HLD-101 (Consistent Hashing)

Monday, 3 January 2022 8:26 PM

Lets start at 9:06PM

---

del.icio.us
- → addBookmark(userId, url)
- → getAllBookmarks(userId)

1 laptop → Bookmark website

del.icio.us

Methods:-

① addBookmark(userId, url)

② getAllBookmarks(userId)

Hosting Website

del.icio.us
↳ domainName (unique)

Domain Registrar

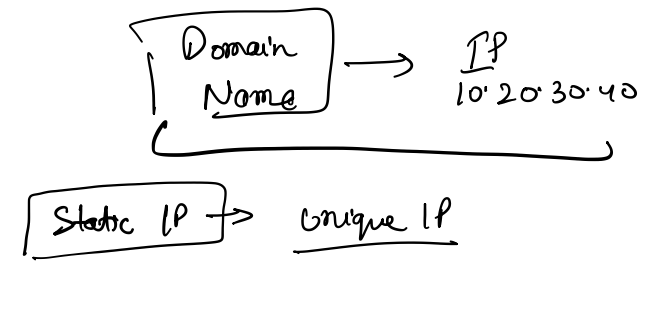
① goDaddy
② Name cheap ← del.icio.us
↓
Agents / Domain Registrar

ICANN

domain → owner	email / ph / ipaddr

→ domain

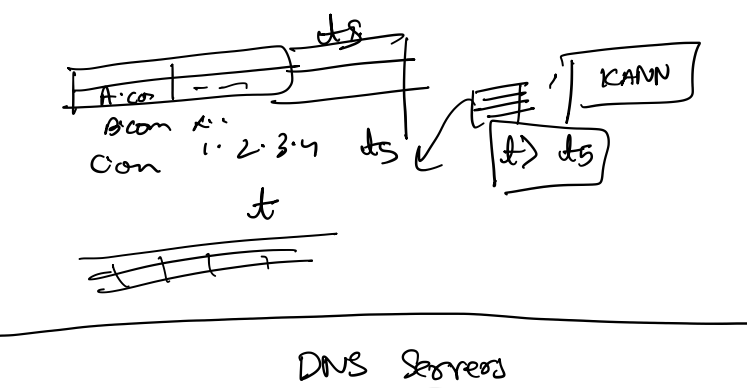
del.icio.us → 10.20.30.40



C
M1
C2
ICANN

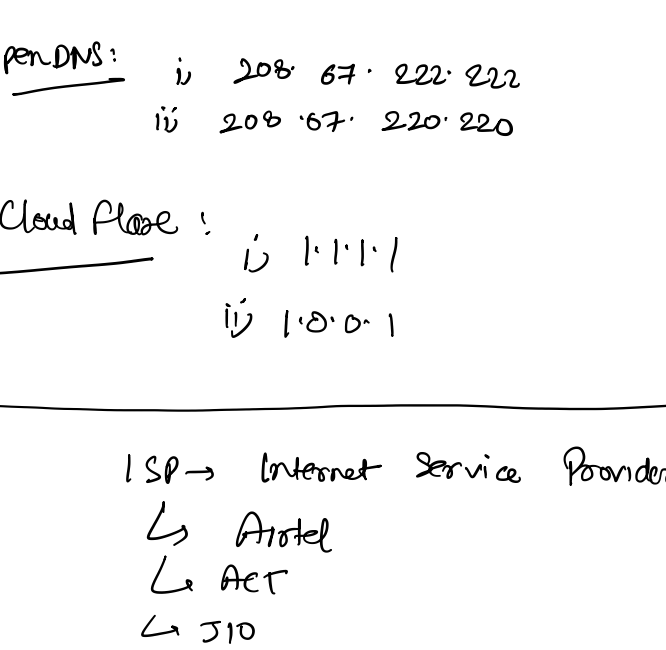
CACHING

DNS Servers (Domain Name System)



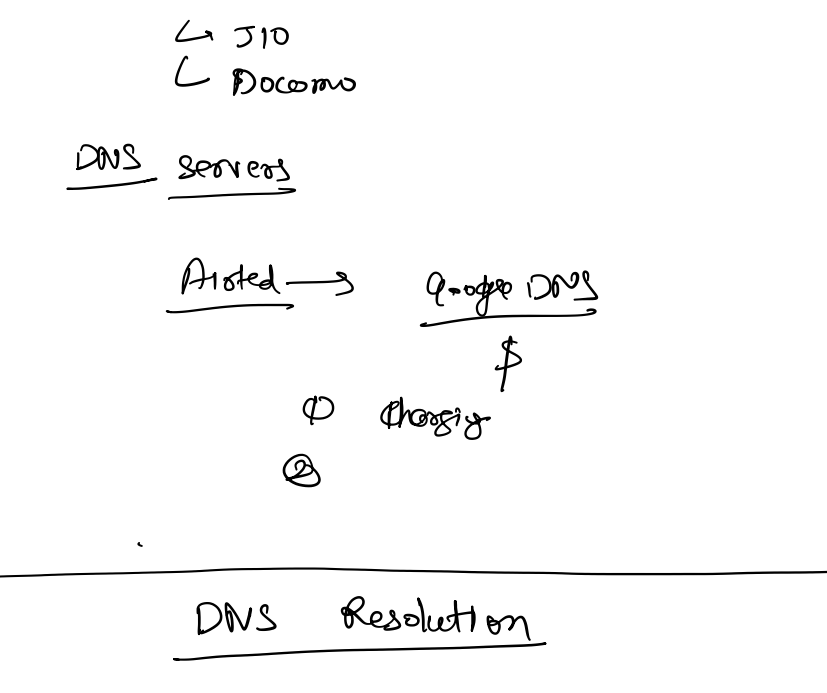
D1
D2
D3
D4
ICANN

ICANN → Directory of internet
↓ ↓ ↓
DNS1 DNS2 DNS3



t1
DNS
t1
ICANN
t>t1
DNS
Domain Name
IP 10.20.30.40

Static IP → unique IP



10.20.30.40
goDaddy
ICANN
del.icio.us
del.icio.us | 10.20.30.40
ts
A.co
Bcom
Com
1.2.3.4
ts
ICANN
t > ts
t

DNS Servers

① google:- i) 8.8.8.8
ii) 8.8.4.4

② openDNS:- i) 208.67.222.222
ii) 208.67.220.220

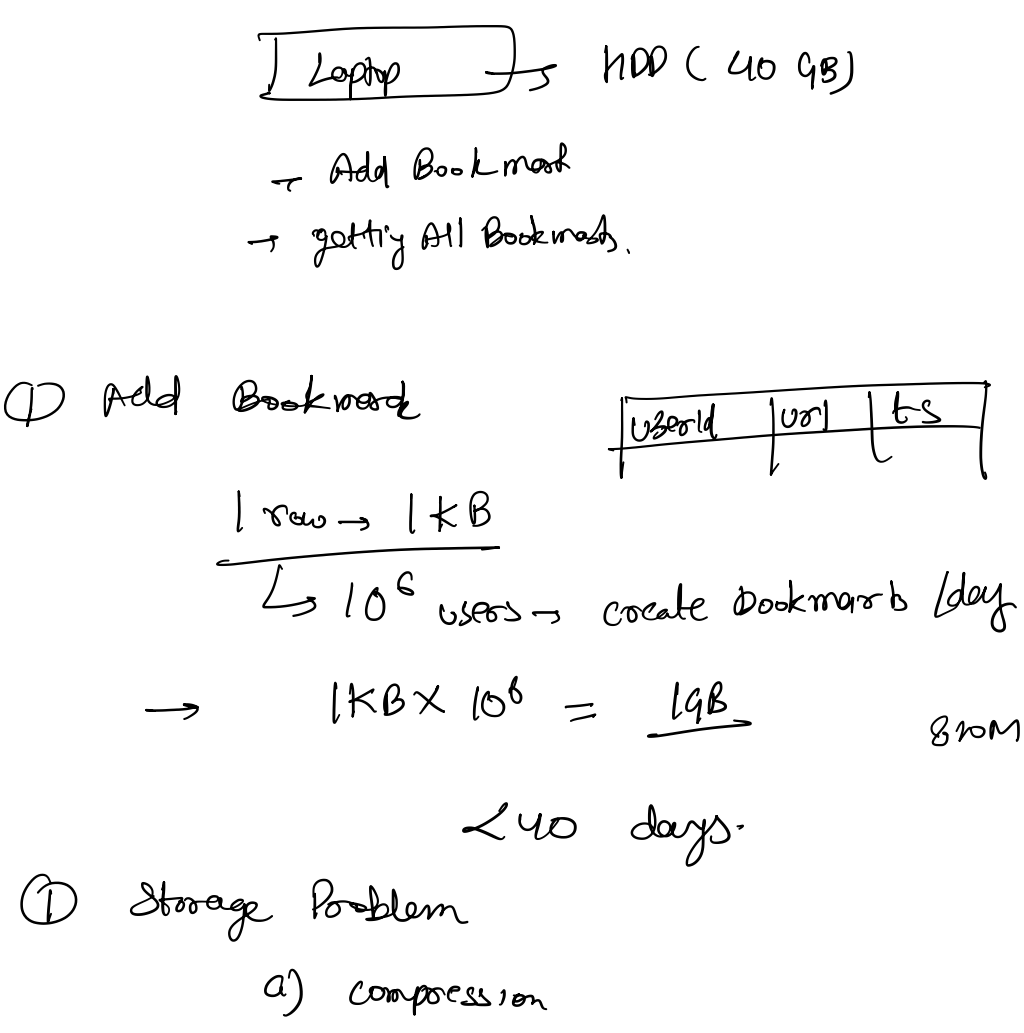
③ Cloud Flare:- i) 1.1.1.1
ii) 1.0.0.1

ISP → Internet Service Providers
↳ Airtel
↳ ACT
↳ JIO
↳ Docomo

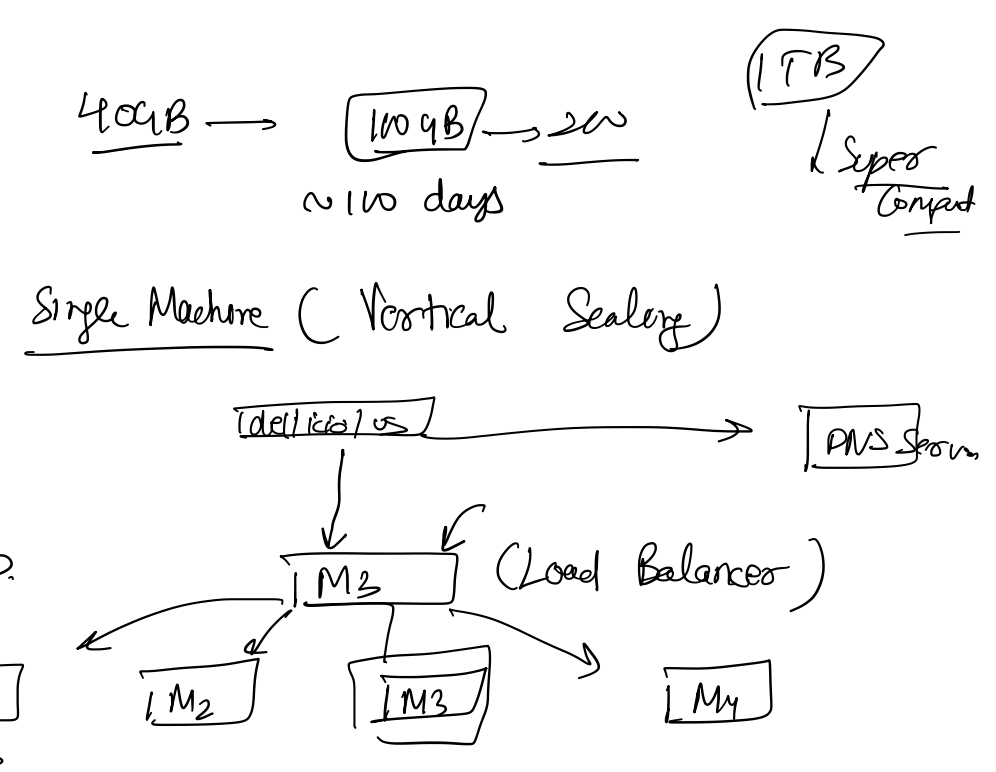
DNS servers

Airtel → Google DNS
$
① charging
②

DNS Resolution



scaler.com → 40
Brow Cache (Local DNS server)
OS Cache (Local DNS Server)
Resolver Cache (ISP level / open DNS)
Root Server
Top level Domain Server
Root
.com
.co
.in
.edu
.com TLD (1985)
a b c ... S
Scaler AS (Authoritative Servers)
chat.scaler
.co



del.icio.us
40
10.20.30.40
DNS Server
Laptop
HDD (40 GB)

→ Add Bookmark
→ getting All Bookmarks.

① Add Bookmark

userId	url	ts

1 row → 1KB
↳ 10^6 users → create Bookmark / day

→ 1KB × 10^6 = 1GB 800MB

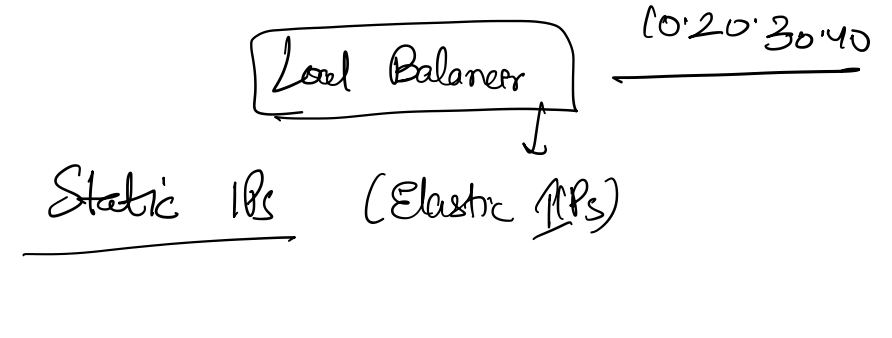
< 40 days.

① Storage Problem
a) compression
b) Increase capacity →
c) More machines

40GB → 100GB → 2w
~100 days

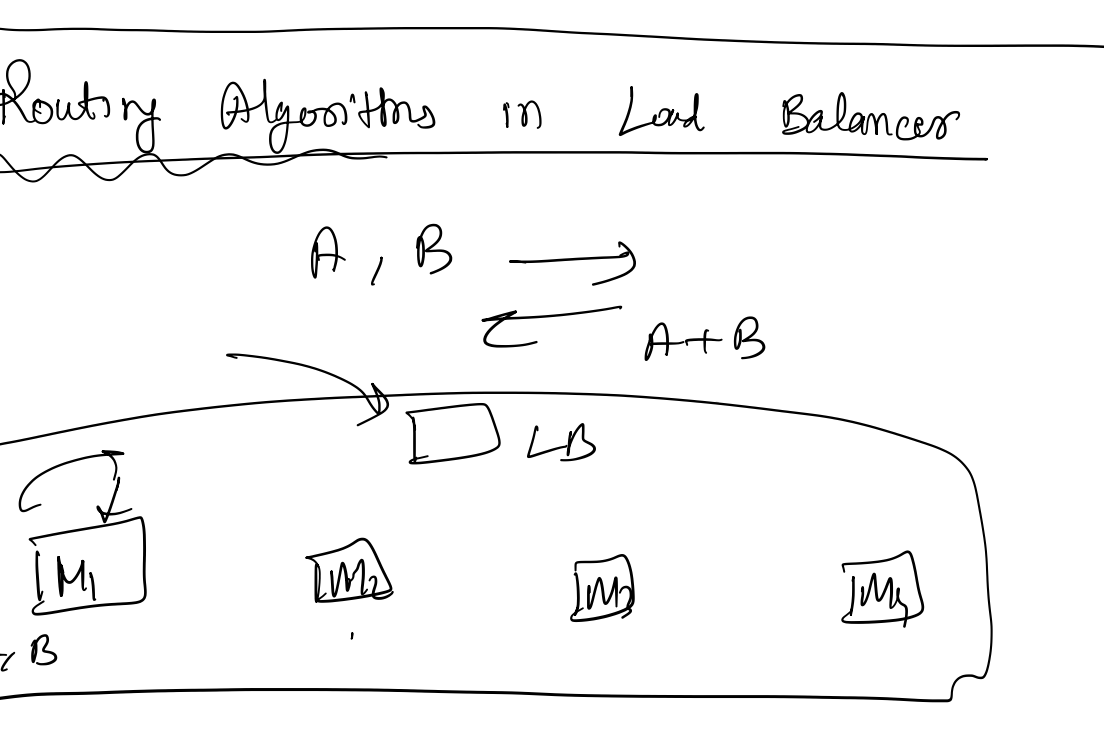
1TB
↳ Super Computer

→ Single Machine (Vertical Scaling)



del.icio.us
DNS Server
IP
M3
(Load Balancer)
M1
40GB
M2
M3
M4
~160GB

→ Horizontal Scaling



Bro
LB
α
S1
S2
S3
S4
S2

Requirements

→ Hot copy LB
→ Primary LB has gone down (Zookeeper)
→ Static IP

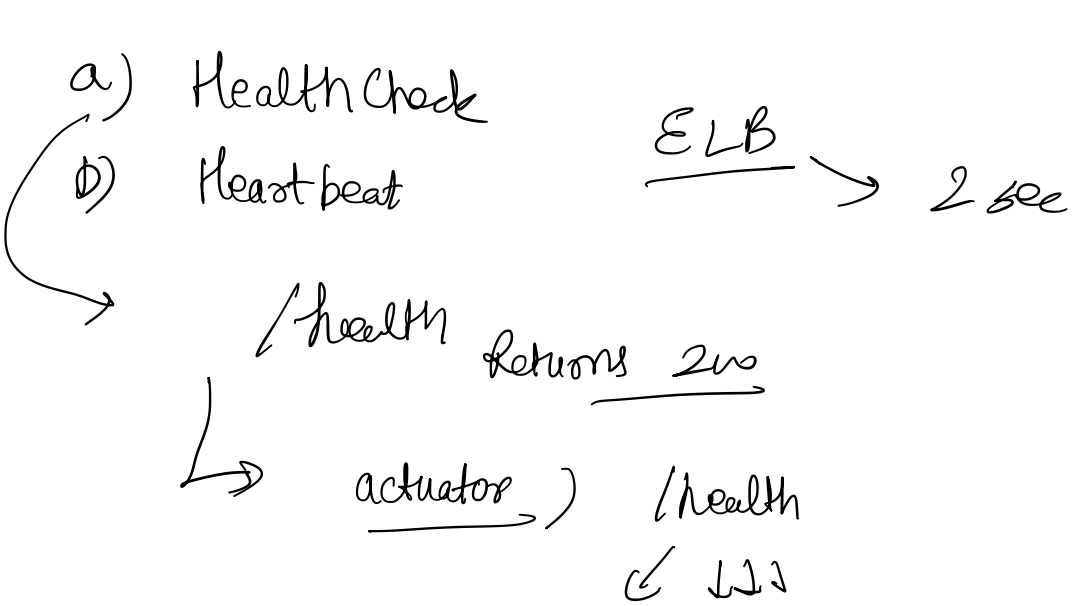
10.20.30.40

LB1 → LB2

Static IP
IP Address ← Public IP
Load Balancer ← 10.20.30.40

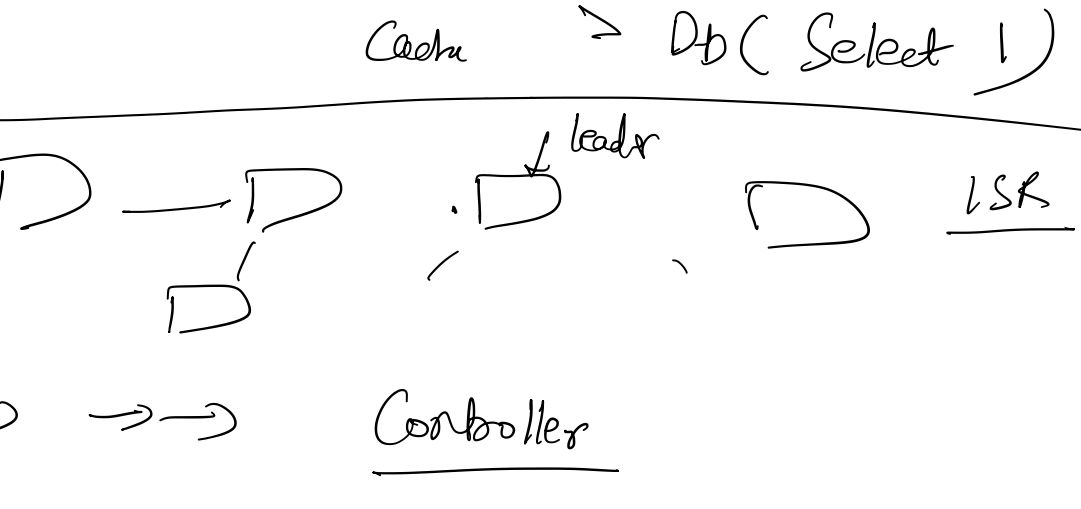
Static IPs (Elastic IPs)

Routing Algorithms in Load Balancer



A, B
A+B
LB
M1
M2
M3
M4
A, B

① Stateless System



LB
AS1
AS2
AS3
AS4

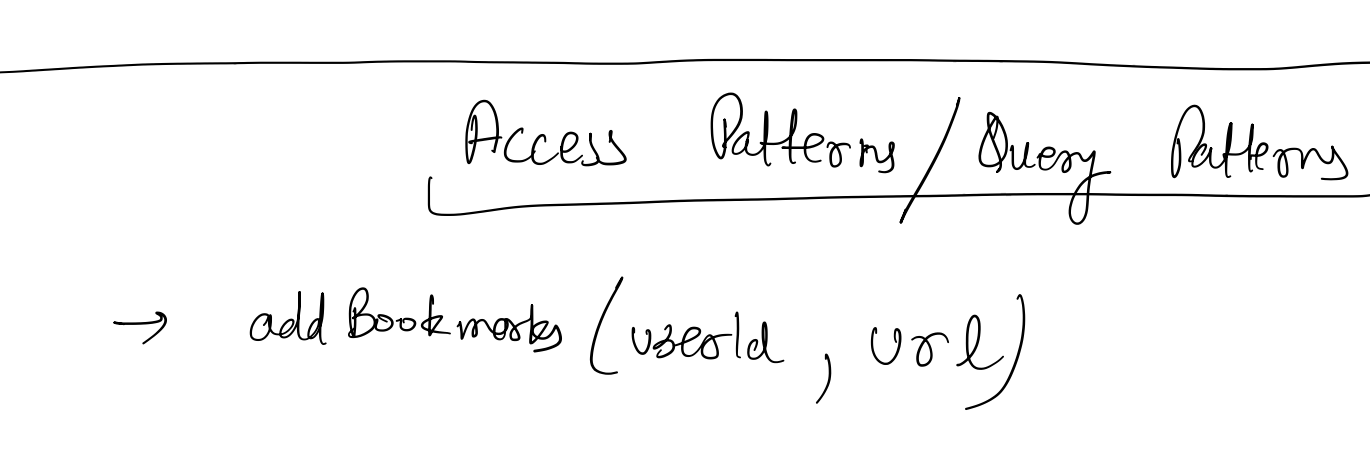
1) Round Robin

2) Weighted Round Robin (RT of servers
Load of servers
top)

a) Health Check
b) Heart beat

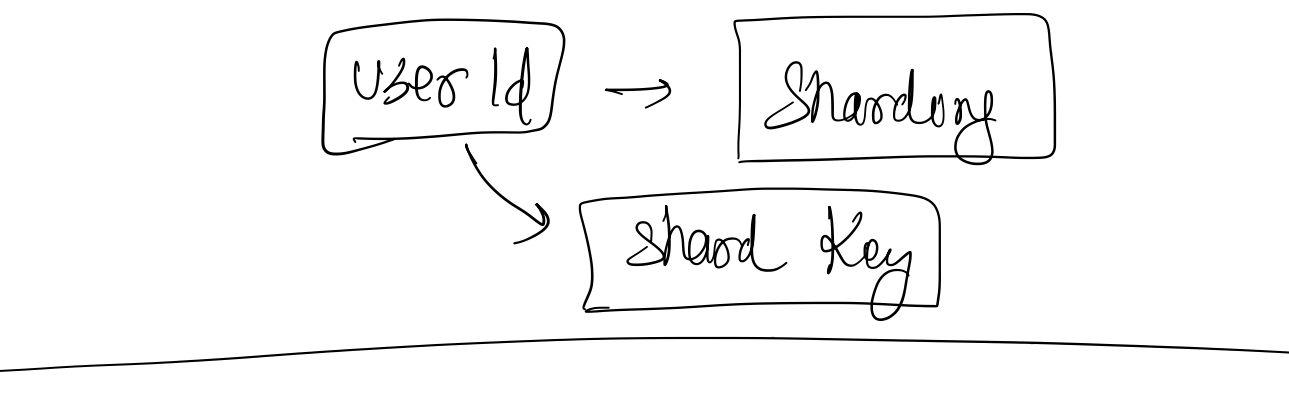
ELB → 2 sec

/health Returns 200
↳ actuator) /health
Cache → DB (Select 1)



leader
ISR
HB
Controller

Stateful System



LB
Not work
add
Cache
Cache

Requirements

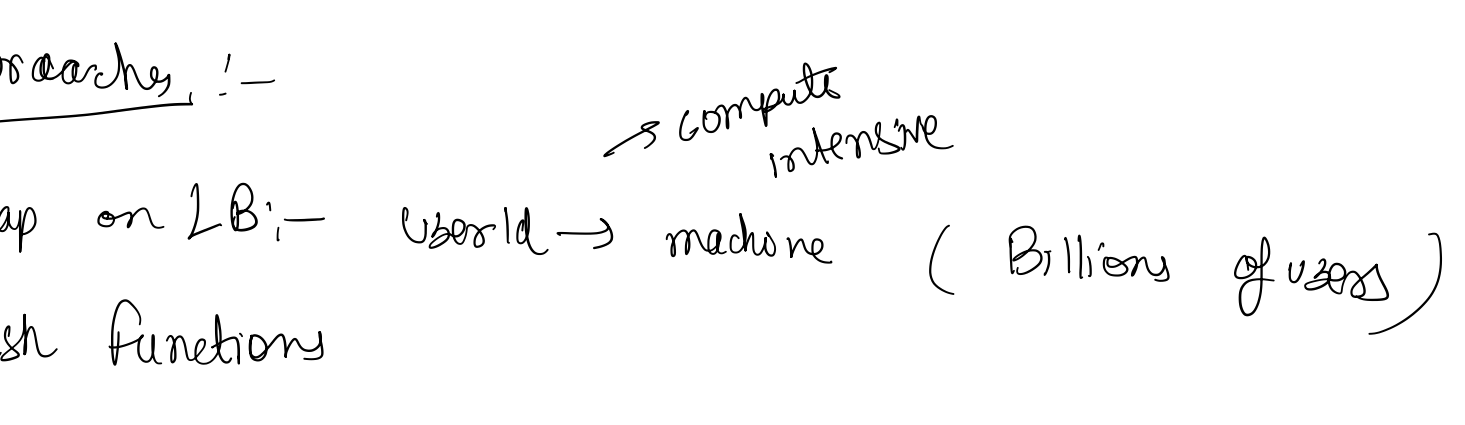
→ Data is split across servers
→ Load Balancer needs a mapping user → machine
→ LB → where to send request
→ it should send request to single server

Access Pattern / Query Patterns

→ addBookmarks(userId, url)

→ getBookmark(userId)

UserId → Sharding
↳ Shard Key



5
LB
(Hash Function)
AS1
AS2
AS3
AS4
A5
0
1
2
3

H(UserId) → Server

UserId % 4 [0, 1, 2, 3] 5

%4
%3
%5

✓ → Equal load on machine ✓

✓ → LB knows the split (userId → machine)

Problem → ① Machine goes down?
② Add more machine?

Approaches:-

① Map on LB:- userId → machine (Billions of users)
→ compute intensive

② Hash Functions

4 servers → 5 servers

H(userId % 4)

uId	machine Id
1 →	1
6 →	2
8 →	0
3 →	3

userId % 5

1 →	1
6 →	1
8 →	3
3 →	3

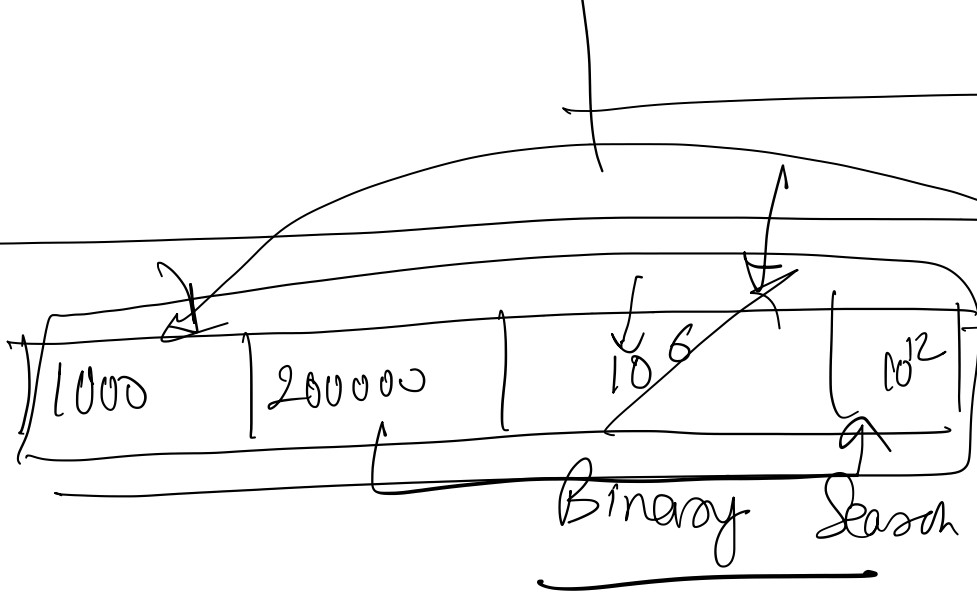
→ Adding a single machine leads to high entropy

→ Can lead to downtime

Consistent Hashing

Hash (userId)

UH(userId) → position in the array

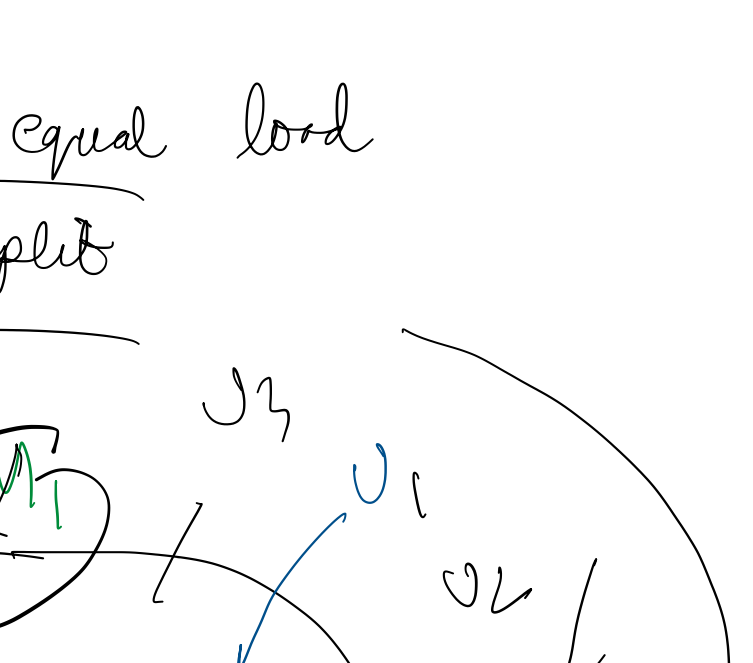


10^18
M1
1
2
U1
M2
U2
M3
U3
M4
U4

Hash Function for Machine

H'(IP) → [0 -- 10^18]

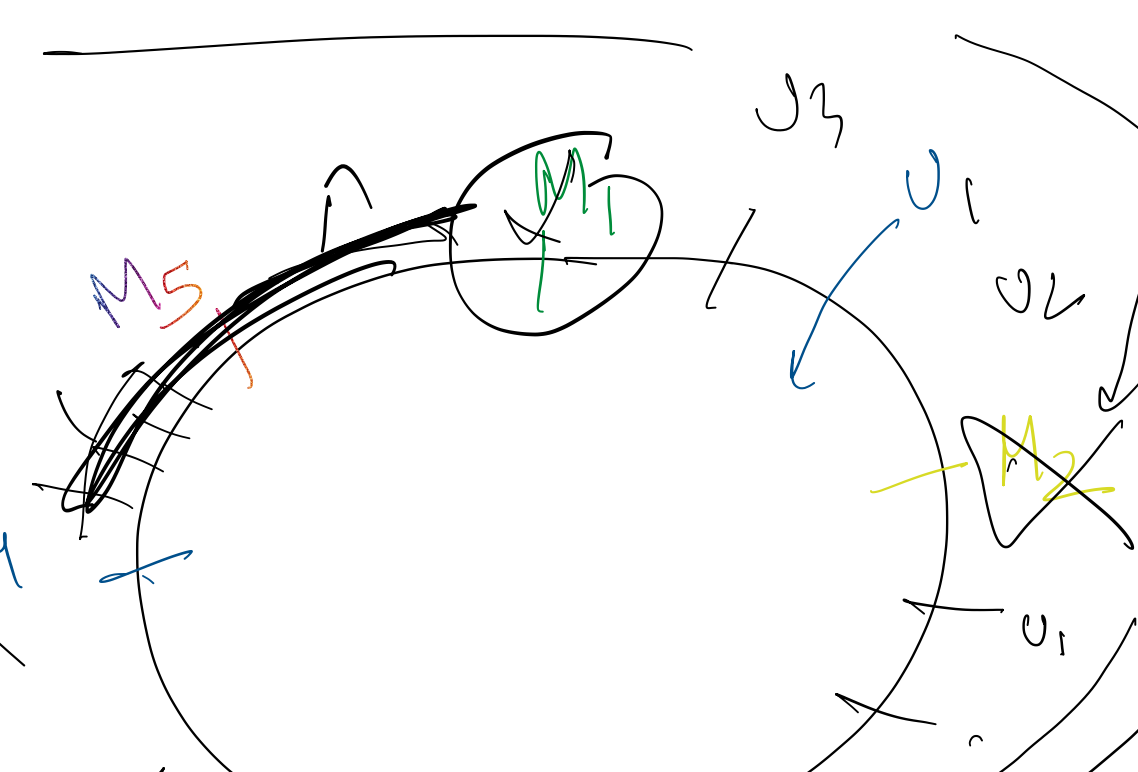
userId()
UH(id) →



AWS
VPC
Public
LB
Private
private IP
Server

---

4 machines



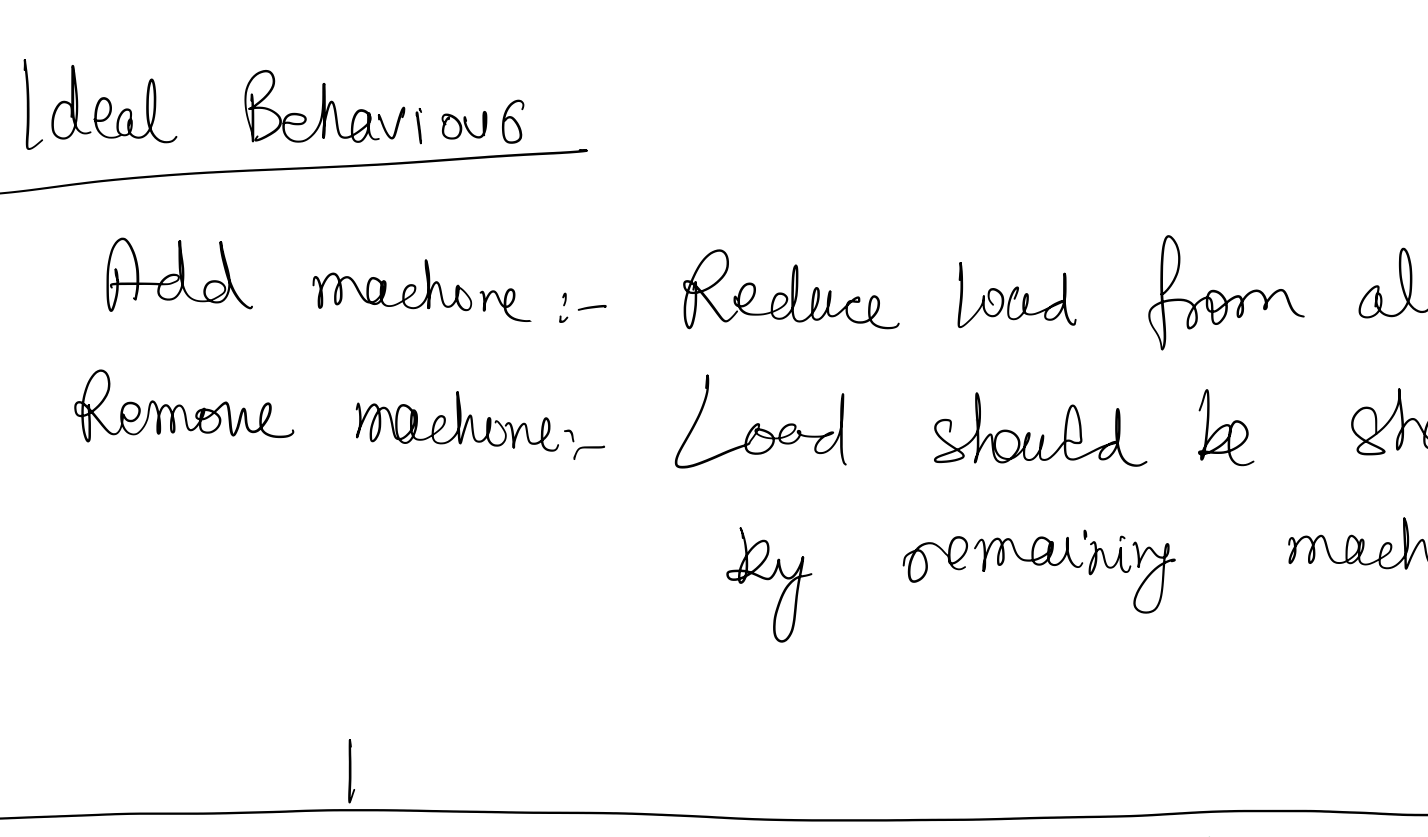
1000
200000
10^6
10^12

Binary Search O(logn)

≡ # machines

→ userId (UH) →

→ All machines get equal load
→ LB knows the split



M1
M5
M4
U3
U1
U2
M2
U1
M3

→ Cascading failures

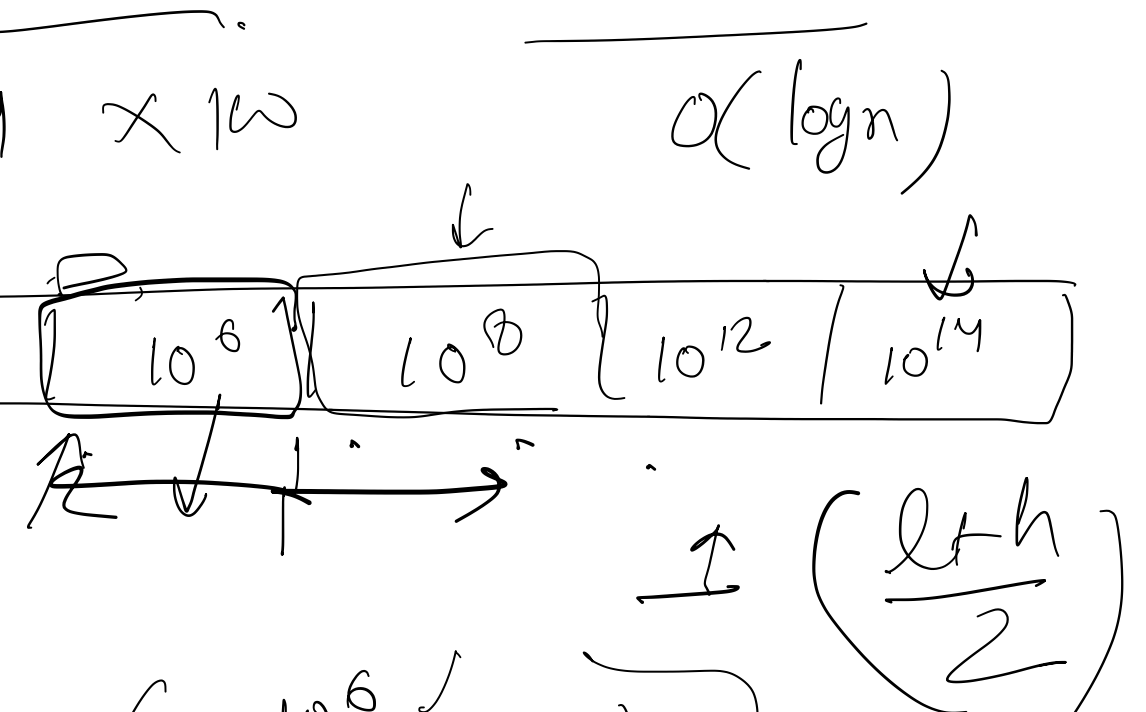
Ideal Behaviour

① Add machine:- Reduce load from all machine

② Remove machine:- Load should be shared by remaining machines

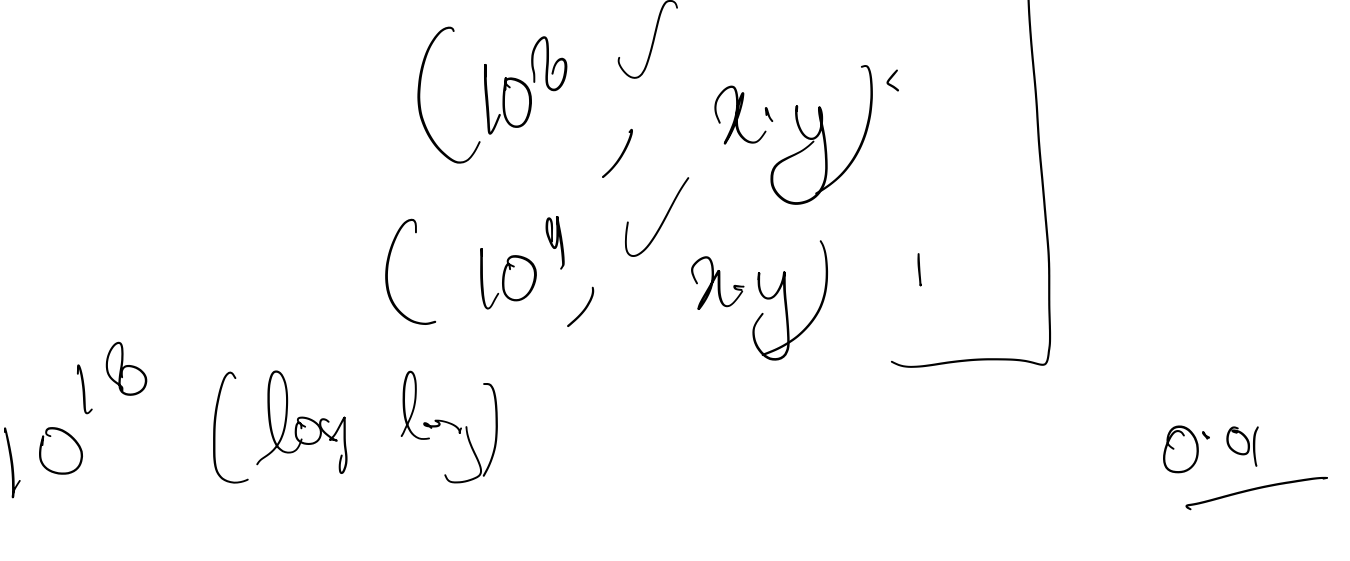
---

UH(userId) : [0 -- 10^18]
MH1 (machine Id)
MH2 (Mid)
MH3 (Mi)
MH4 (Mid)



M1
M2
M3
M5
M1
M2
M3
M5
M4
M1
M5
M4
M3
M1

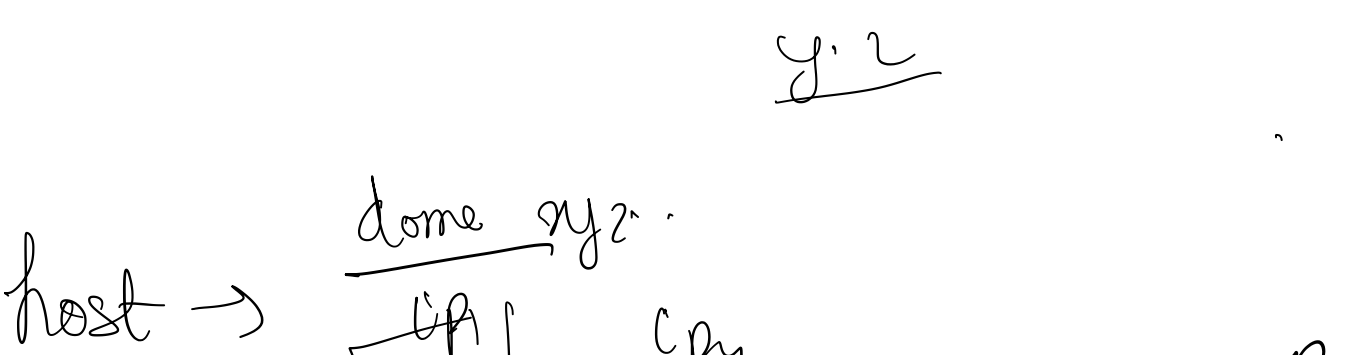
→ 10 Hash function



ob. locations = # machines × # hash functions
= 4 × 10

10 machines <

O(logn)



2
10^4
10^6
10^8
10^12
10^14

→ UH(userId)

$\left(\frac{l+h}{2}\right)$

(10^6 ✓, x.y)

(10^6 ✓, x.y)
(10^4 ✓, x.y)

4 :- 10^18 (log log)

0.01

10^4 : x.y
y.z

host → dome xyz.
cp1
cp2
cp3
cp4

HTTP : 80

telnet xyz.com 80

xyz.com : [1.2.3.4
4.5.6.7
7.8.9.10]

telnet xyz.com 80

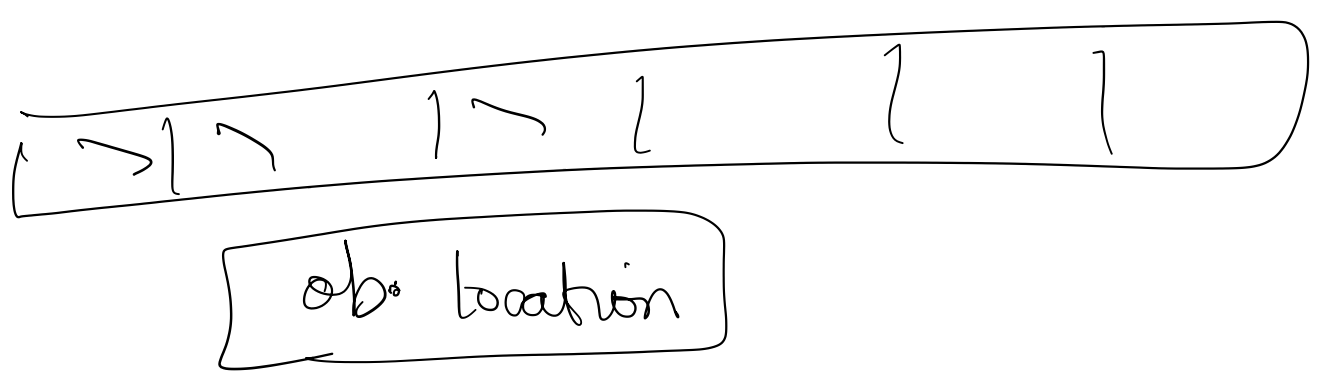
traceroute

whois scaler.com

traceroute scaler.com

→ Airtel DNS. . . .

Object
log location
storing ideas.



ob. location