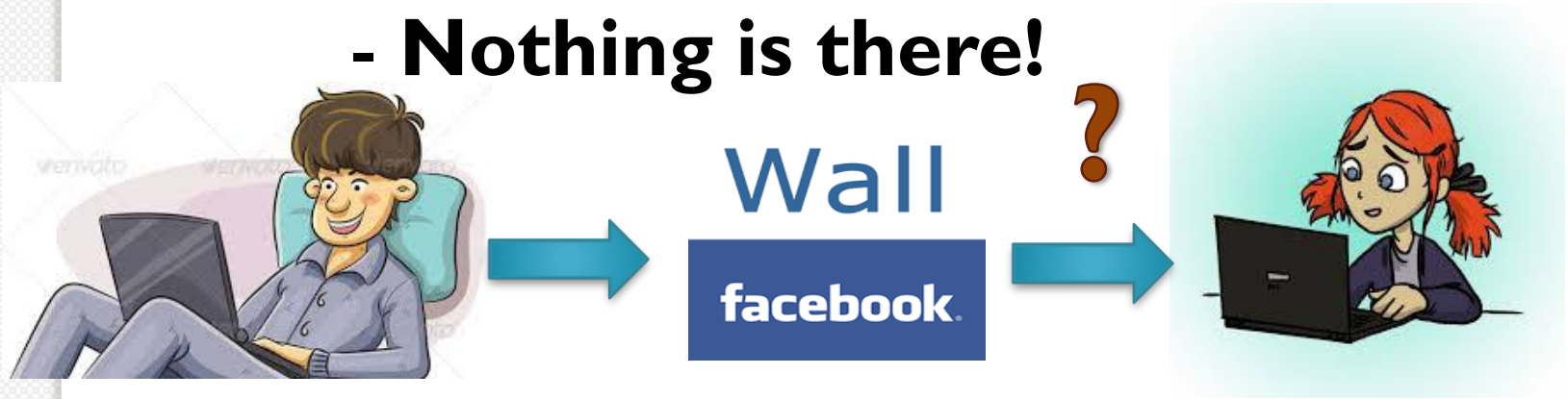
Eventual Consistency Variations

- 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**

Eventual Consistency - A Dropbox Example

- Dropbox enabled immediate consistency via synchronization in many cases.
- However, what happens in case of a network partition?

Eventual Consistency - A Dropbox Example

- Let's do a simple experiment here:
  - Open a file in your drop box
  - Disable your network connection (e.g., WiFi, 4G)
  - Try to edit the file in the drop box: can you do that?
  - Re-enable your network connection: what happens to your dropbox folder?

Eventual Consistency - A Dropbox Example

- Dropbox embraces eventual consistency:
  - Immediate consistency is impossible in case of a network partition
  - Users will feel bad if their word documents freeze each time they hit Ctrl+S , simply due to the large latency to update all devices across WAN
  - Dropbox is oriented to **personal syncing**, not on collaboration, so it is not a real limitation.

Eventual Consistency - An ATM Example

- In design of automated teller machine (ATM):
  - Strong consistency appear to be a nature choice
  - However, in practice, **A beats C**
  - Higher availability means **higher revenue**
  - ATM will allow you to withdraw money *even if the machine is partitioned from the network*
  - However, it puts **a limit** on the amount of withdraw (e.g., $200)
  - The bank might also charge you a fee when a overdraft happens

Dynamic Tradeoff between **C** and **A**

- An airline reservation system:
  - When most of seats are available: it is ok to rely on somewhat out-of-date data, availability is more critical
  - When the plane is close to be filled: it needs more accurate data to ensure the plane is not overbooked, consistency is more critical
- Neither strong consistency nor guaranteed availability, but it may significantly increase the tolerance of network disruption

Heterogeneity: Segmenting **C** and **A**

- No single uniform requirement
  - Some aspects require strong consistency
  - Others require high availability
- Segment the system into different components
  - Each provides different types of guarantees
- Overall guarantees neither consistency nor availability
  - Each part of the service gets exactly what it needs
- Can be partitioned along different dimensions

Discussion

- In an e-commercial system (e.g., Amazon, e-Bay, etc), what are the trade-offs between consistency and availability you can think of? What is your strategy?
- Hint -> Things you might want to consider:
  - Different types of data (e.g., shopping cart, billing, product, etc.)
  - Different types of operations (e.g., query, purchase, etc.)
  - Different types of services (e.g., distributed lock, DNS, etc.)
  - Different groups of users (e.g., users in different geographic areas, etc.)

ACID

- Atomicity
  - all or nothing
- Consistency
  - data is always in a valid state
- Isolation
  - concurrently executing transactions are isolated
- Durability
  - once committed, changes made by transaction are permanent

BASE

- Basically Available Soft-state Eventual Consistency
  - Forfeits the "C" and "I" of ACID in return for availability
  - most NoSQL systems provide BASE instead of ACID guarantees

ACID v/s BASE

- ACID
  - Focuses on consistency and isolation
- BASE
  - weak consistency
    - data can be stale
  - availability first
  - simpler implementation
  - faster
    - is this always true?

ACID Model

- The ACID database transaction model ensures that a performed transaction is always consistent. This makes it a good fit for businesses which deal with online transaction processing (e.g., finance institutions) or online analytical processing (e.g., data warehousing).

- These organizations need database systems which can handle many small simultaneous transactions. There must be zero tolerance for invalid states

ACID : Examples

- Financial institutions will almost exclusively use ACID databases. Money transfers depend on the atomic nature of ACID.
- An interrupted transaction which is not immediately removed from the database can cause a lot of issues. Money could be debited from one account and, due to an error, never credited to another.

- **Which Databases are ACID compliant?**
- One safe way to make sure your database is ACID compliant is to choose a relational database management system. These include MySQL, PostgreSQL, Oracle, SQLite, and Microsoft SQL Server.

- Some NoSQL DBMSs, such as Apache's CouchDB or IBM's Db2, also possess a certain degree of ACID compliance. However, the philosophy behind the NoSQL approach to database management goes against the strict ACID rules. Hence, NoSQL databases are not the recommended choice for those who need strict environments.

BASE Model

- **Basically Available** – Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.

- **Soft State** – Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to developers.

- **Eventually Consistent** – The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality).

BASE : Example

- Marketing and customer service companies who deal with sentiment analysis will prefer the elasticity of BASE when conducting their social network research. Social network feeds are not well structured but contain huge amounts of data which a BASE-modeled database can easily store.

- **Which Databases are Using the BASE Model?**
- Just as SQL databases are almost uniformly ACID compliant, NoSQL databases tend to conform to BASE principles. MongoDB, Cassandra and Redis are among the most popular NoSQL solutions, together with Amazon DynamoDB and Couchbase.

ACID-BASE Trade-offs

- ACID-compliant databases will be a better fit for those who require consistency, predictability, and reliability.
- Those who consider growth to be among their priorities will likely want to choose the BASE model, because it enables easier scaling up and provides more flexibility.
- However, BASE also requires developers who will know how to deal with the limitations of the model.

Apache Zookeeper

- **Apache ZooKeeper** is an open-source server for highly reliable distributed coordination of cloud applications.
- ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
- All of these kinds of services are used in some form or another by distributed applications.