NoSQL Intro + KV DBs

Monday, February 03, 2025 9:25 AM

Distributed DBs and ACID – **Pessimistic Concurrency**

- ACID transactions
- Focuses on "data safety"
- considered a ***pessimistic concurrency model*** because it assumes one transaction has to protect itself from other transactions
  - IOW, it assumes that if something can go wrong, it will.
- Conflicts are prevented by locking resources until a transaction is complete (there are both read and write locks)
- Write Lock Analogy → e.g. borrowing a book from a library… If you have it, no one else can.

See https://www.freecodecamp.org/news/how-databases-guarantee-isolation for more for a deeper dive

Optimistic Concurrency

- Transactions do not obtain locks on data when they read or write
- Optimistic because it assumes conflicts are unlikely to occur
  - Even if there is a conflict, everything will still be OK
- But how?
  - Add last update timestamp and version number columns to very table.. Read them when changing. THEN, check at the end of transaction to see if any other transaction has caused them to be modified.

- **Low conflict systems** (backups, analytical dbs, etc.)
  - Read heavy systems
  - The conflicts that arise can be handled by rolling back and re-running a transaction that notices a conflict
  - So optimistic concurrency works well – allows for high concurrency
- **High Conflict Systems**
  - Rolling back and rerunning transactions that encounter a conflict -> less efficient
  - So, a locking scheme (pessimistic model) might be preferable

BASE – ACID Alternative

- Basically Available
  - Guarantees the availability of the data (per CAP), but response can be "failure"/"unreliable" because the data is in an inconsistent or changing state
  - System appears to work most of the time
- Soft State - The state of the system could change over time, even w/o input. Changes could be result of eventual consistency.
  - Data stores don't have to be write-consistent
  - Replicas don't have to be mutually consistent
- Eventual Consistency - The system will eventually become consistent
  - All writes will eventually stop so all nodes/replicas can be updated

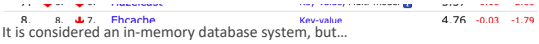
NoSQL

- Generally can refer to non-relational databases or ones that don't use SQL

- **Key-value** - designed around...
  - *Simplicity*
    - Data model is extremely simple
    - comparatively, tables in a RDBMS are very complex.
    - lends itself to simple CRUD ops and API creation
  - *Speed*
    - Usually deployed as in-memory
    - retrieving a value given its key is typically a O(1) op b/c hash tables or similar data structs used under the hood
    - no concept of complex queries or joins… they slow things down
  - Scalability
    - Horizontal Scaling is simple - add more nodes
    - Typically concerned with eventual consistency, meaning in a distributed environment, the only guarantee is that all nodes will eventually converge on the same value
- Use cases:
  - EDA/Experimentation Results Store
    - store intermediate results from data preprocessing and EDA
    - store experiment or testing (A/B) results w/o prod DB
  - Feature Store
    - store frequently accessed feature → low-latency retrieval for model training and prediction
  - Model Monitoring
    - store key metrics about performance of model, for example, in real-time inferencing.
  - Storing Session Information
    - everything about the current session can be stored via a single PUT or POST and retrieved with a single GET …. VERY Fast
  - User Profiles & Preferences
    - User info could be obtained with a single GET operation… language, TZ, product or UI preferences
  - Shopping Cart Data
    - Cart data is tied to the user
    - needs to be available across browsers, machines, sessions
  - Caching Layer:
    - In front of a disk-based database

- **Redis DB** (Remote Directory Server)
  - Open source, in-memory database
  - Sometimes called a data structure store
  - Primarily a KV store, but can be used with other models: Graph, Spatial, Full Text Search, Vector, Time Series

Rank Oct 2024	Rank Sep 2024	Rank Oct 2023	DBMS	Database Model	Score Oct 2024	Score Sep 2024	Score Oct 2023
1.	1.	1.	Redis	Key-value, Multi-model	149.63	+0.20	-13.33
2.	2.	2.	Amazon DynamoDB	Multi-model	71.85	+1.78	-9.07
3.	3.	3.	Microsoft Azure Cosmos DB	Multi-model	24.50	-0.47	-9.80
4.	4.	4.	Memcached	Key-value	17.79	+0.95	-3.05
5.	5.	5.	etcd	Key-value	7.17	+0.12	-1.57
6.	↑ 7.	↑ 8.	Aerospike	Multi-model	5.57	+0.41	-0.86
7.	↓ 6.	↓ 6.	Hazelcast	Key-value, Multi-model	5.57	-0.16	-2.60



- It is considered an in-memory database system, but…
  - Supports durability of data by:
  a. essentially saving snapshots to disk at specific intervals or
  b. append-only file which is a journal of changes that can be used for roll-forward if there is a failure
- Originally developed in 2009 in C++
- Can be very fast … > 100,000 SET ops / second
- Rich collection of commands
- Does NOT handle complex data. No secondary indexes. Only supports lookup by Key.
  Keys:
  - Usually strings but can be any binary sequence
- Values:
  - Strings
  - Lists (linked lists)
  - Sets (unique unsorted string elements)
  - Sorted sets
  - Hashes (String -> String)
  - Geospatial data