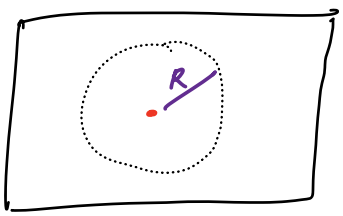
Agenda

→ Quad Trees

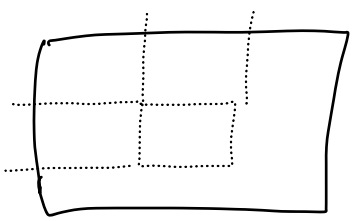
→ Nearest Driver (Uber)

Nearest Neighbor

Circle
↳ very slow



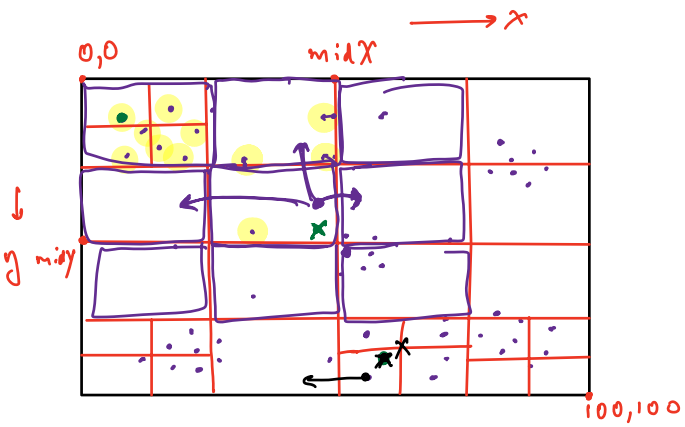
Rectangle
↳ slow



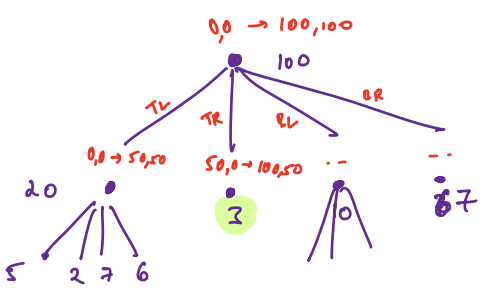
Grid → fast
↳ impossible to find perfect grid size

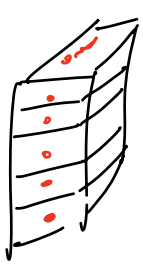
Quad Tree

any cell / leaf node will hold atmost 5 items.




```
Node creatQuadTree(Point2D TL,
                   Point2D BR,
                   List<Point2D> items){
  if(items.len <= 5){
    return Node(
      TL, BR, items
    )
  }
  check if overlapping items
  midX = (TL.x + BR.x)/2
  midy = (TL.y + BR.y)/2

  List<Point2D> TLitems = items.filter(
    i => (i.x <= midX && i.y <= midy)
  )

  TL items = ---
  TR items = ---
  BL items = ---
  BR items = ---

  TLchild = createQT({TL.X, TL.Y},
                     {midX, midY},
                     TL items)
  TRchild = --- ({midX, TL.Y},
                 {BR.X, midY},
                 TR items)
  BL child = ---
  BR child = ---

  Node parent = new Node()
  parent.tl = TLchild
  par.tr = TR---
    .bl =
    .br =
```



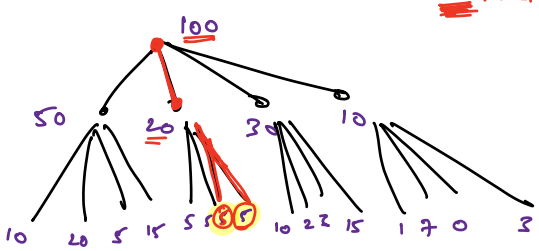


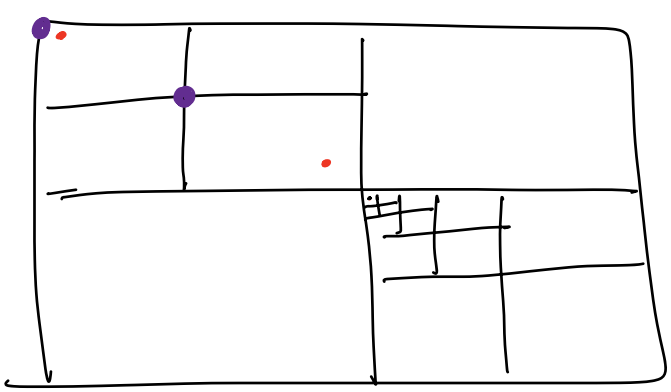
6 items with exactly same cordinates!!

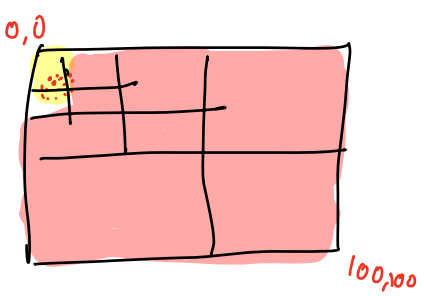
} ext point

find nearest Locations (x, y, k → 10)

10 nearest items to (x,y)







Node {

* TL → 8 bytes
* TR → 8
* BL → 8
* BR → 8

Count → 8

tl coordinates → 16

br coordinates → 16

}

72 bytes → 100 bytes

LeafNode {

List<Point2D, id> items;

neighborPointers[8] → 8*8

tl coordinates → 16

br coordinates → 16

}

100 bytes

In total → 100 million restaurants.

→ location 16 bytes

→ id 8 bytes

24 bytes

Total Space $100 \times 10^6 \times 24$ bytes + (# nodes) * 100 bytes.

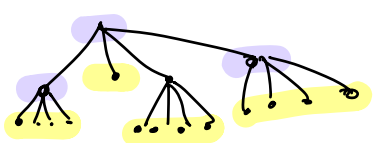
every leaf node has on avg 1 item.

leaf nodes = 100 million.

parent nodes = 100 million / 4

grandparent nodes = 100 million / 4^2

⋮



max possible # nodes = 100 million $\left(1 + \frac{1}{4} + \frac{1}{4^2} + \frac{1}{4^3} + \frac{1}{4} \cdots\right)$

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \qquad |r| < 1$$

$$\frac{1}{1-1/4} = \frac{1}{3/4} = 4/3 \approx 1.33$$

max possible # nodes for 100 million places = 1.33 * 100 million.

Total Space

$$100 \times 10^6 \times 24 \text{ bytes} + (\#\text{ nodes}) * 100 \text{ bytes}$$

$$\#\text{ nodes} = 1.33 \times 10^2 \times 10^6$$

$$10^8 (24 + 133) \approx 2 * 10^{10} \text{ bytes}$$

$$= 20 \text{ GB} \longrightarrow \text{RAM}$$

API for Uber → Intra-city rides

↓ within

Stakeholders

Passengers. Drivers.

findNearestCabs(x, y)

trackRide(driverId)

updateLocation(driverId, x, y) — new location

→ asyn job → sends all nearby drivers a push notification ACCEPT/REJECT

↓ find driver that accepts gets the booking.

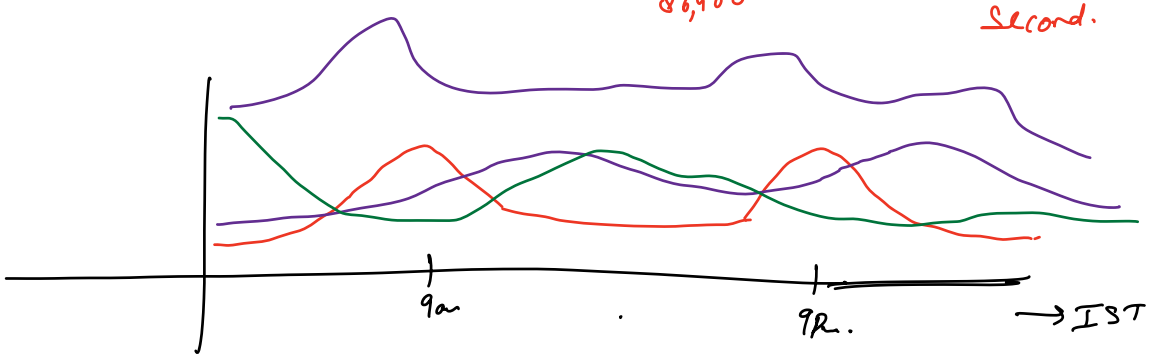
→ mark driver as unavailable untill ride ends

Scale of Uber

large city → 50,000 cabs

10 billion bookings per year ≈ 30M booking per day.

≈ $\frac{30 \times 10^6}{86,400}$ ≈ 400 bookings per second.



Sharding Key → city_id

↓

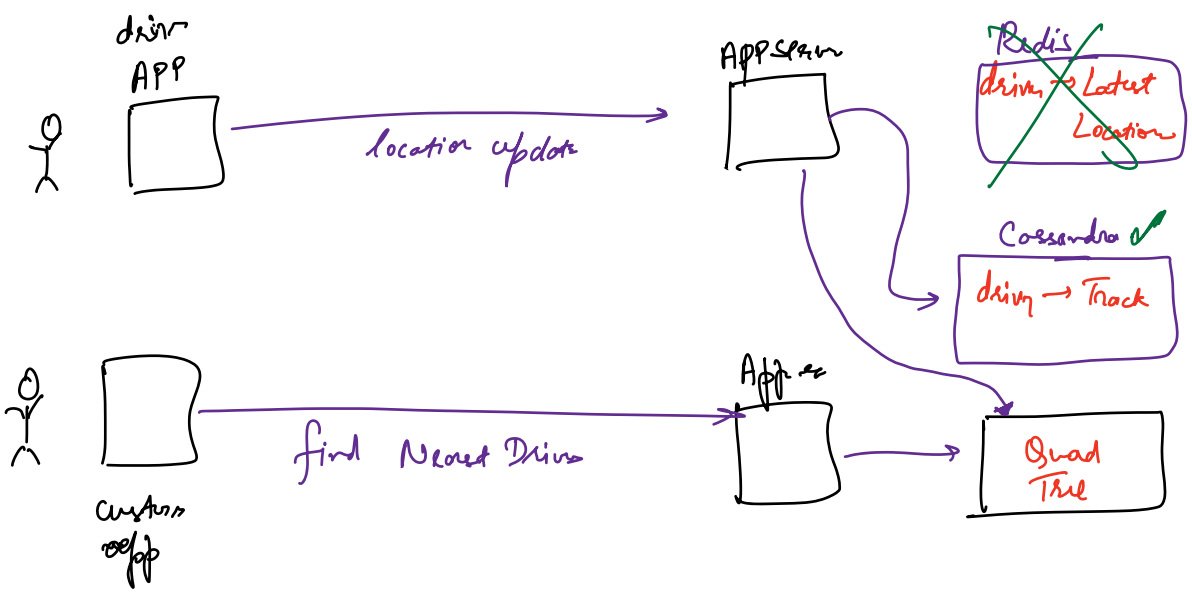
upto 50,000 driver active

Quad Tree for each city.

10:22 → 10:30

User data → Name
Photo
Address
Phone
Car

Driver → ~ 5 M drivers in entire Uber.



Optimizing Location Updates

50,000 drivers

2.5×10^4 req/sec

5M drivers

2.5×10^6 req/sec

↳ Update every 2 sec

→ decide how frq should driver update → once every 20 sec

60 Km/hr

$60 \times \frac{5}{18}$ m/s

16 m/s.

→ if drivers location has changed significantly in the last 1 min → only then send location updates.

2.5×10^5 req/sec

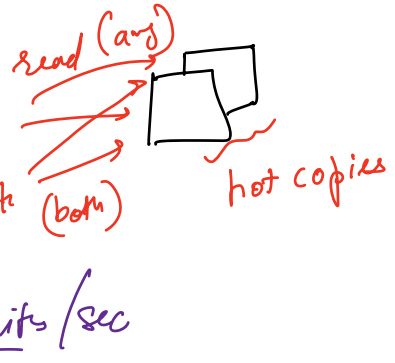
2500 req/sec

≈ 1000 req/sec

---

find Nearest drivers → read → Quad Tree

20 times/sec



1000 writes/sec

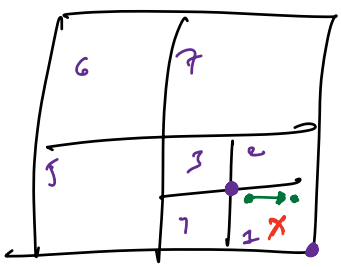
(1) Update Quad Tree at specific intervals.

every 1 hour lets update the tree

1 time / sec

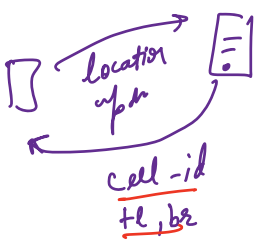
(2) as long as the driver remains within the cell there is no need to update drivers location.

→ grid cannot be dynamic!!!



drivers_app →

cell-id, tl, br



if location with tl, br → do nothing

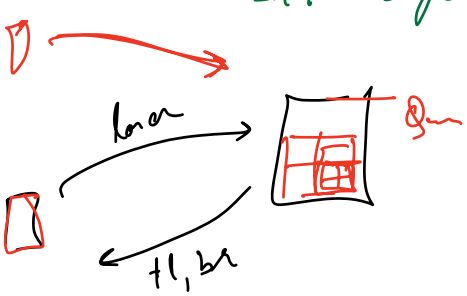
else

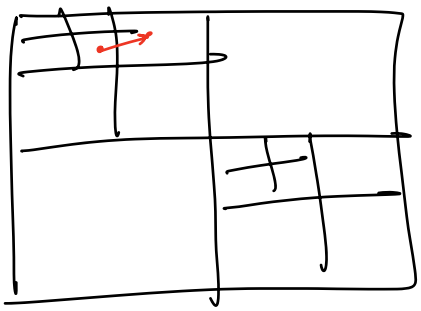
send_location_update (cell id, x, y)

old cell id

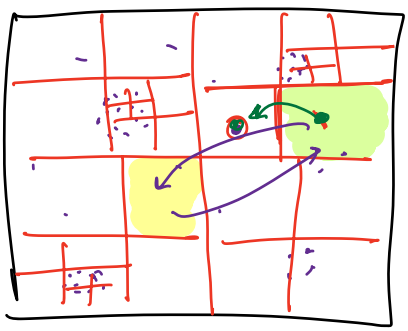
new location

System responds with new cell-id, tl, br.





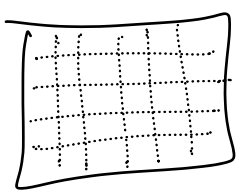
Mumbai



drivers inside leaf node changes →

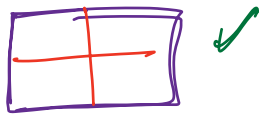
Structure of the tree remains same

↳ no nodes are split or merged!



if we make grid structure static → it will not be able to react to density changes.

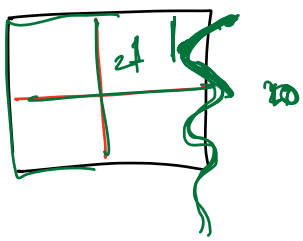
→ every 3 hours → recompute the grid structure



driv <=5

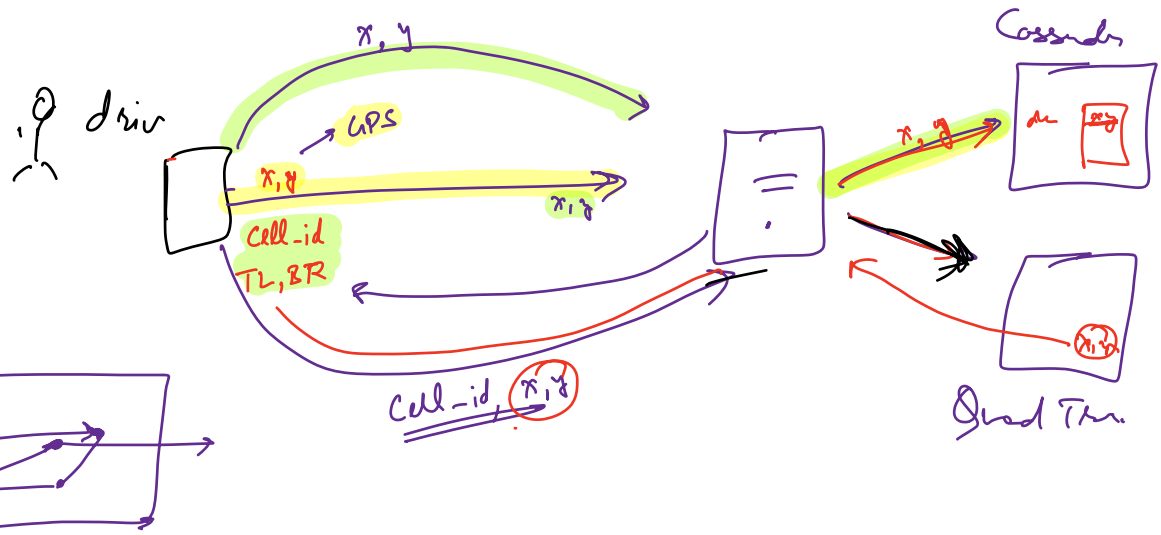
>5

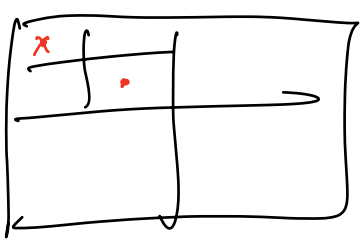
<5



if no of driv >20 → split cell

if no of driv <10 → merge cells



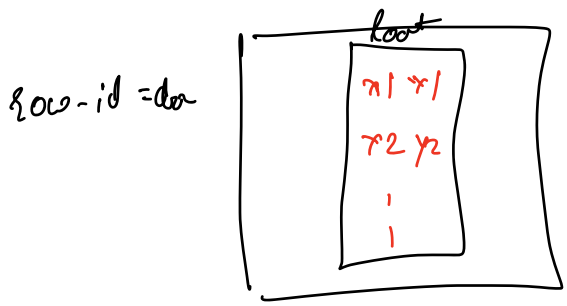


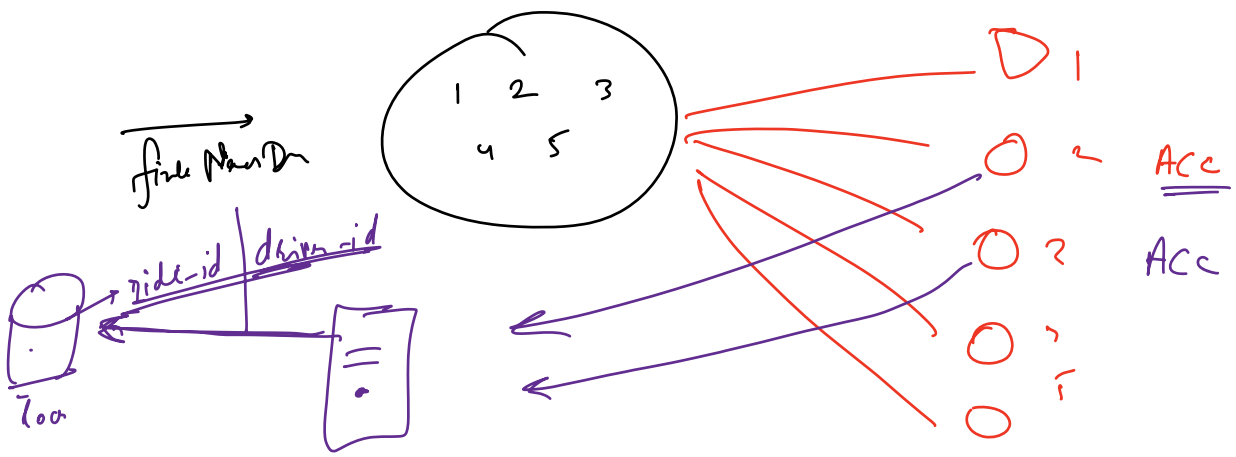
latest location

Key	Value
driver_id	x, y

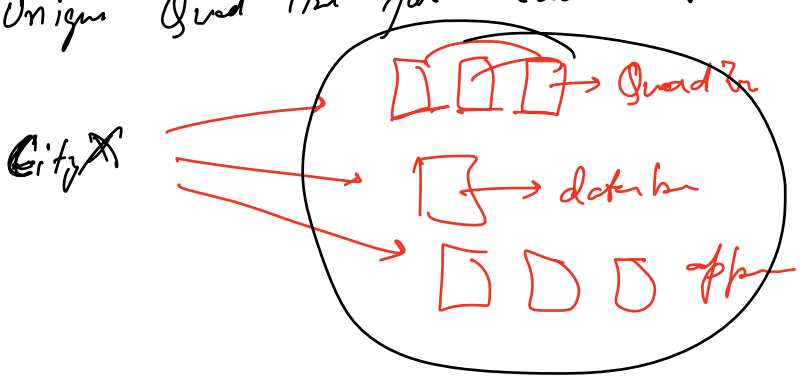
Trace

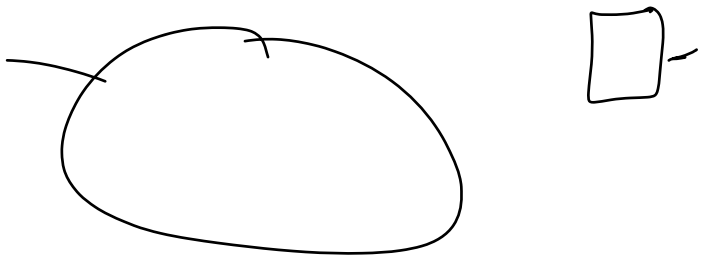
Key	Value
driver_id	[x1y1, x2y2, x3y3]





Unique Quad Tree for each city.



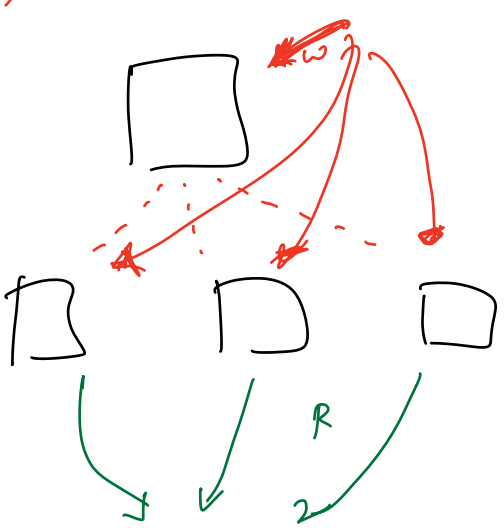


10 secs of drawing time

draw.io

excalidraw.com

R+W ≥ X



high consistency

read/write heavy

↳ split into diff services

if reads don't dep on the heavy writes

↳ heavy sharding

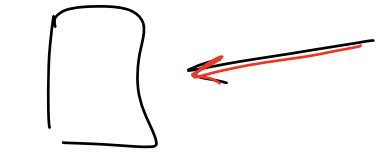
IRCTC

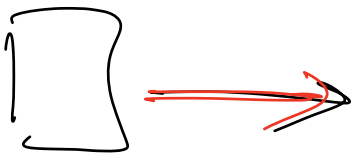
Cascading failure

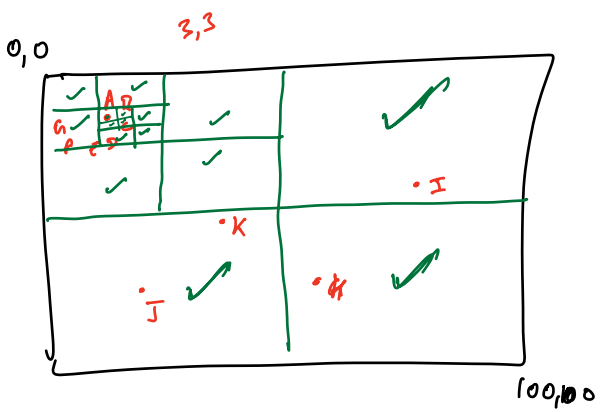
↳ 1 syst fails

{

other syst to other







threshold ≤ ≤

↳ leaf