DB HW Dec 17

Part 1 AWS

1. Memcache

What is Memcache?

Memcached is an open source, high-performance, distributed memory caching system intended to speed up dynamic web applications by reducing the database load. It is a key-value dictionary of strings, objects, etc., stored in the memory, resulting from database calls, API calls, or page rendering.

History of Memcache?

Memcached was created in 2003, by Brad Fitzpatrick for his LiveJournal website. It was initially developed in Perl and then translated into C. It is used by some of the biggest companies out there such as Facebook, Youtube and Twitter.

Components of Memcached?

Memcached is made up of 4 main components:

- Client Software: It is used to give a list of available Memcached servers.
- A Client-based hashing algorithm: It chooses a server based on the key.
- Server Software: It is used to store values and their keys into an internal hash table.
- LRU: LRU stands for Least Recently Used. This determines when to throw out old data or reuse memory.

Key Features?

- It is open source.
- Memcached server is a big hash table.
- It significantly reduces the database load
- It is perfectly efficient for websites with high database load.
- It is distributed under Berkeley Software Distribution (BSD) license.
- It is a client-server application over TCP or UDP.
- It is very scalable; just add boxes with memory to spare.
- Memcached runs as a standalone service. So, if you take your application down, the cached data will remain in memory as long as the service runs.
- The cache nodes are very ignorant: which means they have no knowledge about other nodes participating. This handles the management and configuration of such a system extremely easy.

Memcached is not

- a persistent data store
- a database
- application-specific
- a large object cache
- fault-tolerant or highly available

Disadvantages of Memcached

- It is not a fault- tolerant tool.
- Compared to in-memory cache, it is very slow, mostly because of serialization or deserialization and network latency.
- The cache nodes are very ignorant: For example, there is no way to iterate over all of the cached items.
- It is not a persistent datastore.

How does Memcached Work?

At a high-level Memcached works as follows:

- The client requests a piece of data then Memcached checks to see if it is stored in cache.
- There may a possibility of two possible outcomes:
    1. If the data is stored in cache: return the data from Memcached (there is no need to check the database).
    2. The data is not stored in cache: query the database, retrieve the data, and subsequently store it in Memcached.
- Whenever information is modified or the expiry value of an item has expired, Memcached update its cache to ensure fresh content is delivered to the client.

This setup has various Memcached servers and many clients. Clients use a hasing algorithm to determine memcahed storage server for use. This helps to distribute the load.

And then server computes a second hash of the key in order to determine where it should store the corresponding value in an internal hash table. Some important things about Memcached architecture are:

- Data is only sent to one server
- Servers don't share data
- Servers keep the values in RAM - if RAM runs out the oldest value is discarded.

2. Redis vs. Memcache

What is Redis

Redis, which means Remote Dictionary Server, was created in 2009 by Salvatore Sanfilippo, to improve the scalability of the web log analyzer that his Italian startup was building. The first prototype was written in Tcl and later transcripted to C. When Sanfilippo decided to open source the project it then started to get some traction. Giants like GitHub and Instagram were some of the first companies to adopt it.

What is Memcached

Memcached was created a bit earlier, in 2003, by Brad Fitzpatrick for his LiveJournal website. It was initially developed in Perl and then translated into C. It is used by some of the biggest companies out there such as Facebook, Youtube and Twitter.

Data storage: Redis vs Memcached

- How Redis stores data

  Redis has five data types:

  - String: a text value
  - Hash: A hash table of string keys and values
  - List: A list of string values
  - Set: A non-repeating list of string values
  - Sorted Set: A non-repeating list of string values ordered by a score value

  Redis supports data type operations. This means you can access or change parts of a data object without having to load the entire object to an applicational level, modify it and then re-store the updated version. Redis uses an encapsulated version of the malloc/free memory management, being a simpler approach compared to the Memcached Slab mechanism. Redis supports keys with a maximum size of 512MB and also values up to 512MB. This limit is per element on aggregate data types (Lists and Sets).

- How Memcached stores data

  Unlike Redis, Memcached has no data types, as it stores strings indexed by a string key. When compared to Redis, it uses less overhead memory. Also, it is limited by the amount of memory of its machine and, if full, it will start to purge values on a least recently used order. It uses an allocation mechanism called Slab, which segments the allocated memory into chunks of different sizes, and stores key-value data records of the corresponding size. This solves the memory

fragmentation problem. Memcached supports keys with a maximum size of 250B and values up to 1MB.

How Redis and Memcached scale

- How Redis scales

  Since Redis is predominantly single-threaded and has native support for clustering, it grows well horizontally.

  Redis clustering works with a master/slave architecture. For every master node there are two slave nodes for redundancy, therefore, if the master fails, then the system automatically promotes one of the slaves as the new master. This kind of scalability comes with the disadvantage of upkeep complexity. It's harder to maintain several nodes running synchronously than a single one.

- How Memcached scales

  Memcached is easily scaled vertically, as it is multithreaded. The only requirements are to give it more cores and more memory. It can also be scaled horizontally, on the client side, by the implementation of a distributed algorithm. This comes with the disadvantage of being more complex to implement while Redis has it out of the box.

Data Persistence

Redis is an in-memory (mostly) data store and it is not volatile, Memcached is an in-memory cache and it is volatile.

Also Memcached is limited to the LRU (least recently used) eviction policy whilst Redis supports six different policies.

Part 2 SQL vs. No-SQL

1. Vertical scaling vs. Horizontal Scaling

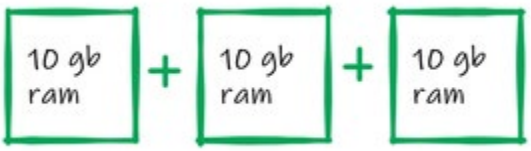
What Is Scalability?

Scalability is the ability to expand from the existing configuration of the system for handling the increasing amount of load. Scaling can be done by either adding extra hardware or by upgrading the current system configuration. There are two different ways to accomplish scaling, one is vertical scaling, and the other is horizontal scaling.

The core of the difference is the approach of adding resources to the infrastructure.

Horizontal scaling

Horizontal scaling means we scale by adding additional machines to our existing bunch of resources.

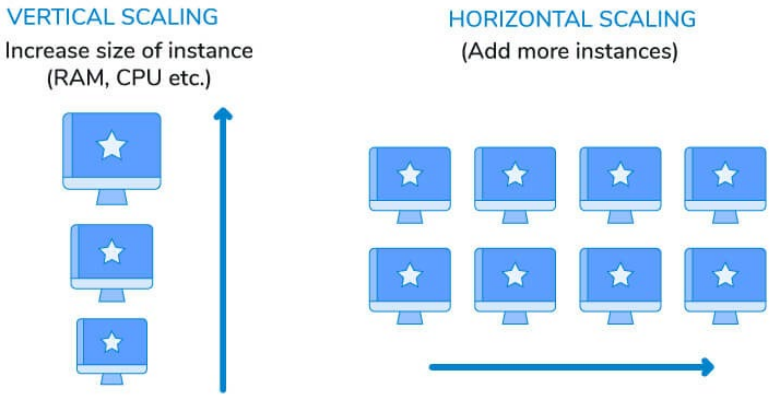


Vertical scaling

Vertical scaling means we scale by adding more computing power like CPU and RAM to an existing machine.

Horizontal Scaling vs. Vertical Sclaing

Horizontal scaling	Vertical scaling
Horizontal scaling is difficult to implement.	Vertical scaling easy to implement.
In Horizontal scaling the databases at each node or site only contains part of the data.	Vertical scaling means we scale by adding more computing power like CPU and RAM to an existing machine.
Scaling can be done with less downtime.	Vertical scaling involve more downtime.
More concurrency.	Less concurrency when compared to Horizontal scaling.
Data sharing is complex.	Data sharing is easy.
Horizontal scaling is more reliable.	Less reliable when compared to Horizontal scaling.
Horizontal scaling is very costly.	Vertical scaling is cheaper.
It is also known as scale out.	It is also known as scale up.
More easy to upgrade in the future.	Upgradation in future is not so easy.
Consumes more power.	Consume less power.

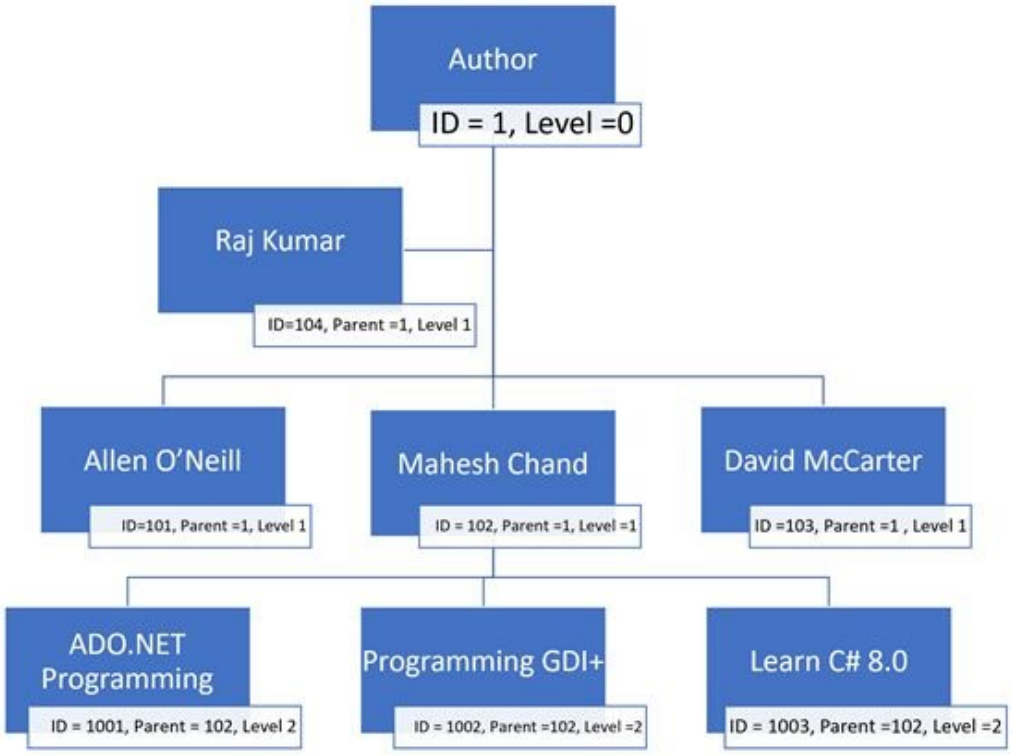


2. Hierarchical datastore

What is Hierarchical Database?

A hierarchical database is DBMS that represent data in a tree-like form. The relationship between records is one-to-many. That means, one parent node can have many child nodes. A hierarchical database model is a data model where data is stored as records but linked in a tree-like structure with the help of a parent and level. Each record has only one parent. The first record of the data model is a root record.

The following tree diagram represents above tabular data into a hierarchical form with their parent and level.



Pros and Cons of hierarchical databases

Hierarchical databases are useful when you need to represent data in a tree like hierarchy. The perfect example of a hierarchical data model is the navigation file or sitemap of a Website. A company organization chart is another example of a hierarchical database.

Advantages of hierarchical databases

- Traversing through a tree structure is very simple and fast due to its one-to-many relationships format. Major several programming languages provide functionality to read tree structure databases.

- Easy to understand due to its one-to-many relationships.

Disadvantages of hierarchical databases

- It's rigid format of one-to-many relationships. That means, it doesn't allow more than one parent of a child.
- Multiple nodes with same parent will add redundant data.
- Moving one record from one level to other level could be challenging.

3. BASE Principle (for No-SQL DB)

BASE consists of three principles:

- Basic Availability. The NoSQL database approach focuses on the availability of data even in the presence of multiple failures. It achieves this by using a highly distributed approach to database management. Instead of maintaining a single large data store and focusing on the fault tolerance of that store, NoSQL databases spread data across many storage systems with a high degree of replication. In the unlikely event that a failure disrupts access to a segment of data, this does not necessarily result in a complete database outage.
- Soft State. BASE databases abandon the consistency requirements of the ACID model pretty much completely. One of the basic concepts behind BASE is that data consistency is the developer's problem and should not be handled by the database.
- Eventual Consistency. The only requirement that NoSQL databases have regarding consistency is to require that at some point in the future, data will converge to a consistent state. No guarantees are made, however, about when this will occur. That is a complete departure from the immediate consistency requirement of ACID that prohibits a transaction from executing until the prior transaction has completed and the database has converged to a consistent state.

Part 3 SQL Tuning

1. View and Stored Procedure

View

A view is a virtual table that consists of columns from one or more tables. Though it is similar to a table, it is stored in the database. It is a query stored as an object. A view is an object that derives its data from one or more tables. These tables are referred to as base or underlying tables. Once you have defined a view, you can reference it like any other table in a database.

Advantages of Views

- A view consists of a SELECT statement that stored with a database. Because views are stored as part of the database, they can be managed independently of the applications that use them.
- A view behaves like a virtual table. Since you can code a view name anywhere you can code a table name. a view is sometimes called a viewed table.
- Views can be used to restrict the data that a user is allowed to access or to present data in a form that is easy for the user to understand. In some database users may be allowed to access data only through views.

Stored Procedure

A stored procedure is a set of one or more SQL statements that are stored together in database. To create a stored procedure use CREATE PROCEDURE statement. To use the stored procedure you send a request for it to be executed. When server receives the request, it executes the stored procedure.

Advantages of Stored Procedure

- A stored procedure is one or more SQL statements that have been compiled and stored with database. A stored procedure can be started by application code on the client.
- Stored procedure can improve database performance because the SQL statements in each procedure are only compiled and optimized the first time they are executed. In contrast SQL statements that are sent from a client to the server have to be compiled and optimized every time they are executed.
- In addition to SELECT statement, a stored procedure can contain othe SQL statements such as INSERT, UPDATE, DELETE. It also contains control-of-flow language.
- A trigger is a special type of procedure that executes when rows are inserted, updated or deleted from table.
- A user defined function(UDF) is a special type of procedure that can return a value or a table.