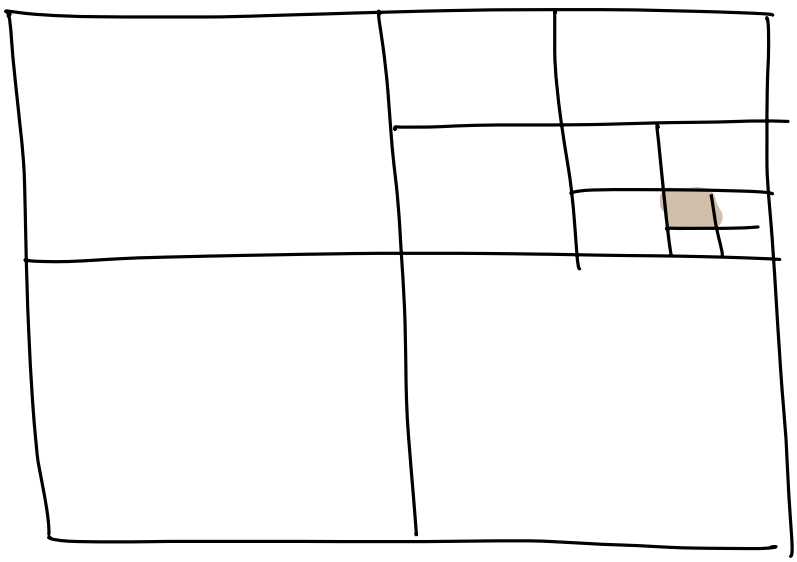
Agenda.

→ Quad Tree recap.

→ Uber.

Size of the Quad Tree.



Size of the Quad Tree = Quad tree cells + Places of interest.

100 M

Place: id | name | type | lat | long

8B 50B 4B 8B 8B

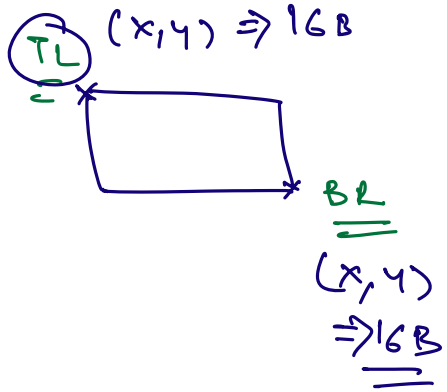
100 Bytes.

100 M × 100 Bytes.

10000 MB ⇒ 10 GB.

Quad Tree Cells

↳ # of cells * 32 B.



1 place of Interest per cell ⇓ 100M cells.

100 places of Interest per cell ⇓ $\frac{100M}{M}$ ⇒ 1M cells.

50

On an avg, every cell ≈ 20 places of interest.

$$\# \text{ of cells} = \frac{100M}{20} = 5M$$

5M x 32 Bytes.

160 MB.

Total size of the Quad tree = 10 GB + 160MB

= 11 GB.

Design Uber.

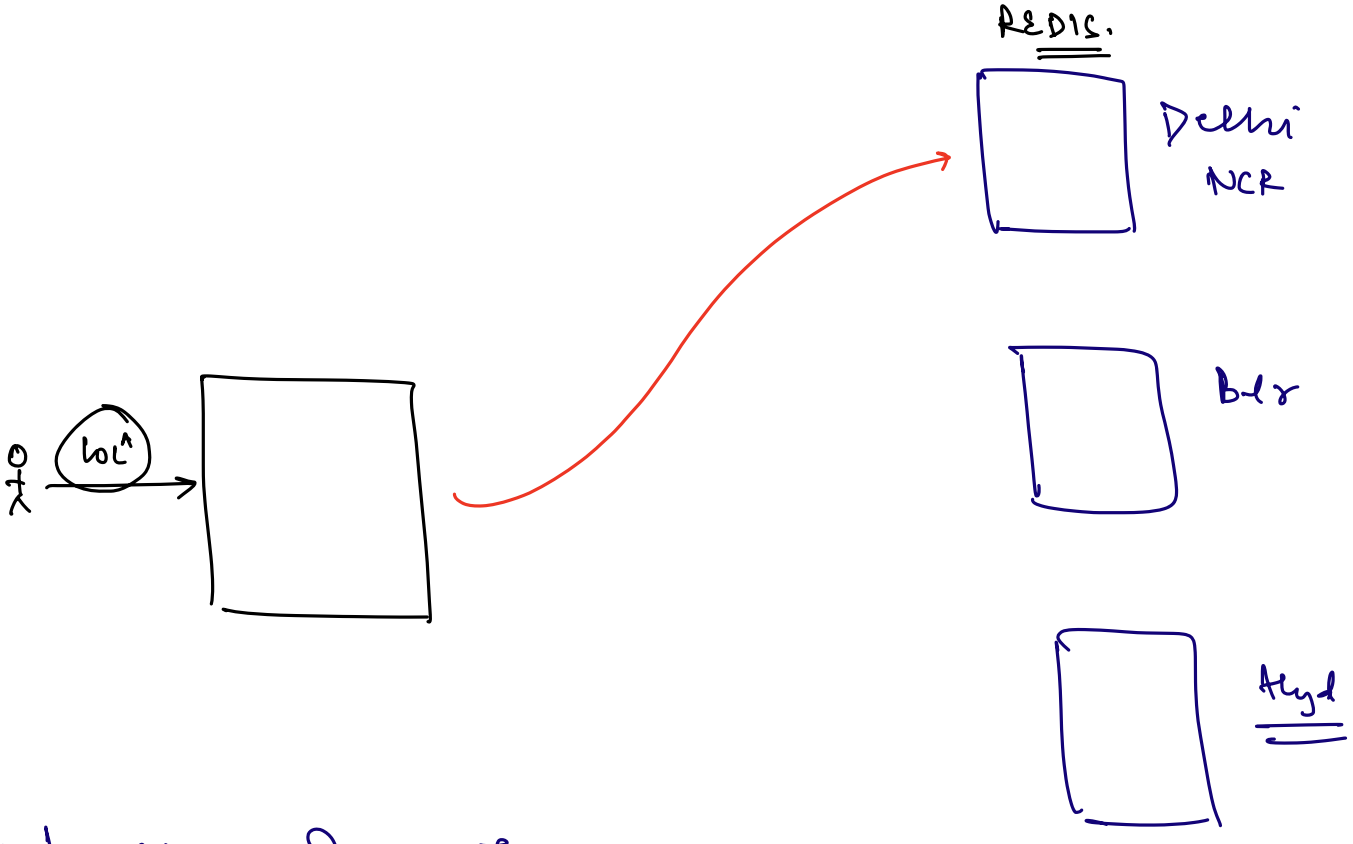
⇒ Driver locations.

⇒ Multiple cities.

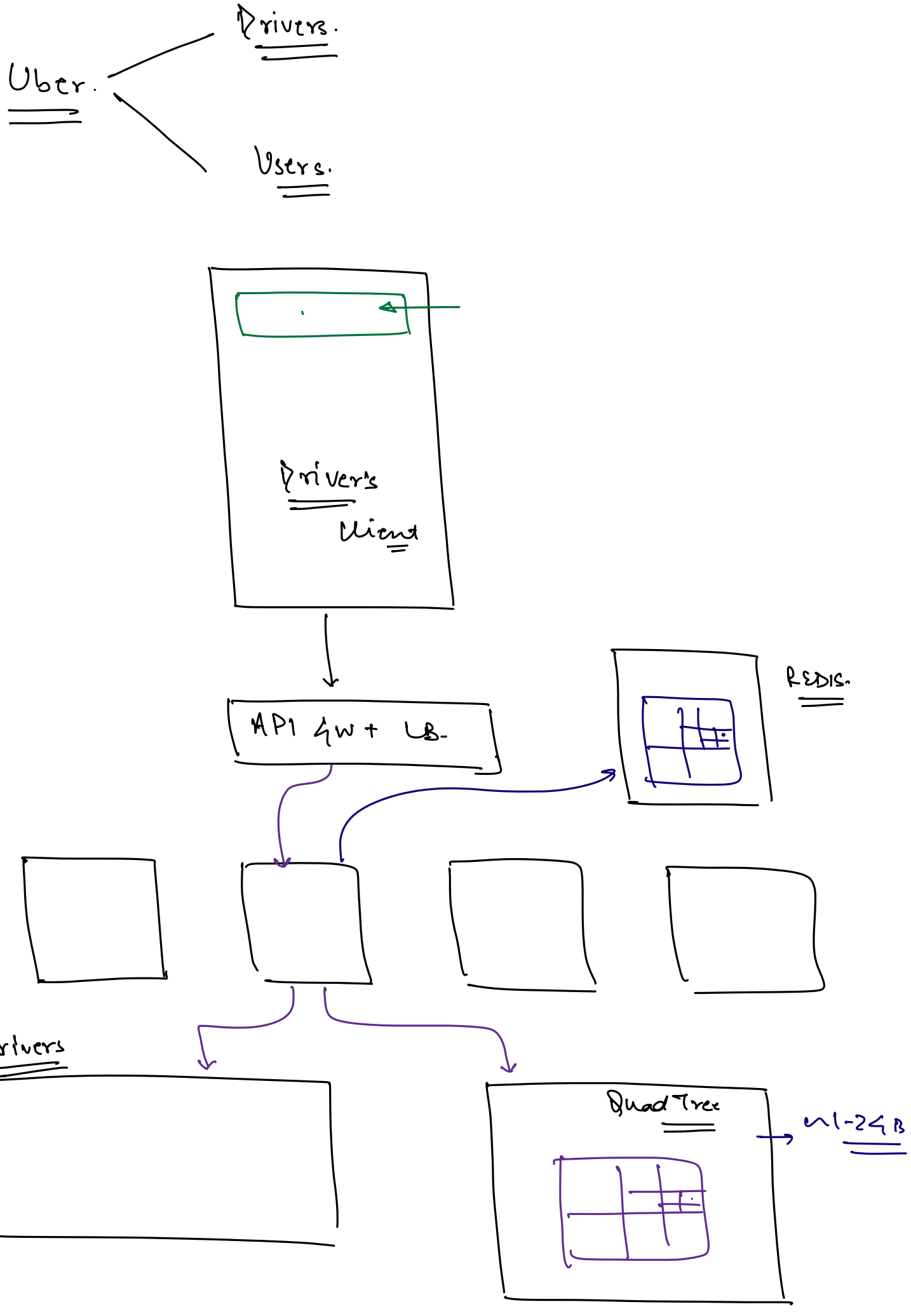
⇒ Single Quad Tree for the entire world.

VS

⇒ Quad tree for every city.



⇒ Localized Quad Trees.


Uber.
Drivers.
Users.
Drivers
Client
API GW + LB.
REDIS.
drivers
QuadTree
in 1-2 GB

⇒ Uber will first precompute the Quad Tree for every region.

⇒ Uber will have different databases for storing different types of data like users data, drivers data, Quad tree etc.

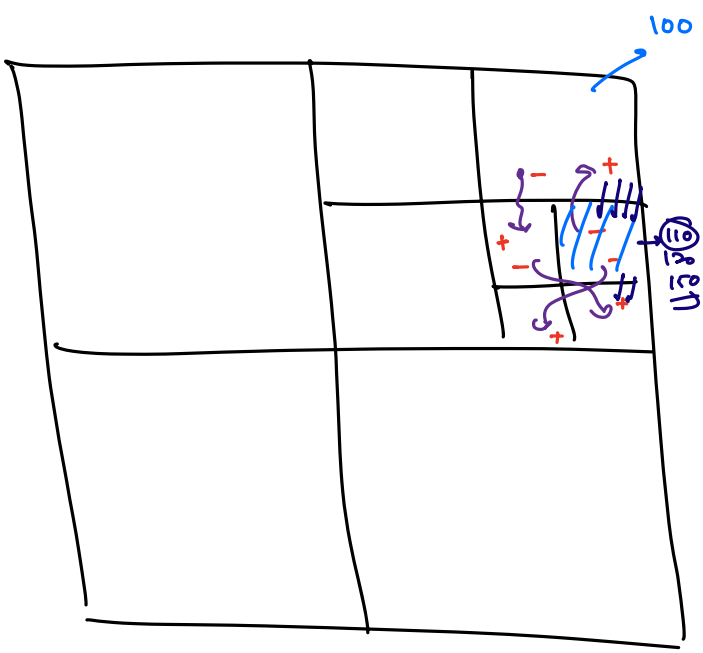
→ Quad Tree : Current loc^ of the Driver.

→ loc^ history.

↳ NoSQL. (TimeSeries DB)

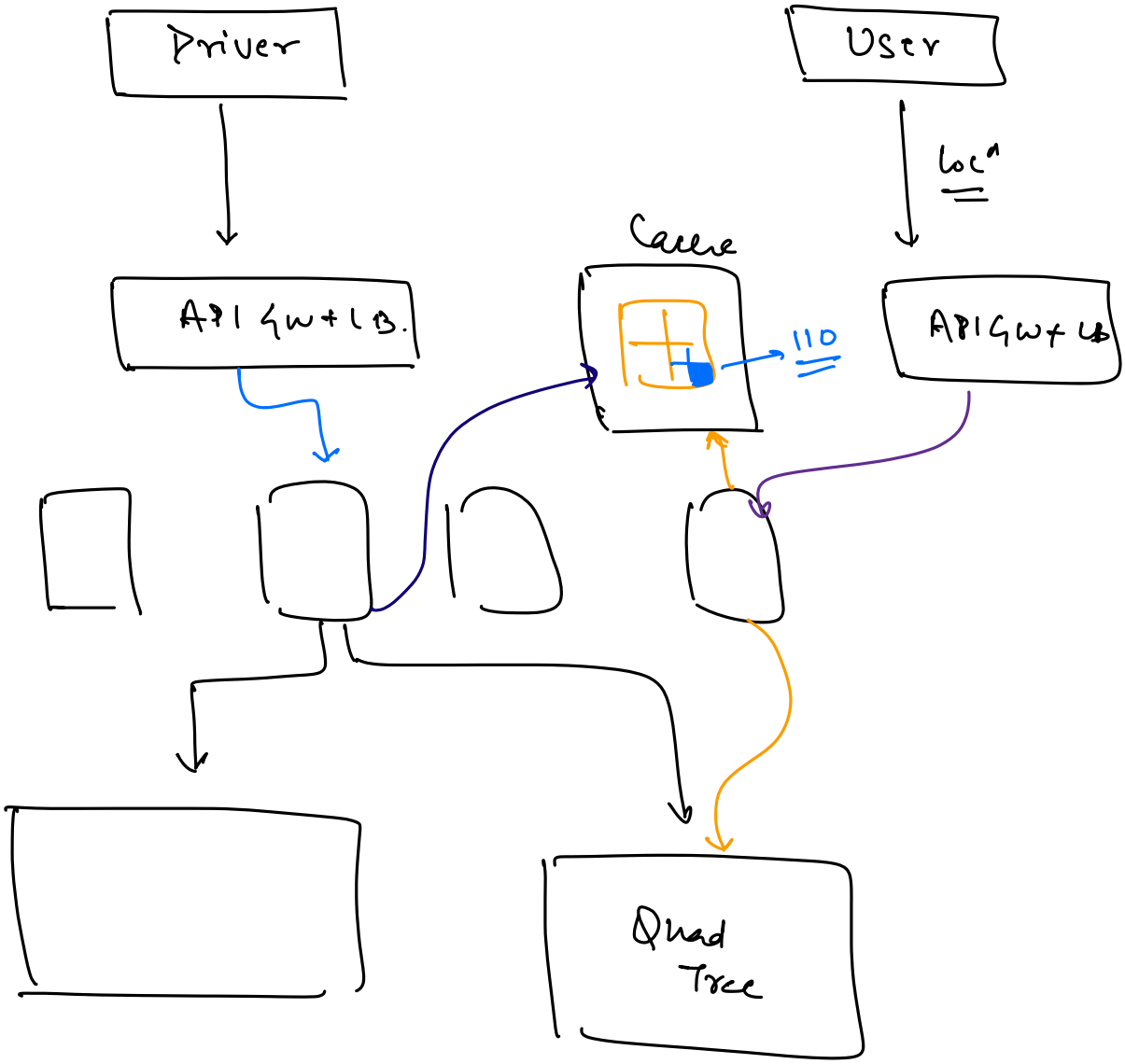
→ User's DB.

⇒ Drivers will keep on sending their live location to the Uber Backend after every 1 min, then Uber updates the location of the Driver in loc^ DB and Quad tree as well.



Lot of Updates in Quad Tree.

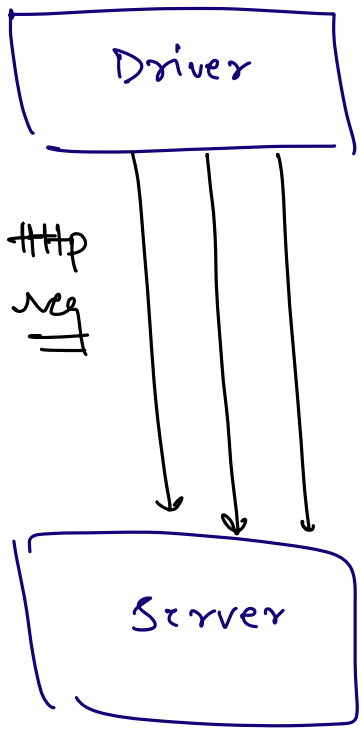
⇒ Whenever driver changes the location, Uber needs to update the location in the Quad Tree but division | merging of cells can happen in batches.



⇒ Uber will notify the first 10 nearest drivers to the user.

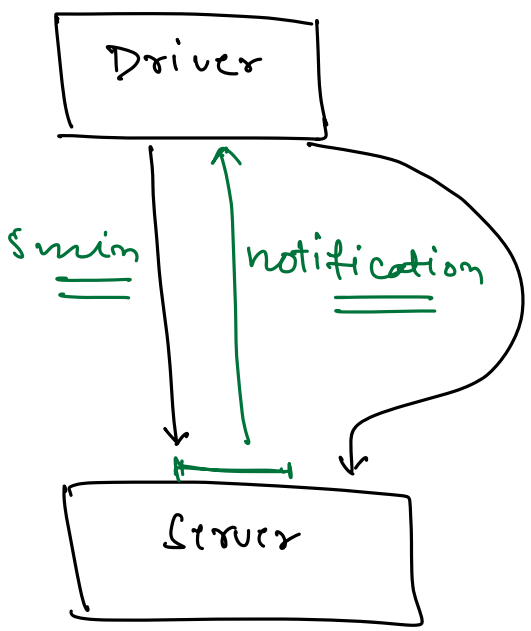
Timed Notification ≤ 30sec

Polling: Driver keeps on making constant request to the Backend to check if there's a ride available for them.

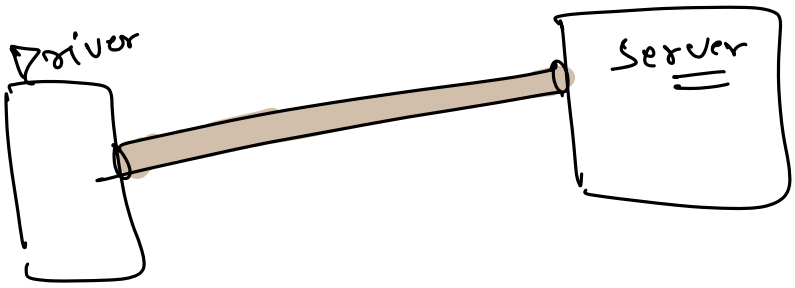


Long Polling.

↳ Http Req Response with timeout.



Web Socket.



Amount of resources for long polling will be less than Web Socket connection.

H/W.

⇒ Geo Hash

Popular Interview Ques.

Rate Limiter

Unique ID.

Thursday.

Friday