HLD 101

Tuesday, 1 September 2020 5:56 PM

LLD	Concurrency	HLD
- modular - extensible - readable/ Sharable - maintainable	-parallelism - threads - utilize multiple cores - trade off b/w serial & parallel	How to build scalable systems → concurrent users → QPS → data storage + structure the data so that queries are performant. → Consistent data + common tradeoffs. → Available → Partitioned/sharded.
code structure	faster on a single machine	

⊙ We are going to look at how to build such large scale systems.

+

problems around building such systems

+

several examples on how to actually build such systems.

- facebook messanger
- twitter
- Google search
- Netflix

Trade offs.

- ⊙ Not language specific
- ⊙ Schema design is an integral part. so lots of LLD
- ⊙ Designing APIs

---

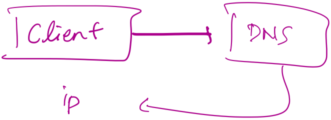
www.google.com
CLIENT
• Browser/web app
• Mobile
• SDK consumer

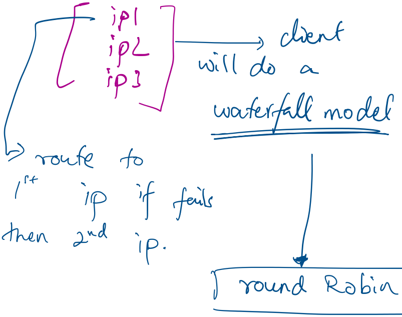
⊙ to reach to any asset/resource on internet you need an ip address.

→ DNS is just a mapping of ip with domain names

→ Client hits the DNS to fetch ip or list of ips.

⊙ DNS is also a system whose ip is known to the client

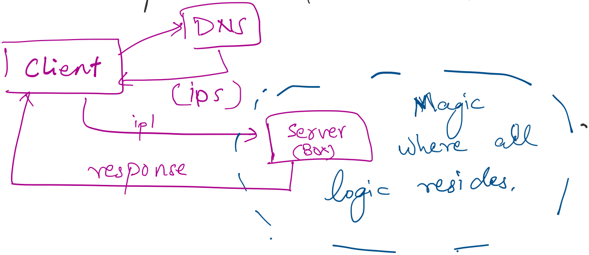




→ route to 1st ip if fails then 2nd ip.

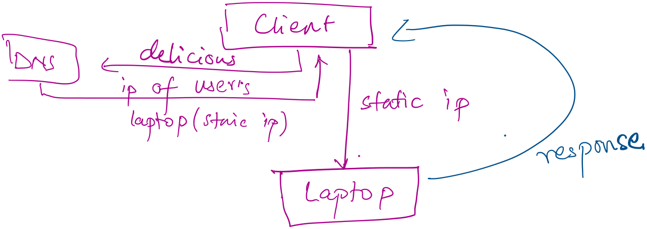
- ⊙ try things 1 after other
- ⊙ try 1st ip if fails then try 2nd ip.

if the request to ipl succeeds,



Delicious was a bookmarking service

- go to a computer and since it is a shared resource, users wanted to remember a url.
- login and save bookmarks (2005/2006)
- How did he build a system



Data was persisted on user's laptop's Disk.

- Bookmarks will be read from

secoundary storage (Disk) to primary storage (RAM)

- filter the bookmarks by userId.
- 1 RAM, processor, HDD on a single laptop
- fine for 100 users

started of with 100 user with 10x growth every month

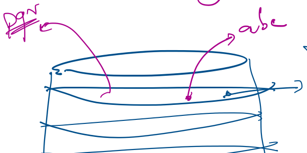
1st — 100
2nd — 1K
3rd — 10K
4th — 100K
5th — 1M

PROBLEMS

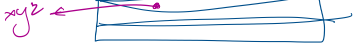
① **Processor** : concurrent users increase. handling ## requests will choke the processor.

② **Primary Storage** : RAM run out of space.

③ **Secondary Storage** : ⊙ HDD running out of space
⊙ lack of efficiency.



Data is stored on diff sectors in a 3D structure.



Ex fetch data for "xyz" → will go to a specific section on the HDD and then read its data

Next comes request for "abc", the pointer will physically move from xyz → abc.

④ Number of open internet connections. Either TCP/UDP. There is a limit on # of connections you can use.

⑤ Network Bandwidth

⑥ Laptop's battery went off.

---

static ips Vs Dynamic IPs

↳ lease line given by isp to only you. Available only to you throughout the world.

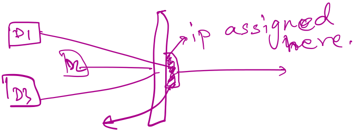
- My devices have a dynamic ip which are bound to change. To check do a ifconfig on terminal. Turn off router and repeat, you may get a totally different ip. Ip also changes across locations.

Assigned by DHCP.

- Billion of devices and not everyone is using internet at the same time.
- Not every device is a server.

A server takes a request, do some computatn & responds.

- ISP also give same ips to all devices within the same network.



ip assigned here.

maintains a mapping

⊙ Done to reduce time & compute.

→ Coming back to the same problem

How do we scale = ??

⊙ Keep adding better hardware.

- RAM
- HDD
- processors

1980-1990

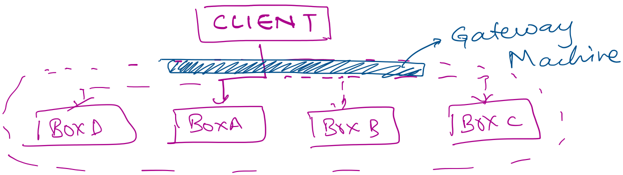
⊙ Replace old machine with new ones

⊙ Cons

→ Costly

- → wasting old boxes
- → down time
- → plan in advance/predict demand & growth.

HORIZONTAL SCALING



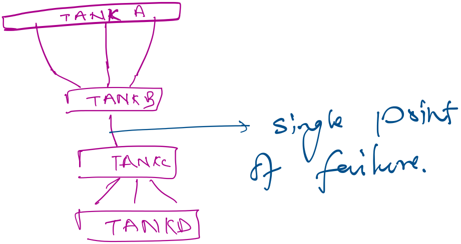
→ Changes due to this approach

⊙ How to route different machines?

– DNS will resolve to gateway machine.

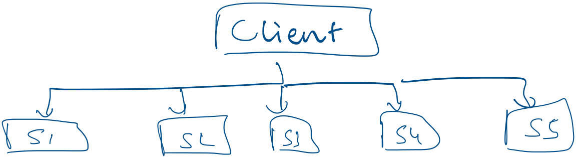
– Gateway will decide which server should cater to the request

* Single point of failure.



- We are making this as a consious choice. ∵ it hides complexity of internal system.
- Reduced the work done so that it is easily replacable.

- connection has an ip
- server has a mac address
- we can tranfer the same ip from server 1 to server 2.



- torrent
- peer to peer sharing

Vertical	Horizontal
Cons	**Pros**
- costly + wastage	- less costly + waste
- demand predictⁿ	- fluid to demand
	DAY demand ↑
	Night demand ↓

Cons

- Much complex to handle.
  - inter box commun
  - network latency
- Consistency.

---

properties of load Balancer

- divide traffic optimally
- app server boxes going down / coming up
  - ↳ load changes
  - ↳ physical outages

- Software loadbalancer
- Hardware loadbalancer.
  - ↳ less storage
  - ↳ huge RAM
  - ↳ huge simultaneous connectⁿ

[Runs on commodity hardware]

DNS allows list of ips

DNS : [LB1, LB2, LB3, - -, LBN]

QPS on Google:

populatⁿ : 8 Billion

internet : 50% of populⁿ

50% of user do google search.

- 2 Billion Q P Day
- $\frac{2000 \text{ million}}{60 \times 24} = \boxed{1.4 \text{ M QPM}}$

---

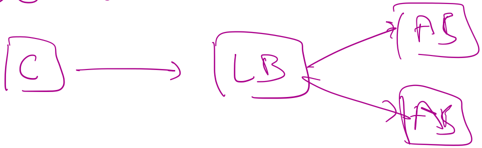
single LB can't hold so many connectn.

→ DNS hold mapping of all LB.

→ Google will have many data centers. (DC)

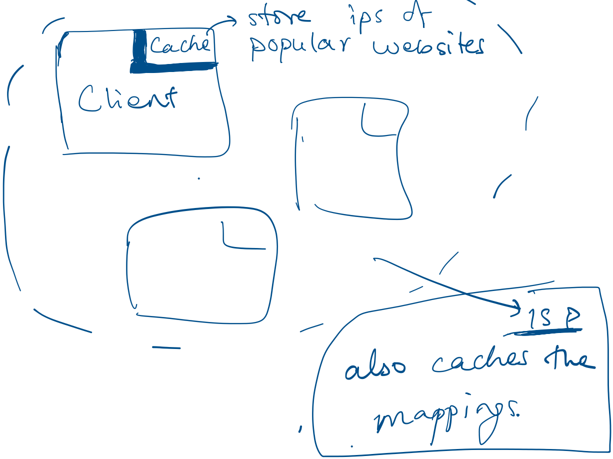
→ DC will have a common LB.

→



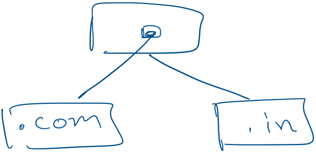
DNS in itself is distributed.
All will have same data.

DNS reshuffles the ips in the list and the returns,



store ips of popular websites

also caches the mappings.

- for cache miss it fetches data from DNS.
- finally reaches top level domain
  - . com
  - . org
  - . gov.