Key Take aways

- Challenges involved
- Use cases & functional requirements
- Find cabs near me
- Quad Trees
- Optimizing updates

Mastodon Constraints

Challenges involved in Uber → ride booking / sharing Service

- 10 million drivers
- 200 million MAU
- 50 million trips per day
- across the world

Minimal Use cases & functional requirements

- intra-city cabs
  - Out station rides
  - schedule rides for later
- Users & (cabs/drivers)
  - identify (user-id) ⇝ login/auth
  - location of user
  - identity (driver-id)
  - location of driver
  - is driver available
- User can book a cab
  - source, destination → input
  - current location
  - nearby cabs
- driver can accept/reject

Use-case

- I'm a user at location X (latitude, longitude)
- match me with the nearest cabs
- which are available

- send notification to all nearby cabs
  - Accept
  - Reject

Round Robin

10 million cabs worldwide

Store data about these cabs

location

available

XL / prime / - - - ⇒ meta data about cab

→ does all this data easily fit on a single machine?

→ if all this data is on a single machine, will the machine be able handle load of 50 M rides / day.

No!!

→ Shard the data

Sharding / Partitioning

↓ split data

↓ special case of partitioning

Split data across multiple machines.

Suppose I'm in Mumbai → book a cab

↳ I only care about drivers in Mumbai

sharding key → location

city-id

Intra-city } diff

Inter-city } product

↓ diff architecture.

within the city

Certain cities are more active than others — Ghaziabad

Bangalore

Mumbai

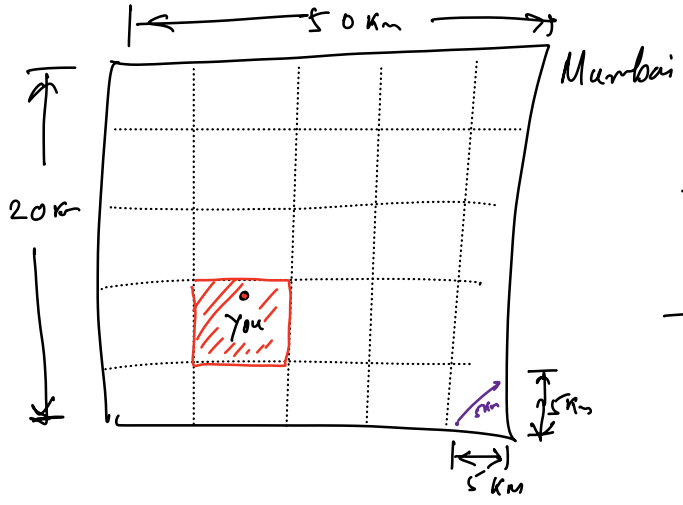
{

↓

makes the problem a little easier

10 M drivers → 50,000 drivers in a city

Within city → some areas may be more "dense" than others.



task: have a smaller set of cabs to search through

city-id

→ get all drivers in Mumbai

50,000 cabs

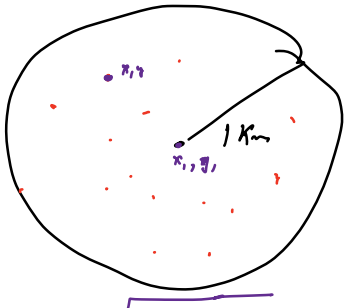
→ nearby cabs

- go through all cabs in Mumbai
- Compute distance to me
- Sort
- pick top 20

idea-1 ⇒ uniform grid

Andheri → 5 Km takes 2 hrs

Marine drive → 5 Km takes 10 mins



dist = $\sqrt{(x_1-x)^2+(y_1-y)^2}$ <= 1 Km

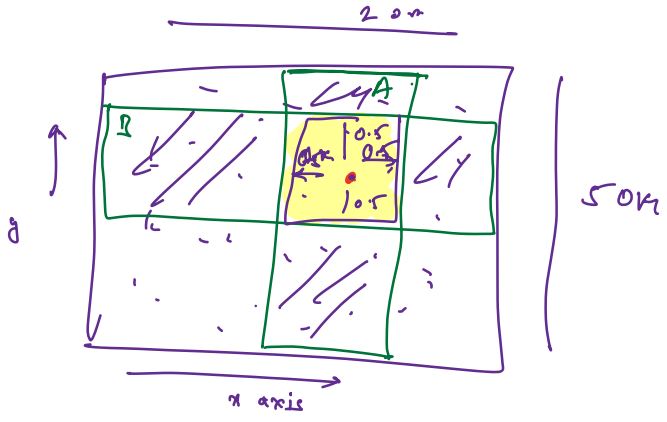
index on x
index on y
No!
only 1 index can be used.

```
select * from drivers
where x-50 <= driver.x <= x+50
and   y-50 <= driver.y <= y+50
```

can use an index!

alter table drivers add index on drivers.x,

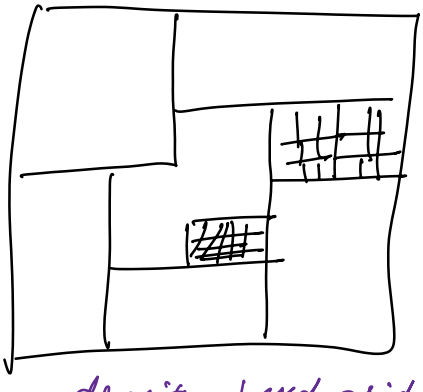
lower bound & upper bound.



AOI

Uniform grid → doesn't work.

dynamic grid



density-based grid.

density & available cabs

Realtime cab location data

↳ every 1 min driver's app will ping its location to Uber server.

↳ store it in Redis (in memory)

↳ key-value store (caching)

↳ My SQL database?

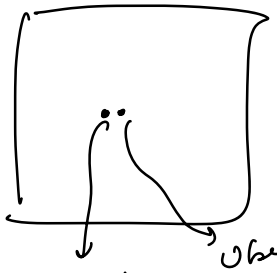
① change frequently? ✓

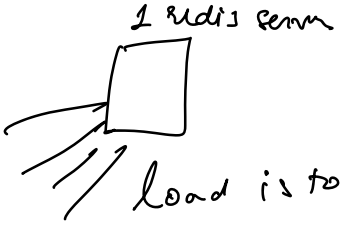
② consistency

↳ strong & immidiate

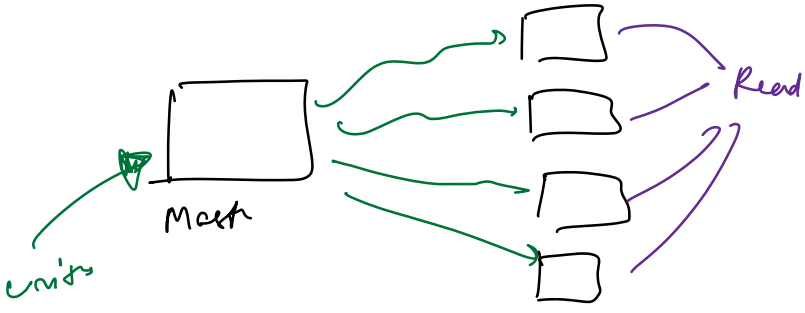
↳ eventual consistency

↳ some data loss → no consistency



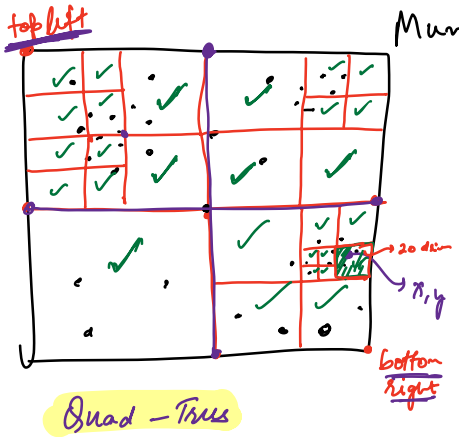


Master slave architecture



① created ✓
② find nearest drivers

tunable – consistency



Quad – Trees

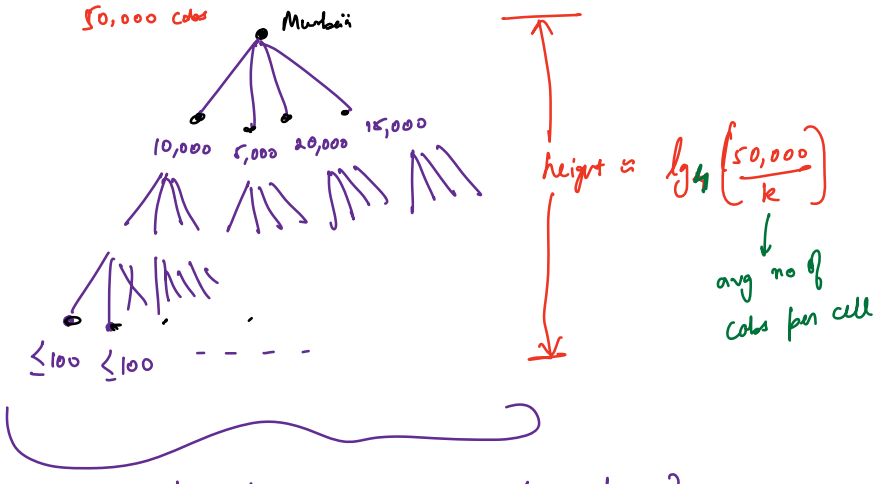
live data ✓

divide grids?

↳ ① sparse areas should have wider grid

↳ ② denser areas should have more granular grid

as long as any grid has 3+ points (drivers) in it, we will split it
↳ to make it smaller.



total no of nodes in the tree? separate

worst case → every point goes into a leaf node

$$\underline{\underline{50,000}}\ \text{leaf} + \boxed{\frac{50,000}{4}}\ \text{points} + \boxed{\frac{50000}{16}}\ \text{quadr} + \boxed{\frac{50000}{64}}\ \text{of} \cdots$$

$$50,000 * \left(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots\right) \longrightarrow \infty$$

G.P.

$$50,000 * \left(\frac{1}{1-1/4}\right)$$

$$\sum_{i=0}^{\infty} r^i = \left(\frac{1}{1-r}\right) \quad |r|<1$$

$$= 50,000 * \frac{1}{3/4}$$

$$50,000 * \left(\frac{4}{3}\right) \hookrightarrow \simeq 1.33$$

$$= 50,000 * 1.33 \text{ nodes}$$

$$= 70,000 \text{ nodes} \rightsquigarrow \text{easily fit in memory.}$$

① given x, y find the leaf node that it falls in → 50,000 drivers in overall grid

height of tree = $O(\log_4 50000)$ → 8 → 100ms

$4^? \geq 50{,}000$

$\approx 50 \times 10^3$

$2^6 \quad 2^{10}$

$4^8 \approx 2^{16}$

① 2^{34} approx

$\approx$ 16 Billion

② $> 16B$ ✓

$< 16B$

50,000 updates each min

$\frac{50{,}000}{60} \approx 1000$ updates per second.

① Drivers ping their location every 1 min

② we construct a Quad Tree based on this data

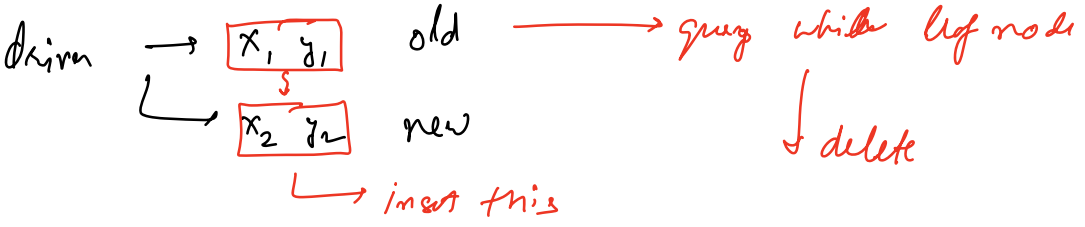
③ whenever user asks for nearby drivers we query this Quad Tree.

Do the drivers really need to ping every 1 min? Only if location changed!!

→ only if cab is not booked

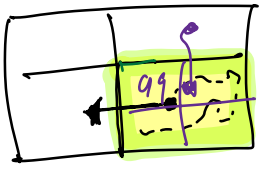
↳ if booked → not available

→ if location is not changing → don't ping.



1000 updates on quad tree per second → high

Driver asks to update QuadTree only if they move out of grid?



→ location updates ✓

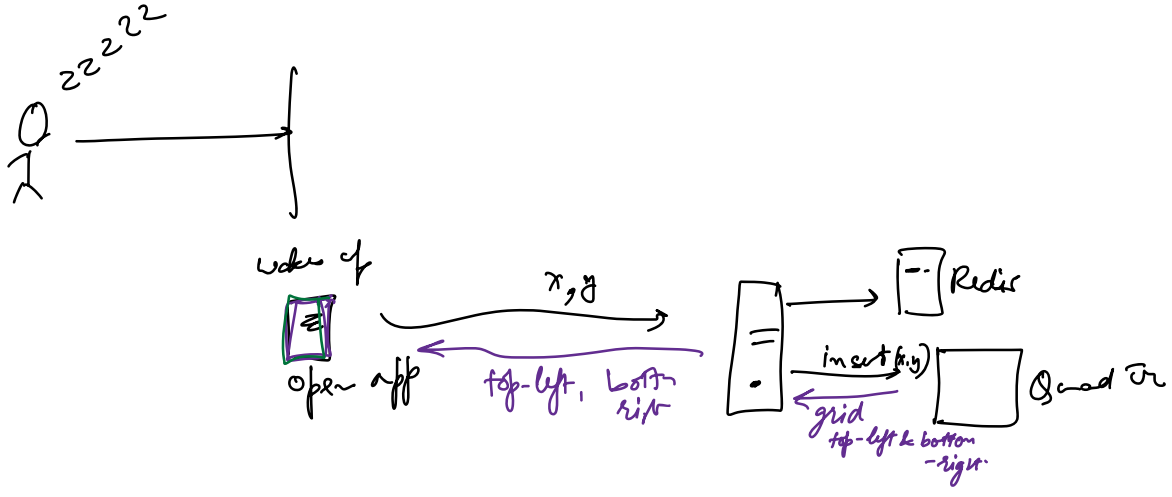
↳ every location update must not update QuadTree

→ quad tree will change only when driver crosses the grid boundary

→ driver can make

pingLocation (x, y) → every min

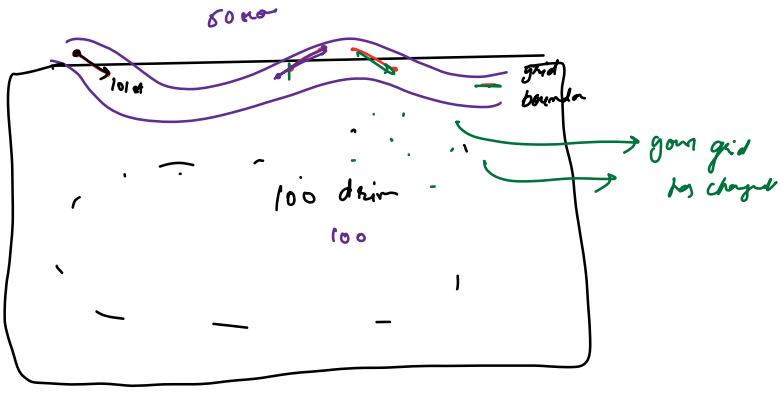
updateQuadTree (x, y) → when it moves out of grid.



```
if ( x < topleft.x || x > bottom right.x
   || y < topleft.y || y > bot r .y ) {
```

topleft, bottom Right ⇐ updateQuadTree (driver-id, x, y)

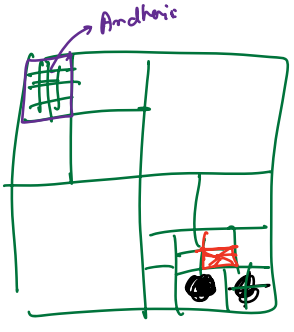
}



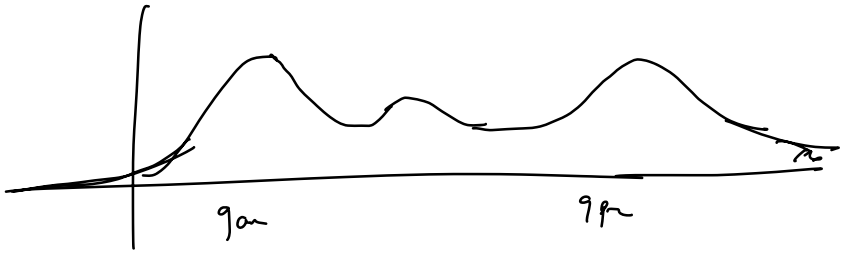
if some drivers is frequently changing grids
↳ a lot of grid splits & recombines

(1) limit the freq at which grids can change

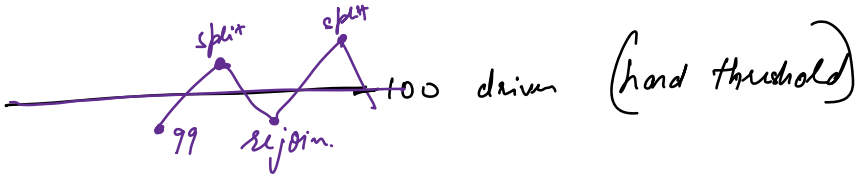
↳ time 12:00 noon → grid
↳ updateQuadTree at 1:00pm
↳ 2:00pm
↳ 3:00pm ✓

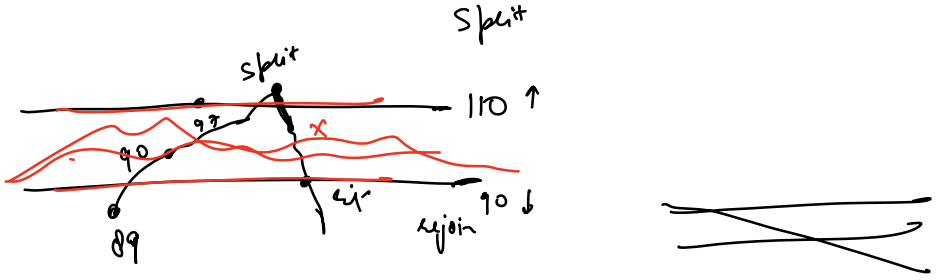


→ Strong control over how freq we wish to update grid!
→ from 9am – 9pm → ev 1 hr
→ rest → 3 hrs



② Widen threshold for split & re-joining grids.?

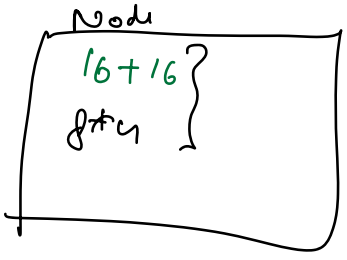


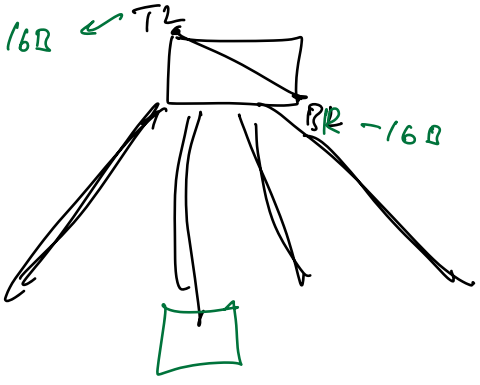


DSA, LLD, HLD → Create a repository of general problem solving techniques

50,000 drives

≈ 70,000 nodes in the





32+32 = 64 B

64 B + 70,000

50,000 drive id, 4B drive location 16B

50,000 (1 + 1/4 + 1/16 + --)

≈ 70,000

50,000 * 20B + 70,000 * 64B

~~100 * 10 B~~

10^5 B + 4.50×10^7

4.5×10^6

≈ 4.6×10^6 B = 4.6 MB ting