Tutorial-2

Name: Nagaveni L G
Sec: 6F
SRN: PES2UG21CS315
Date: 16-04-2024

1> Comparicion b/w two differat approaches to addressing the CAP theorem trade-offs in distributed syste

Ans: Consistency oriented approach:

- this approach prioritizes consistency over availability in the face of n/w partitions.
- Systems following this approach ensures that all nodes in system have the most recent data before allowing furthur operations
- ex: MySQL, postgreSQL
- providing consistency guarantees, they might sacrifice availability during n/w partitions

Availability-Oriented Approach

- prioritizes availability over consistency.
- systems following this approach aim to remain operational & responsive even in presence of n/w partitions
- may sacrifice consistency temporarily, allowing divergent states across nodes, which are later

reconciled
- ex: Cassandra, DynamoDB

2) are there any strategies or approaches to mitigate the limitations imposed by the CAP theorem in distributed systems? if yes, describe one such strategy?

Ans: yes.

Quorum Consistency:
- technique used to achieve a balance b/w consistency, availability, & partition tolerance in distributed systems.
- involves dividing data into replicas & allowing operations to be performed on a subset of these replicas, known as quorum. By carefully configuring quorums, systems can maintain both consistency & availability even in the face of n/w partitions

3) How does CAP theorem impact the design & implementation of distributed systems?

Ans: Architectural choices: CAP theorem compels system designers to prioritize two out of there characteristics: Concietency, availability, partition tolerance,

* Architects must carefully consider the requirements of their applications & choose an appropriate trade-off strategy

Data model Selection : Choice of Data model (eg: relational, NoSQL) is influenced by CAP theorem priorities

Concurrency control mechanisms : systems emphasizing consistency use distributed locking & serializability for data consistency

Replication and synchronisation: Availability-focused systems rely on replication & synchronisation for data availability & eventual consistency

Error Handling & Recovery : are crucial for coping with n/w partitions & failures

4) Can you explain the trade-offs described by CAP theorem? Why is it impossible for a distributed data store to simultaneously gurantee all 3 properties of consistency, Availability, partition tolerance?

Ans Tradeoffs b/w Consistency, Availability, partition tolerance indistributed systems.

Consistency: Every read recieves the most recent write or an error - in other words all nodes have the same data at the same time

Availability: Every request recieves a response, without the guarantee that it contains the most recent data. system remains responsive even in face of n/w failures

partition tolerance: system continues to operate despite n/w partitions, that prevent some nodes from communicating with others.

why is it impossible?

Ans - Consistency v/s availability: ensuring both simultaneously is challenging because mainting consistency often requires coordination and synchronisation among nodes.

- for ex: in a n/w partition scenerio, maintaining consistency might require waiting for communication to be restored leading to reduced availability

N/w partitions: n/w failures are inevitable in distributed systems due to factors like latency, h/w failures or n/w congestion

Achieving all 3 properties simultaneouly is not feasible due to fundamental constraints imposed by n/w partitions & need for system responsiveness

5) Differences b/w CAP and ACID properties ?

Ans

CAP	ACID
- focusses on distributed data stores' theoretical limitations	- ensures reliability & Correctness of Transaction within a single DB
- Deals with tradeoffs b/w Consistency, Availability, partition tolerance	- Encompasses Atomicity, Consistency, Isolation, durability
- pertains specifically to distributed systems and their behaviour during n/w partitions	- ~~Applicable~~ Applicable primarily to relational DB's
- Forces designers to prioritize 2 out of 3 properties based on system requirements	- Transactions must adhere to all 4 properties without compromise.