NoSQL

Principles (Consistency)

Consistency

- Biggest change from centralized relational databases to cluster-oriented noSQL databases
  - RDBMS: strong consistency
    - be consistent
  - noSQL: relaxed/eventual consistency
    - everything is going to be alright, not now but eventually

Conflicts

- Read-read (simply read) conflict
  - two users see different data at the same time
- Write-write conflict
  - two users updating the same data at the same time
- Read-write conflict
  - write & read requests on the same aggregate are made concurrently
  - when not treated, this will cause inconsistency window
  - propagation of changes to all replicas might take some time
    - until this process finishes, inconsistent read might happen

Solutions

- two approaches
  - pessimistic
    - preventing conflict from occurring
    - techniques: locks
  - optimistic
    - conflicts may occur, but detect and solve them later
    - techniques: version stamps

Forms of Consistency

- Strong or immediate
  - ACID transactions
- Eventual
  - you may have replication inconsistency but eventually all nodes will be updated to the same value and becomes consistent

ACID Properties

- Atomicity
  - partial execution of a transaction is not possible (all or nothing)
- Consistency
  - transactions bring the DB from consistent state (valid) to another consistent state
- Isolation
  - transactions executed in parallel don’t see the uncommitted state of each other
  - reading & writing can happen concurrently (parallel)
    - failing operation should not affect others
  - ensures to leave the DB as if the transaction were executed in sequence
- Durability
  - if the transaction is committed, it should stay at that state even if a failure happens

CAP theorem

- Relax consistency
- CAP properties = properties of distributed system
  - **C**onsistency
  - **A**vailability
  - **P**artition tolerance
-

CAP Properties

- Consistency:
  - write and read operation must be atomic
    - runs on the cluster as they were running on a single node
  - The write operation must be atomic
    - which means changes must be propagated to all replicas
  - After a write operation, all readers must see the same data
    - Since any node can be used for handling the read request

CAP Properties

- Availability:
  - If node is working, then it must respond to user requests
    - every request must result in a response
    - no guarantee that the response contains the most recent write
- Partition tolerance
  - system continues to operate even if couple of nodes gets isolated
  - network failure should not result in a shutdown for the whole system

CAP theorem

- CAP theorem

It is not possible to have a distributed system that would guarantee consistency, availability, and partition tolerance at the same time.

Only 2 of these 3 properties can be enforced.

CAP theorem consequences

- at most two properties can be guaranteed
  - CA: Consistency + Availability
  - CP: Consistency + Partition tolerance
  - AP: Availability + Partition tolerance

It is not possible to have a distributed system that would guarantee

consistency, availability, and partition tolerance at the same time.

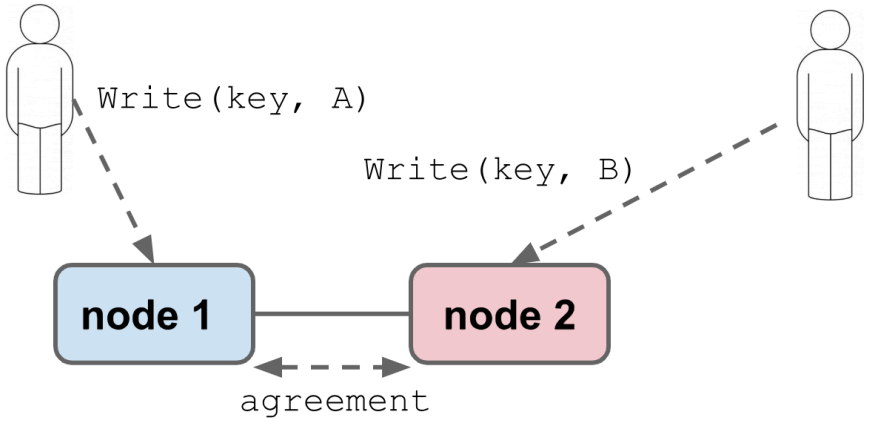
Only 2 of these 3 properties can be enforced.

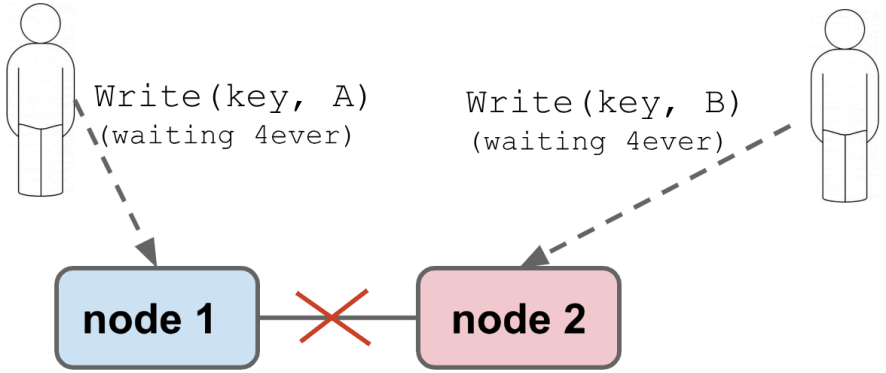
CAP: Real Applications

- Single server system is always CA
- A distributed system has to be tolerant of network partitions
  - it is difficult to detect all network failures
  - does that mean only CP and AP are possible?
  - Partition tolerance is a must
    - but tradeoff between Consistency & Availability

Partition tolerance & consistency

- Example: two users, two nodes , two write attempts
- before the write is committed, both nodes should agree (strong consistency)
- if nodes are partitioned, then we are loosing availability





CAP: Real Applications

- Single server system is always CA
- A distributed system has to be tolerant of network partitions
    - tradeoff between Consistency & Availability
    - give up some consistency to achieve availability
- Issue
    - how much tolerant ?
    - approach BASE

BASE Properties

- New concept (NoSQL)
  - **B**asically **A**vailable
    - basically the system works all the time
    - partial failure can occur, without causing total system failure
    - spread data on the cluster with enough replication factor
  - **S**oft state
    - changes occur all the time
  - **E**ventual consistency
    - sooner or later the system will be at consistent state
    - at some point the system will converge to a consistent state
      - different from the immediate consistency in ACID

Strong consistency?

- How many nodes must be involved in order to get strong consistency ?
  - If all nodes holding the replicas, then this is a strong consistency

Quorums

- How many nodes must be involved in order to get strong consistency ?
- instead of all, use something in the middle (quorum)
  - how many nodes you need to contact to make sure that you have the most up-to-date change?
- if N is the replication factor, W number of confirmed write , R number of reads
- write quorum : $W > N/2$
  - W number of nodes participating in the write
  - N number of nodes in replication
- read quorum: $R > N-W$
  - R number of nodes participating in the read
- a write or a read operation has to obtain a write quorum or a read quorum

Quorums & Consistency level

- R (read quorum), W (write quorum), N (all nodes in replication)
- Consistency level
  - strong: $W = N$
  - eventual $W + R < N$
  - write quorum: $W > N/2$

Summary

- a wide range of options to be considered
  - scalability
    - how does the system scale?
  - availability
    - when nodes refused to accept requests?
  - consistency
    - what is the level of consistency?
  - latency
    - delay in handling user requests?
  - durability
    - are the committed data written reliably?
  - resilience
    - can the data be recovered in case of failure?

Summary

- Many noSQL BDs are aggregate-oriented data modeling
- Conflicts:
  - (write-write conflict)
  - (read-write conflict)
- Relaxing consistency
  - CAP
  - Quorum (read quorum , write quorum)

Summary

- Different noSQL is applying different consistency levels
- It is good to know the options and choose the right one based on the circumstances
  - CA (Google BigTable)
    - if network error happens, all nodes stop receiving requests
  - AP
    - Apache Cassandra , Apache couchDB