NOSQL

Mirunalini.P

SSNCE

May 25, 2022

Table of Contents

Session Objective

- Emergence of NOSQL systems
- Characteristics of NOSQL systems
- Categories of NOSQL systems
- CAP Theorem

Session Outcome

At the end of this session, participants will be able to

- Understand the NOSQL Systems
- Understand the categories of NOSQL system
- Understand CAP theorem

NOSQL

- The term NOSQL is generally interpreted as "Not Only SQL" rather than "NO to SQL" to manage the big data needs.
- NOSQL systems are distributed databases or distributed storage systems
- Focus on semistructured data storage, high performance, availability, data replication, and scalability and opposed to an emphasis on immediate data consistency, powerful query languages, and structured data storage.
- Different class of systems have been developed to manage large amounts of data in organizations such as **Google, Amazon, Facebook, and Twitter**
- Typical applications that use NOSQL systems are social media, posts and tweets, weblinks, road maps and spatial data etc.,

When to use NOSQL?

- When huge amount of data need to be stored and retrieved
- The relationship between the data you store is not that important
- The data changing over time and is not structured.
- Support of Constraints and Joins is not required at database level
- The data is growing continuously and you need to scale the database regular to handle the data.

Emergence of NOSQL

Organizations that were faced with data management and storage applications decided to develop their own systems:

- **BigTable** : Googles proprietary NOSQL system, Column-based or wide column store referred as column family store.
- **DynamoDB (Amazon)** : Key-value data store
- **Cassandra (Facebook)**: Uses concepts from both key-value store and column-based systems
- **MongoDB and CouchDB** : Document stores
- **Neo4J and GraphBase** : Graph-based NOSQL systems
- **OrientDB** : Combines several concepts Database systems classified on the object model Or native XML model

NOSQL Characteristics related to distributed databases and distributed systems

NOSQL systems emphasize

- High availability
- Scalability (to adopt large volume of data)
- High performance

NOSQL Characteristics related to distributed databases and distributed systems

- **Scalability:** It determines the extent to which the system can expand its capacity without interruption.
- **Horizontal scalability**: The distributed system is expanded by adding more nodes for data storage and processing as the volume of data grows.
- **Vertical scalability:** Each existing nodes are expanded by expanding the storage and computing power

Techniques for distributing the existing among new nodes without interrupting system operations are necessary.

NOSQL Characteristics related to distributed databases and distributed systems

Availability, Replication and Eventual Consistency:

- NOSQL systems require **continuous system availability.**
- Data is replicated over two or more nodes, if one node fails, the data is still available on other nodes.
- Replication improves data availability and can also improve read performance
- Eventual consistency are used, if no updates take place for a long time all replicas will gradually and eventually become consistent.

NOSQL Characteristics related to distributed databases and distributed systems

Two major replication models Master-Slave and Master-Master replication.

- **Master- Slave replication :** Only one copy be the master or primary copy; all write operations must be applied to the master copy and then propagated to the slave copies, usually using **eventual consistency.**
- Master handles the writes and slaves synchronize with the master handle the writes.

Advantages and Disadvantages of Master-Slave Replication

Advantages:

- More read requests, add more slave nodes
- Ensure that all read requests are routed to the slaves
- Should the master fail, the slaves can still handle read requests
- Good for datasets with a read-intensive dataset

Disadvantages:

- The master is a bottleneck and limited by its ability to process updates and to pass those updates on
- Its failure does eliminate the ability to handle writes until: the master is restored or a new master is appointed.
- Inconsistency due to slow propagation of changes to the slaves
- Bad for datasets with heavy write traffic

Master - Master Replication

- Allows reads and writes at any replicas and all members are responsive to client data queries
- All the replicas have equal weight, they can all accept writes
- Replication allows writes to any node; the nodes coordinate to synchronize their copies of the data.
- The loss of any of them doesn't prevent access to the data store.

Master - Master Replication

Advantages:

- Node failures without affect access to data
- Nodes can be added easily to improve your performance

Disadvantages:

- Inconsistency!
- Slow propagation of changes to copies on different nodes
- Two people can update different copies of the same record stored on different nodes at the same time a write-write conflict
- Inconsistent writes are forever.

Sharding of Files

- In any applications files can have of millions of records and these records can be accessed by thousand of users.
- Not practical to store the whole file in one node
- Horizontal partitioning of the file records is often employed in NOSQL systems.
- This serves to distribute the load of accessing the file records to multiple nodes.
- The combination of sharding the file records and replicating the shard works in tandem to improve load balancing as well as data availability.

High - Performance Data Access

- Hashing or Range partitioning are the techniques used to obtain individual records.
- **Hashing :** A hash function h(K) is applied to the key K and the location of the object with the key K is determined by the value of h(K).
- **Range Partitioning:**The location is determined via a range of key values, i would hold the objects whose key values k in the range $Ki_{min} <= K <= Ki_{max}$

NOSQL characteristics related to data models and query languages.

- Not requiring a Schema:
  - Semi-structured and self-describing data of NOSQL systems does not requires a schema.
  - Partial schema was allowed to improve storage efficiency JSON is used in several NOSQL systems
- Less powerful Query languages:
  - NOSQL system provide a set of functions and operations as a programming API.
  - Reading and writing of data objects are accomplished by calling appropriate operations by the programmer.
  - CRUD operations : for Create, Read, Update, and Delete. SCRUD operations because of an added Search (or Find) operation
- Versioning: Some NOSQL systems provide storage of multiple versions of the data items, with the timestamps of when the data version was created.

Types of NOSQL systems

- **Document-based NOSQL systems:** System store data in the form of documents using well-known formats, such as JSON (JavaScript Object Notation). Documents are accessible via their document id, but can also be accessed rapidly using other indexes.
- **NOSQL key-value stores:** These systems have a simple data model based on fast access by the key to the value associated with the key; the value can be a record or an object or a document or even have a more complex data structure.
- **Column-based or wide column NOSQL systems**: These systems partition a table by column into column families where each column family is stored in its own files. They also allow versioning of data values.
- **Graph-based NOSQL systems:** Data is represented as graphs, and related nodes can be found by traversing the edges using path expressions

Types of NOSQL systems

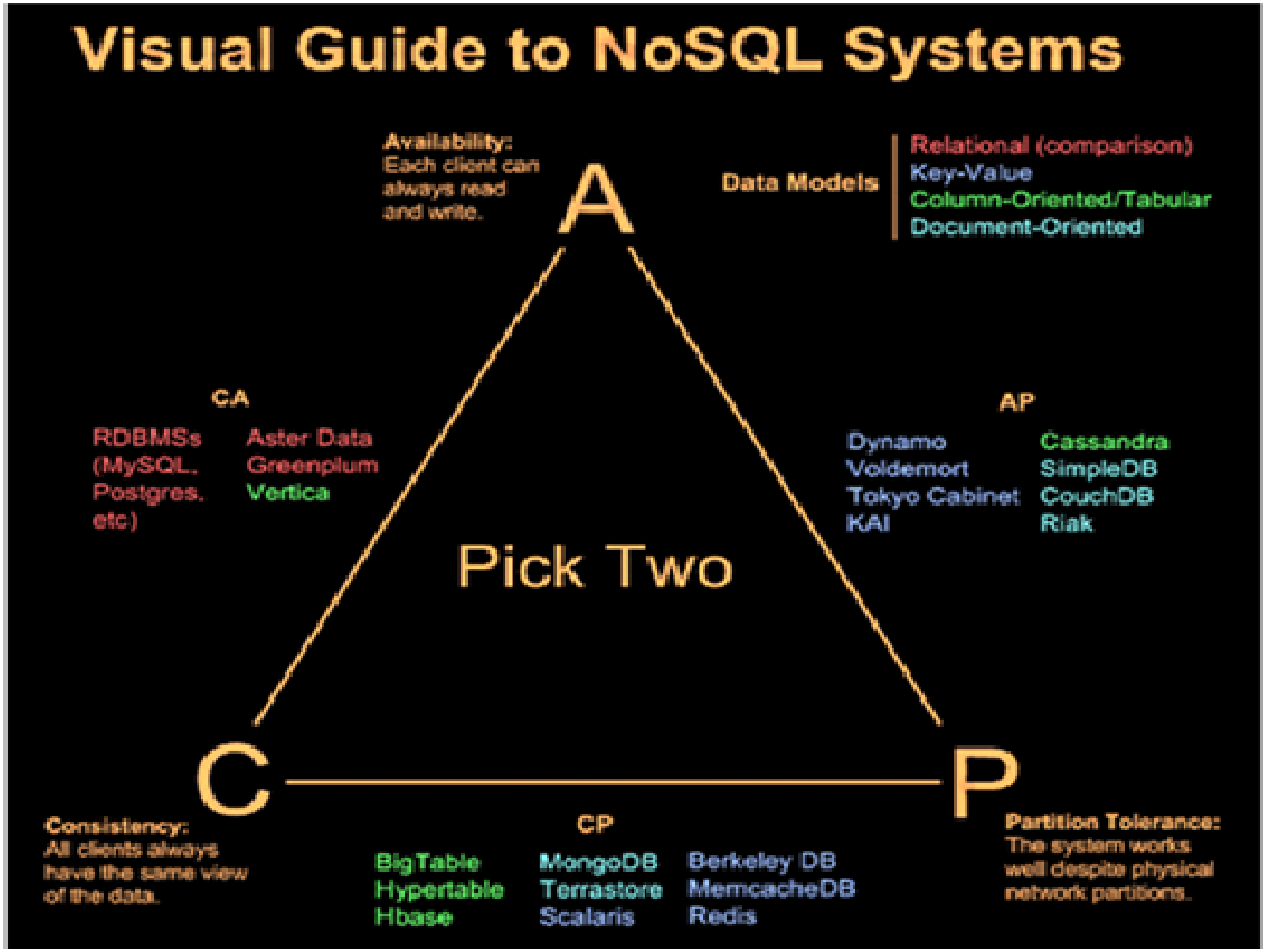
- Three letters in CAP refer to three desirable properties of distributed systems with replicated data
  - **Consistency**: The nodes will have same copies of a replicated data item visible for various transactions.
  - **Availability**: Guarantees that every request receives a response about whether it was successful or failed.
  - **Partition Tolerance**: System can continues to operate despite communication breakages that separate the cluster into partitions, where the nodes in each partition can only communicate among each other.
- The CAP theorem states it is not possible to guarantee all three of the desirable properties at the same time in a distributed system with data replication.
- In NOSQL distributed data store a weaker consistency level is acceptable with other two properties.

CAP Theorem

NoSQL databases are classified based on the two CAP characteristics they support:

- **CP database**: A CP database delivers consistency and partition tolerance at the expense of availability. When a partition occurs between any two nodes, the system has to shut down the non-consistent node until the partition is resolved.
- **AP database:** An AP database delivers availability and partition tolerance at the expense of consistency. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others.
- **CA database**: A CA database delivers consistency and availability across all nodes. It can't do this if there is a partition between any two nodes in the system, however, and therefore can't deliver fault tolerance.

Visual Guide to NOSQL Systems



Reference

Fundamentals of Database systems 7th Edition by Ramez Elmasri.