2024

Exploring CAP Theorem, BASE, and NoSQL Databases

CLASS ACTIVITY

Ahmad Abdullah i22-1609

Abdul Sami Qasim i22-1725

1. Introduction to CAP Theorem (15 minutes)

Questions to Answer:

1. **What does the CAP theorem stand for?**
   CAP refers to Consistency, Availability, and Partition Tolerance.
2. **What does each letter represent in the context of database systems?**
   - **Consistency**: Ensures that all nodes in the system reflect the same data at any given moment.
   - **Availability**: Guarantees that each request receives a reply, even if the request cannot be completed successfully.
   - **Partition Tolerance**: Ensures the system continues functioning despite network disruptions or communication breakdowns.
3. **Why is the CAP theorem important when designing distributed systems?**
   It provides a framework to understand the compromises required when balancing consistency, uptime, and fault tolerance, aiding in designing systems that meet specific needs.
4. **What is the relationship between consistency, availability, and partition tolerance?**
   The theorem asserts that a distributed system can only prioritize two out of these three characteristics during a network partition.
5. **Can a system satisfy all three conditions simultaneously? Why or why not?**
   It's not feasible because, during network partitioning, one must choose between providing consistent responses or remaining highly available.
6. **Examples of real-life systems:**
   - **Consistency & Partition Tolerance**: HBase, MongoDB
   - **Availability & Partition Tolerance**: Cassandra, DynamoDB

2. BASE Model (10 minutes)

Questions to Answer:

1. **What does BASE stand for, and how is it different from ACID?**
   - **BASE**: Refers to **Basically Available**, **Soft state**, and **Eventually consistent**. It emphasizes high availability and tolerating delays in consistency.
   - **ACID**: Focuses on maintaining strict **Atomicity**, **Consistency**, **Isolation**, and **Durability** for transactions.
2. **Key principles of BASE:**
   - **Eventual Consistency**: Guarantees that data will eventually align across all nodes.

    - **Soft State**: Allows data to change without external input due to delayed synchronization.
    - **Basic Availability**: Ensures the system responds, even if the responses aren't always accurate or immediate.

3. Types of NoSQL Databases (40 minutes)

Document-based Databases:

- **Description**: Stores information as structured or semi-structured documents, typically in formats like JSON or XML.
- **Key Characteristics**:
    - Adaptable schemas, ideal for hierarchical data storage, and supports horizontal scalability.
- **Examples**: MongoDB, CouchDB
- **Use Cases**:
    - **Problem**: Managing semi-structured data.
    - **Why it fits**: Easily modifiable schemas and excellent scalability.
    - **Challenges**: Requires careful planning for queries and indexing.

Key-Value Stores:

- **Description**: Data is stored as pairs of keys and their associated values.
- **Key Characteristics**:
    - Straightforward design, optimized for rapid read and write operations, and scales efficiently.
- **Examples**: Redis, DynamoDB
- **Use Cases**:
    - **Problem**: Session management and caching.
    - **Why it fits**: Provides low-latency and high-speed lookups.
    - **Challenges**: Limited ability to handle complex queries.

Column-family Stores:

- **Description**: Data is arranged in a table-like structure, where rows can have a variable number of columns.
- **Key Characteristics**:
    - Highly scalable and suited for large-scale analytical workloads.
- **Examples**: Cassandra, HBase

- **Use Cases**:
  - **Problem**: Handling logs or time-series data.
  - **Why it fits**: Distributed nature and optimized for high write speeds.
  - **Challenges**: Maintenance and administration can be intricate.

Graph Databases:

- **Description**: Represents data as nodes and their relationships using edges.
- **Key Characteristics**:
  - Tailored for exploring and analyzing complex relationships between data points.
- **Examples**: Neo4j, Amazon Neptune
- **Use Cases**:
  - **Problem**: Social network analysis or recommendation engines.
  - **Why it fits**: Provides efficient relationship traversal.
  - **Challenges**: May not scale efficiently for extremely large datasets.

4. Critical Thinking Questions (15 minutes)

1. **How does the CAP theorem affect the choice between different NoSQL databases?**
   The theorem influences system design by prioritizing either consistency, availability, or fault tolerance based on specific application needs. For instance, systems requiring real-time data access might favor availability and partition tolerance over strict consistency.
2. **Why do companies choose NoSQL over RDBMS?**
   NoSQL databases offer greater flexibility with schema design, improved scalability, better performance for big data, and cost efficiency, making them ideal for modern dynamic applications.
3. **Potential risks or trade-offs of using BASE instead of ACID:**
   - **Risks**: Temporary data inconsistencies and added complexity for developers to ensure eventual consistency in application logic.
   - **Trade-offs**: While BASE sacrifices immediate consistency, it provides better scalability and system availability.