Individual Study : 2110391

Large-Scale Web Application and Distributed System

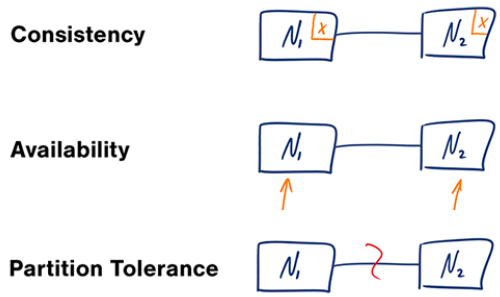
Kosate Limpongsa 5731012721

Introduction

The main objective of study is designing the architect for large scale website. How we construct the system for support high traffic user visiting. What is the main factor for website in difference environment. How do we handle the server in special case.

CAP Theorem

CAP theorem is states that it is impossible for a distributed computer system to simultaneously provide all three following guarantee (Most possible is picking two). (C) Consistency — A read is guaranteed to return the most recent write, (A) Availability — Every request gets a response in reasonable time, (P) Partition Tolerance — System continues function when network partition occur.



In CAP theorem, you can only picks two our of three states. For example, given that networks aren't completely reliable, if you consider to design network with time out or error response to client but every request must given the recent version, now you got CP. But instead of error response, by given currently available content in server, now you got AP.

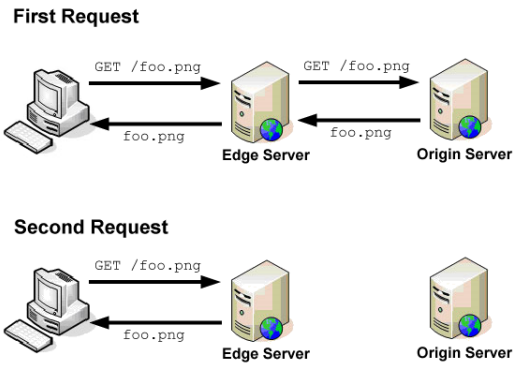
In summarize, CAP theorem let you building system with many advantages, but also adds complexity. You may choosing the right path for your application, get right from the beginning before you first deployment.

Content Delivery Network

Content Delivery Network (CDN) is a globally distributed network which serve the content to end-users by using multiple server.

Most advantage of CDN is reducing the round-trip time by distributing the content to multiple regional data centers. When user request the content, it's response with content in nearest data center depends on geolocation of user.

The most related technique were caching, CDN stores a cached version of its content in multiple geographical locations (a.k.a. PoPs) providing superior coverage to your users, when user request the file, the browser makes a DNS request to DNS server and lookup geographical based on IP address and request to target site. It's can be more efficiently, by caching the content in each PoPs along the request routes. One of handy method is storing the caches in proxy servers and response content inside the caching server instead of original server.



One of missions of CDN is minimize the latency. From architecture standpoint, CDN is high performance to build optimal connectivity and also high reliability with auto-routing approach when server was drowned. It's also scalability to handle any amount of traffic, that is the reason why CDN can protect website

from DDoS attack, its were used in most CDN provider in nowadays, Cloudflare in the example.

Distributed Database Management System

Distributed database is a database in which storage devices are not attached to processor. It may be stored in multiple computer, located in the same physical location, or maybe dispersed over a network.

Design Strategies

Distributed database can be storing separate copies of the database in multiple sites. It is also be a popular fault tolerance technique for distributed databases.

Data replication is a good way for reliability and reducing data response time. But it's require many storage and increase cost and complexity of data updating.

Fragmentation is another method that contrast from data replication. It is the task of dividing a table into a set of smaller tables. The subsets of the table are called fragments. Fragmentation can be of horizontal, vertical, and hybrid.

The horizontal fragment is method that divided the table in each row. It's can may be divided by using id range of primary key in the table. Also same ideas as vertical fragment, vertical fragment divided with attribute in table (a.k.a. table column), while hybrid are using both technique.

The advantages of fragmentation is local query optimization and less storage but it's lack of back-up copies data in different sites.

ACID

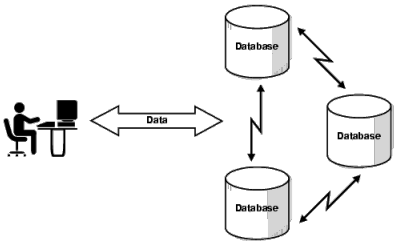
ACID is a set of properties of database transactions. A transaction in database system must maintain (A) Atomicity — Transaction must be teated as an atomic unit, either all operations are executed or none. The must be no state in a database where a transaction is partially completed. (C) Consistency — Ensures that any transaction will bring the database from one valid state to another, (I) Isolation — A transaction in process and not yet committed must remain isolated from any other transaction. (D) Durability — Committed data must saved, it will remain so, even in the event of failure, or power loss.

Base

Base is used to describe the properties in certain databases, usually NoSQL. It's referrer to as the opposite of ACID. (BA) Basically available — The contain states that the system does guarantee availability of data, (S) Soft state — Stores don't have to be write-consistent, but can change state to consistent in late time with (E) Eventual consistency property.

Transparency

Distribution transparency is method for hiding internal information and distribution process from users.



There are 3 methods of distribution transparency 1. Location transparency — the address of the remote sites and the access mechanism are completely hidden. 2. Fragmentation Transparency — Some query are combining of table fragmentation It also conceals the fact that the fragments are located at diverse sites. 3. Replication Transparency — Users only see the single table but actually its replicate to multiple sites.

Query Optimization

Query language can be represent as relational algebra definition. Using some algebra operations (selection, union, intersection, etc.) and rearrange order and different operation that process to same result but more efficient way. Then, adopt to the original query language from algebra operations.

Concurrency Control

Transaction is a program including a collection of database operations, It's may be process in the same time, there probably has some conflict situation during execution. The solution is serializability the transaction or re-schedules to avoid the conflicting.

Another method is using locking based concurrency protocols. The meaning of lock is, when data is locked, all operation performed on that data will be waiting for process.

Locking-based concurrency control can use either one-phase or two-phase locking protocols.

One-phase locking, each transaction locks the data before use and releases the lock as soon as it finished.

Two-phase, It's divided to two path. First, growing phase. expanding the lock to data in transaction and released the lock in phase two called shrinking phase.

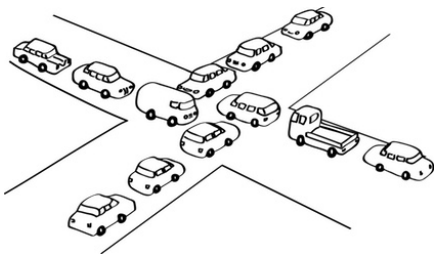
Transaction Operations & States

Having serialization and locking protocols is not enough for transaction operation. All transactions operation can be divided to three task: read data item, modify the data item, and write the data item.

If we look further to low level operation in transaction the single transaction consists of 1. beginning of transaction, 2. read or write operation, 3. ending of transaction, 4. the commit part of transaction (This operation is not the same as ending, this state happened when you confirmed this operation is completely finished) 5. rollback (This method use when you try to abort the current transaction)

Deadlock Handling

Deadlock is a state of database having two or more transactions and each transaction is waiting for a data item that being locked by some other transaction.



For example, T1 waiting for T2, T2 waiting for T3, and T3 waiting for T1.

There are prevention approach does not allow any transaction to acquire locks that will lead to deadlocks situation.

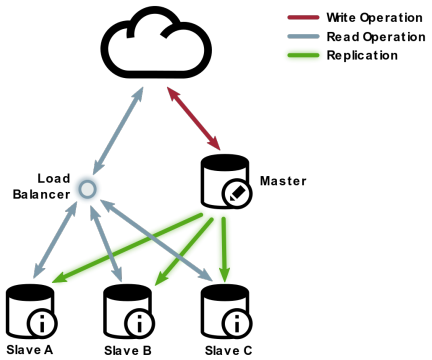
And avoidance approach handles dead-locked before they occur. Two algorithms for this purpose, namely wait-die and wound-wait.

Wait-die will abort the transaction which is younger and allowed to wait for older transaction, while Wound-wait is opposite to Wait-die.

Replication Control Algorithm

While transaction distribute to multi sites for replication the data, consistency are require. There are some of the replication algorithms of synchronous data in every sites.

First, master-slave approach, consists of master site and N slave sites. Master site will detect conflicts and manipulate transaction all the slave sites.



Master-Slave Replication

Next, voting algorithm, this approach is similar to previous one, but the different is every slave site must voting for acceptation of executing the transaction request. Every slave sites need to voting OK for perform the transaction.

If one of any slave vote NOT OK. Then, the master will transmit the reject request to all slave to telling them not update the transaction. The approach will makes the data consistency.

Another approach called consensus algorithm. This method is similar to voting algorithm but changing the condition from "one of any" to "most of any". If the result of slave sites reject more than the accept. The master site

will sent reject request to all slave sites to cancel instruction operation.

Failure & Committing

Distributed database is susceptible to a number of failures. Some failure may causes to memory loss or even the bad situation case.

The damage from data failure can divided two strategies, soft failure — when the operation system is failure or some main memory crash or power failure, hard failure — Happened when power failure, some corruption is the disk, or some damaging that happened to non-volatile memory like hard disk.

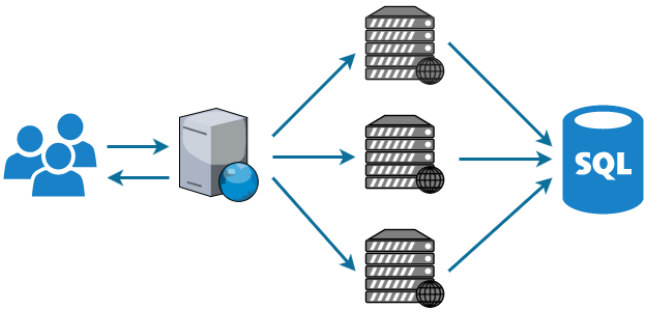
Commit protocols are designed for preventing these problem. Commit protocols are scenario using either transaction undo (rollback) or transaction redo (roll forward). It's come up with commit point, the point of time at which decision is made whether to commit or about a transaction, usually use when database is consistent, for performing the undo and redo operation.

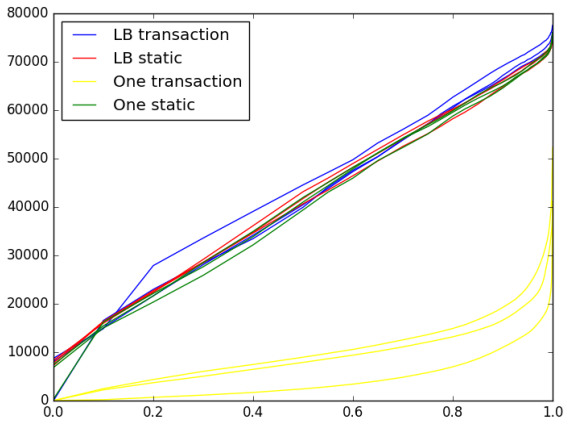
Commit point are very useful for hard failure. You can recover all data and revert to consistent state without damage any previous data. By the way, most of operation about commit protocols are done with nonvolatile storage reading and writing.

Distributing Experiment

I'm interested in load balancing approach, and try to implemented its with Nginx. The process of setup the load balancing mostly spend on configuration in Nginx and set up environment for each server.

I using Digital Ocean for run this experiment, created 1 server as load balancing server, 3 server for serving the content with dynamic content, also connected to 1 database server using MySQL as store collecting.





After experiment, I just plot a graph with wrk2 (I ever use TPC-E but it's too hard for distributing benchmark), but I just figure it out, it's measure latency time of packet response. So the graph between single server and load balancing may look not different but if it's measure by the throughput may be given the better result.

I didn't capture the screen but when you run the load balancing the graph from CPU from server will reach to 80 - 90% while benchmark with wrk2. But for single server setup only the server which receive the request, the CPU graph are reach to 100%.

See my source code : https://github.com/neungkl/web-distributing-individual-study