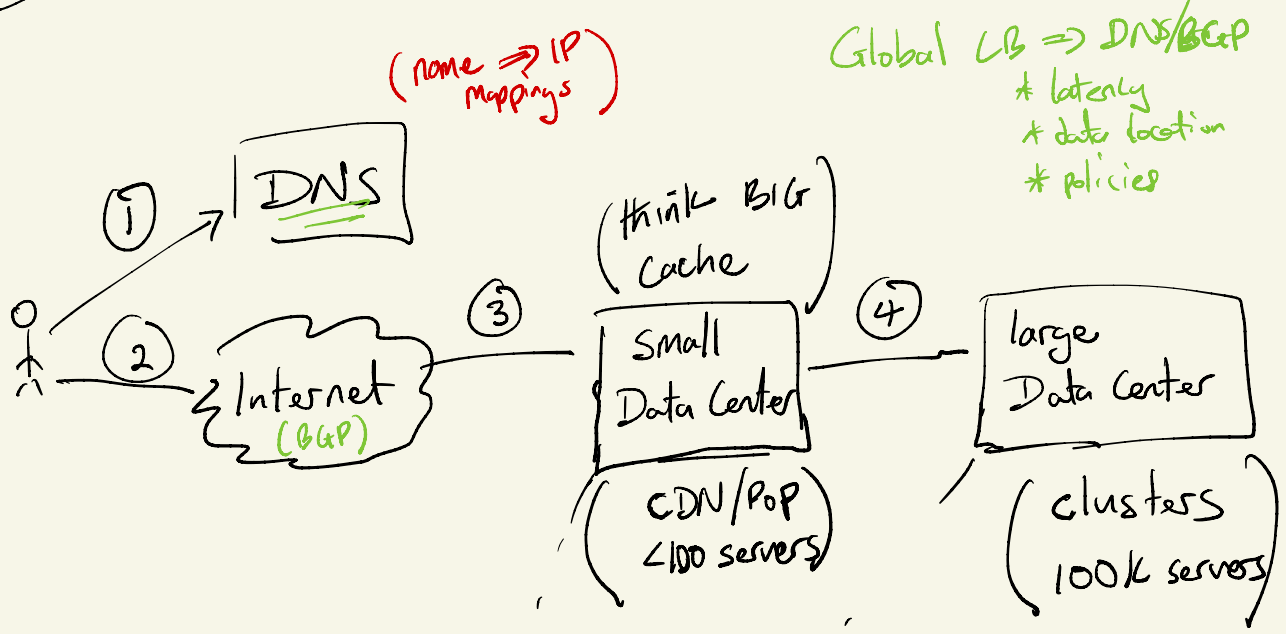
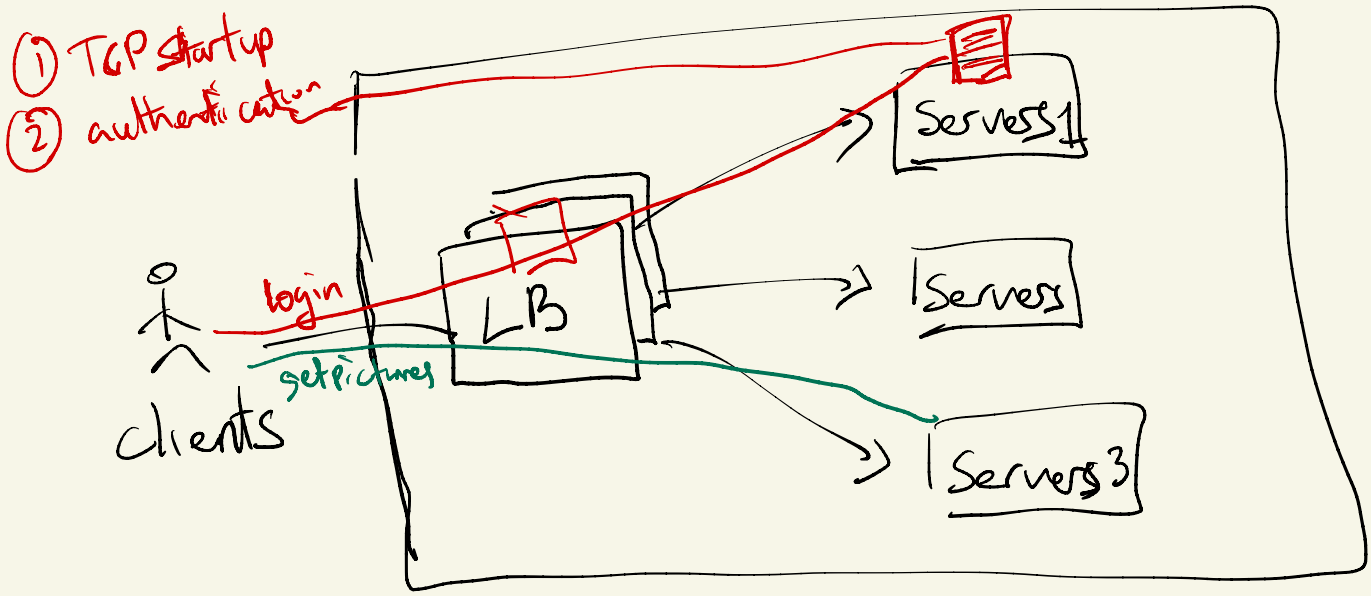
Today

1. Review of Day 6 (Please answer Top Hat Questions)
2. Modulo L/B
3. Consistent hashing

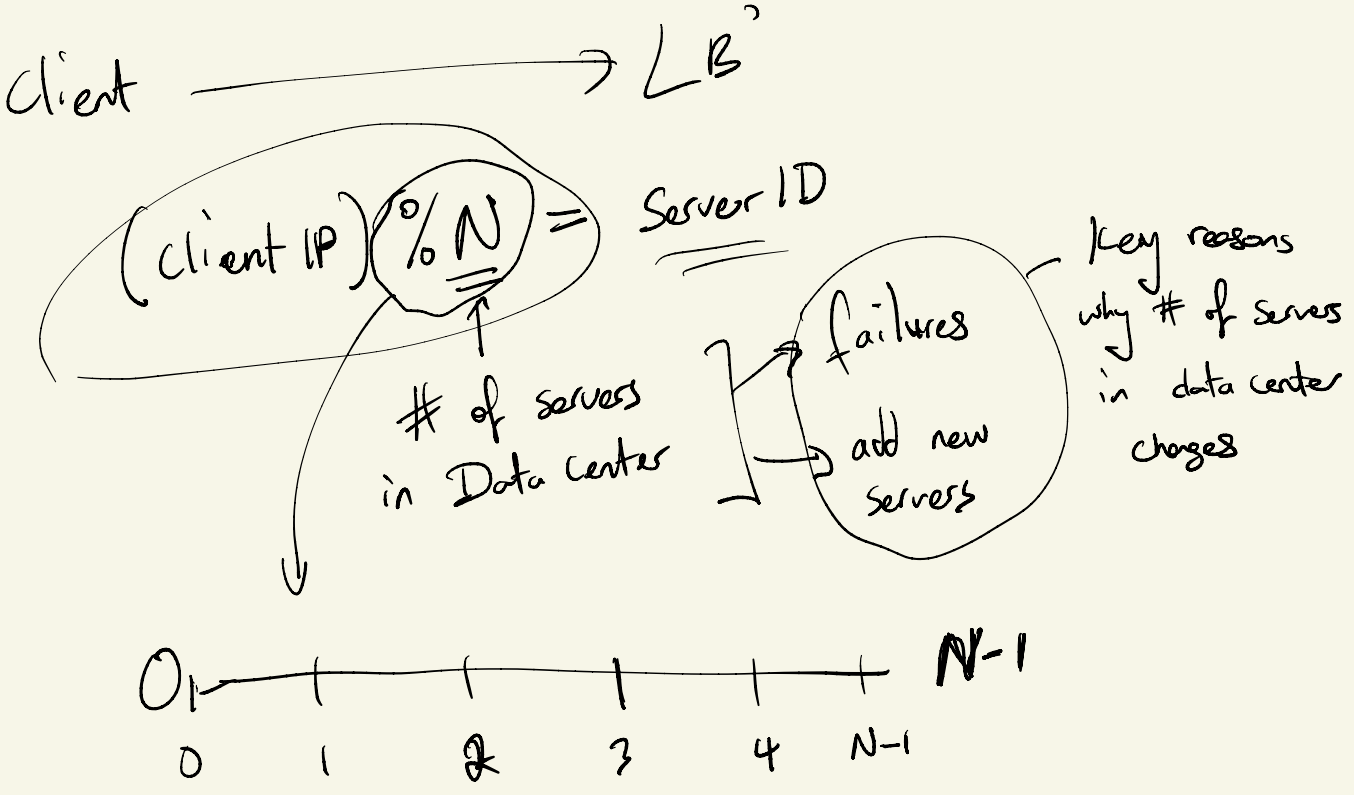


Today



LB techniques	eBay's Neutrino	Google's Maglev	FB's Katran	AWS ELB	GitHub LB	fastly
least connection	✓			✓		
Round robin	✓			✓		
Chash (consistent hashing)		✓	✓			
R Hash (rendezvous hashing)		✓	✓			
Modulo LB					✓	✓

Modulo LB

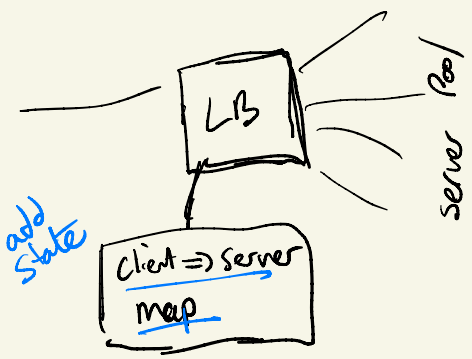


N: 6 ⇒ 7 $\left(\frac{28}{42}\right)$

How can you extend LB (Random / Modulo) to provide consistent mapping of client to servers?

1. LB should hold cache of client ⇒ server mapping
2. store the mapping in an external location (disk or S3)
3. Random w/ Cache

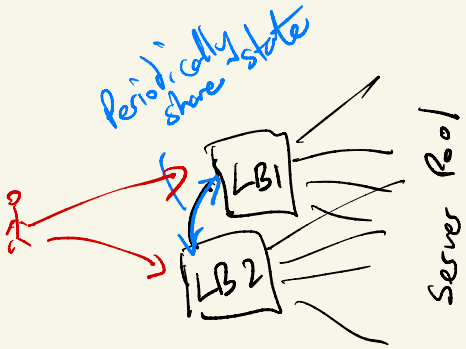
Extending RoundRobin



on LB failure ⇒ find way to replicate state

(external ⇒ S3 or persistent)

if many LB ⇒ find way to share state



(1) external K/V

(2) RPC calls to exchange state

Maintaining & sharing state is complex

introduces issues during LB failure/restart

hard to share state fast enough

How can you be stateless & provide consistent LB?

MD5 ⇒ 64 bits

Consistent Hashing

32 bits

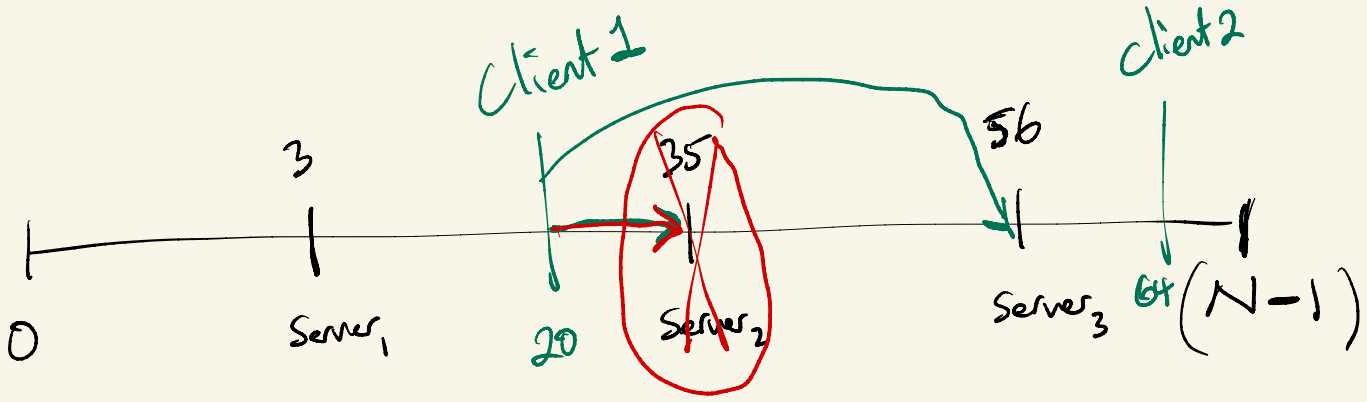
Keyspace

(1) Server ID = hash (IP address) % N

MD5 SHA1

128 / 256

(2) client ID = hash (client IP address) % N

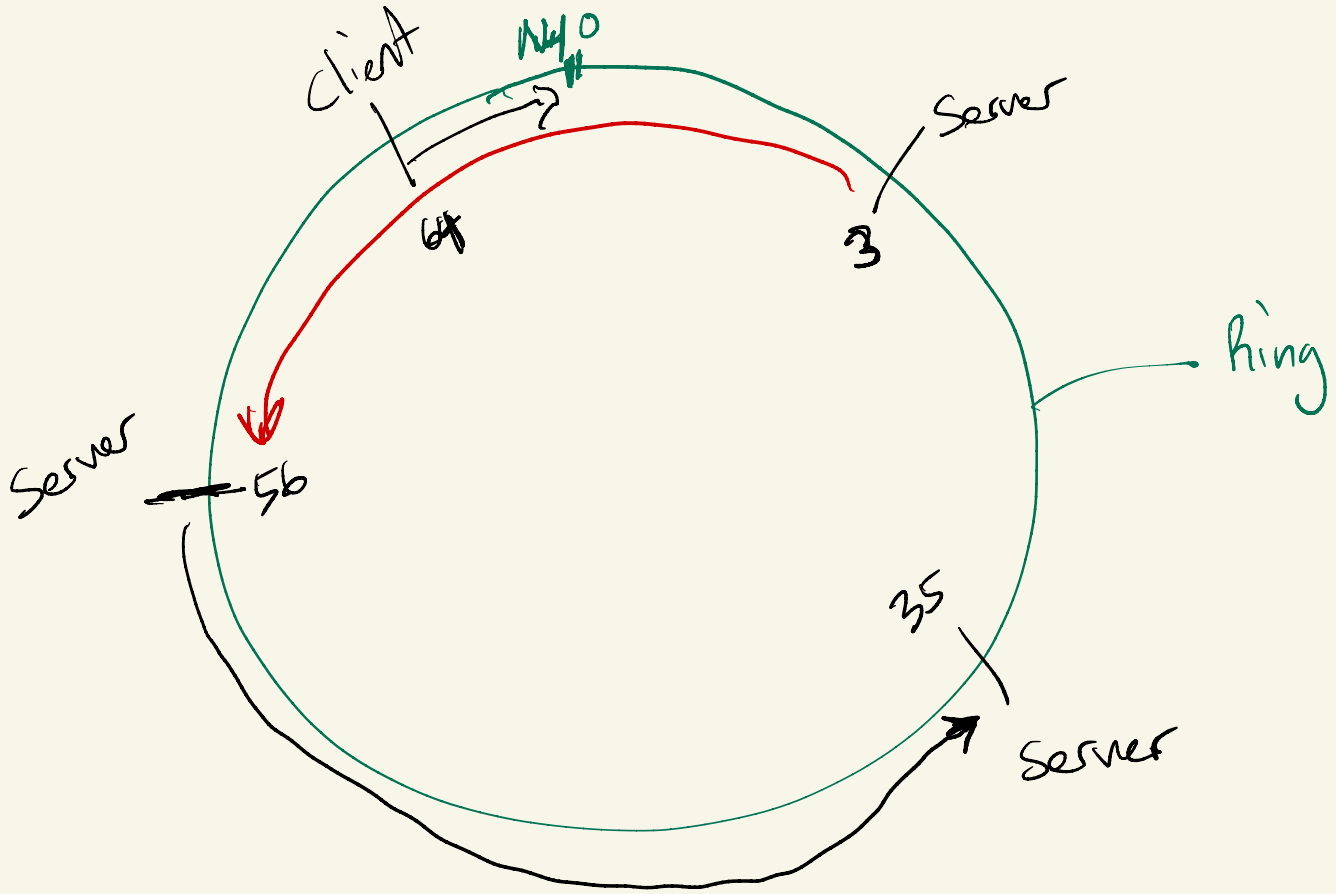


using the equations ⇒ every LB can calculate server ID without state

(1) How to deal w/ hash collisions?

(a) client ⇒ this okay: the clients are assigned to the same server

(b) server ⇒ change IP address of the server




```
serverlist = sort server ID
foreach server in serverlist {
    if (server > clientID) {
        myserver = server
        break
    }
}
```

find first server with ID greater than client

```
if (myserver == nil)
    myserver = serverlist[0]
```

if no server w/ ID greater than client then you are in wrap around edge case then pick first server (which turns out to be next server when in "ring")

0 — N-1

N

"key space" wrap around