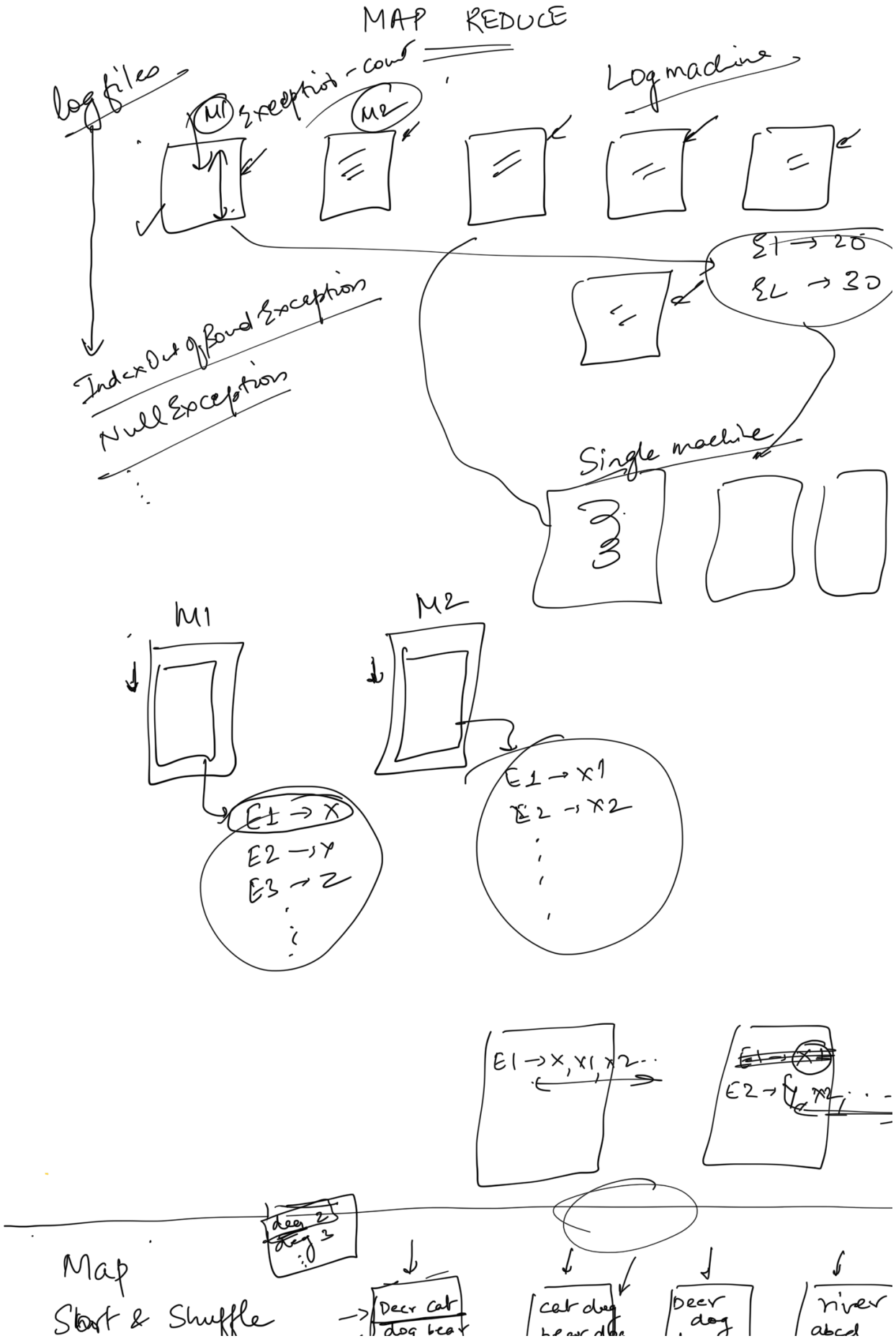
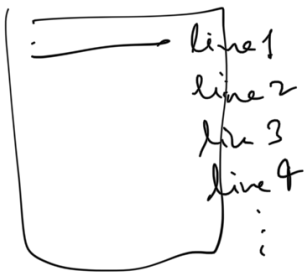

MAP REDUCE
log files
M1
Exception - count
M2
Log machine
Index Out of Bound Exception
Null Exception
E1 → 20
E2 → 30
Single machine
M1
M2
E1 → X
E2 → Y
E3 → Z
E1 → X1
E2 → X2
E1 → X, X1, X2...
E2 →
Map
Start & Shuffle
Deer Cat
dog bear
cat dog
deer
dog
river
abcd

Reduce

word → count



① Map function will run in parallel on all machines

② map (line_of_text)

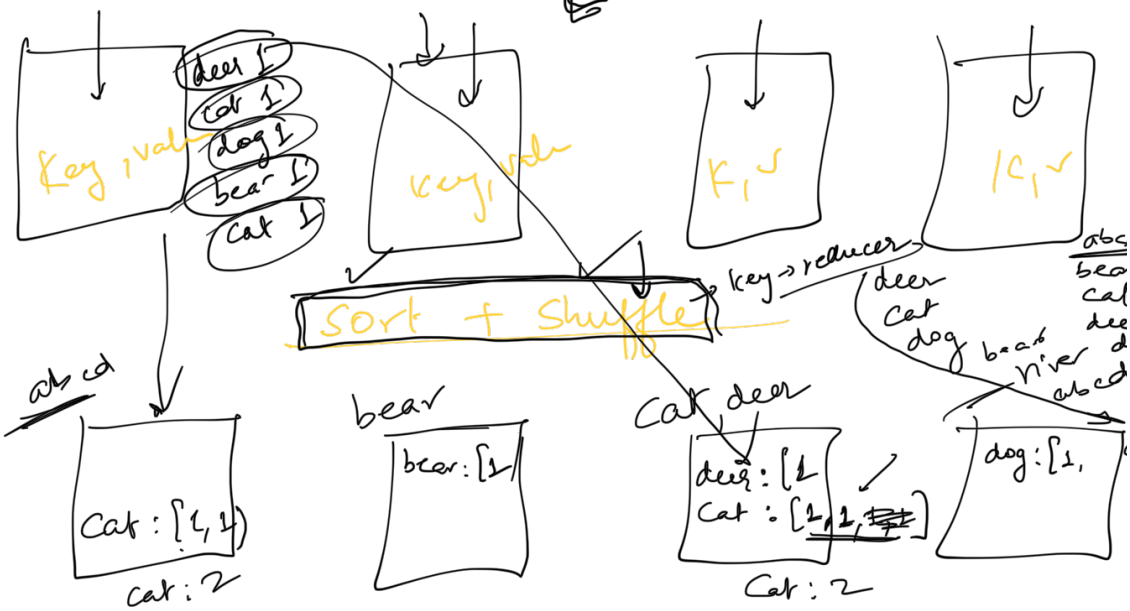
↳ list of (key, value)

map (line_of_text):

words = line_of_text.split(

for word in words:

emit (word, 1)



Reducer

reduce (key, list_of_val

for value in list_

sum += value

return key, sum

Map: map (key, value) → (key, value) 0 or more

Line → (key, value)

Reduce :

reduce

key : [v1, v2, v3, ...]

↳ value

Sort + Shuffle

Reducer

key : [...

Keys

a b c d e f

1 (1) Local sorting of keys in every mapper

k1, v1
k2, v2
k3, v3
kn, ...

Apache pig

M1 M3

global min key

global max k

Reducer

key1 → key2 → gmin → keyx

1 →

2 →
3 → 10,000
4 →
5 →
6 - -
i

(length_of_word, #)

((key, val

map (line):

length of word
& count

"dog cat deer bear"

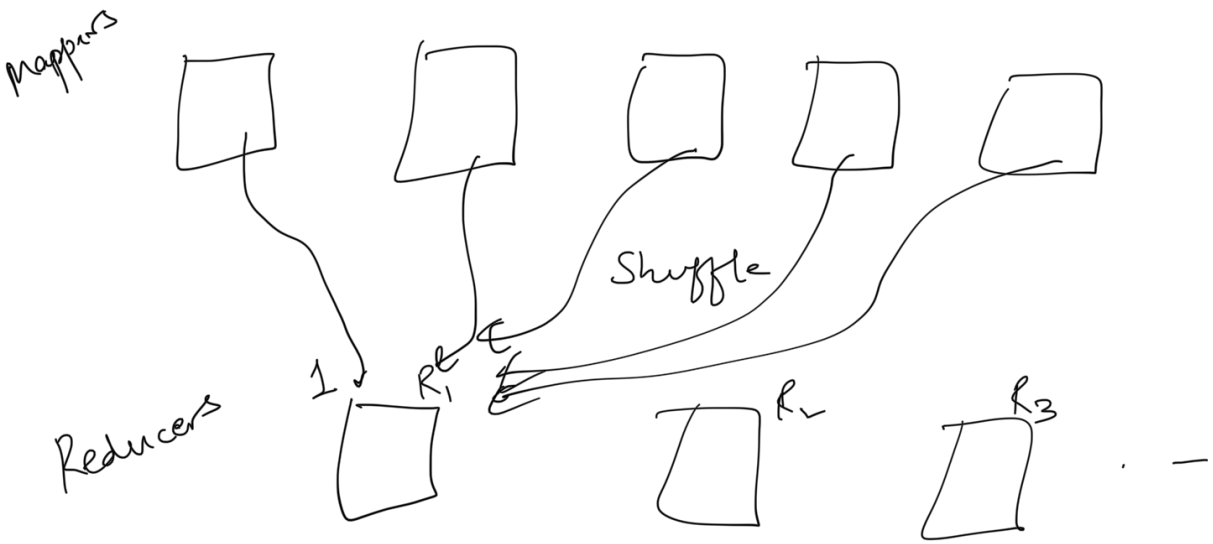
reduce

key, (list of value → sum

3 → 2
4 → 2

2 : [10, 2, 1, 15, ---

key 1: 12>
2: 50>
3: =



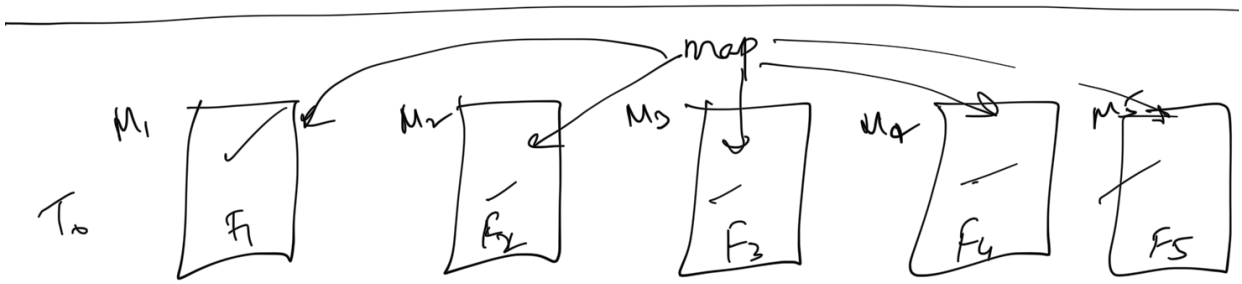
key, 1

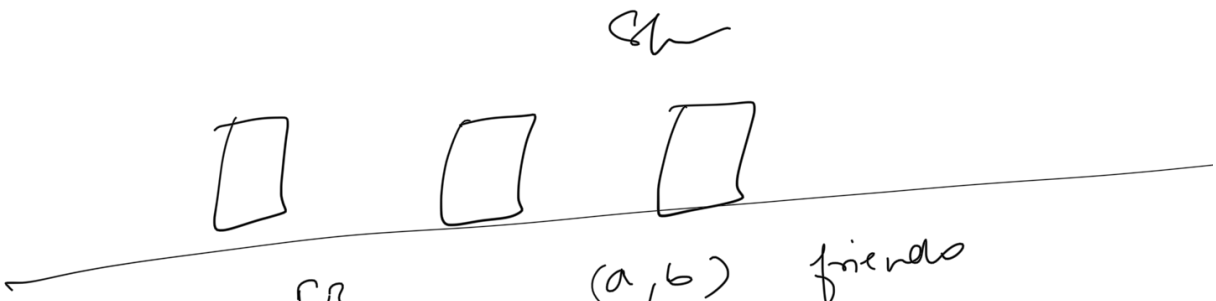
map (line_of_text)

for word in line_of_text.spl
emit len(word), 1

count = { }
for word in --- split:
count[len(word)]+

for K,V in ~~t~~ iterable counts.



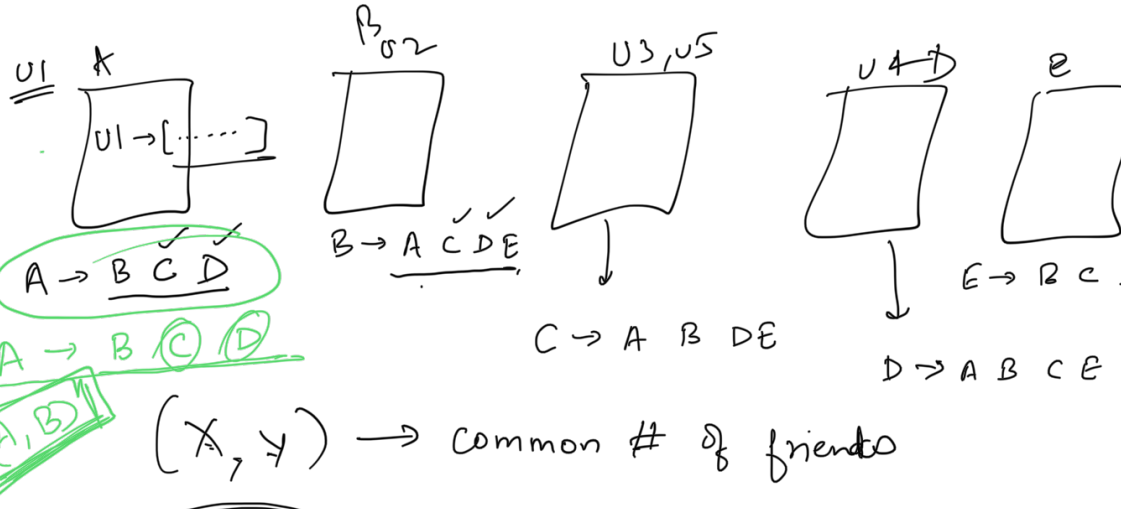


FB (a, b) friends

↓

X common friends.

user_id



(X, Y) → common # of friends

(A, B) → (C D) → 2

(B, C) → (A, D, E) → 3 ✓

(U1, U2) → # count of mutual friends

map (line-no-text) {

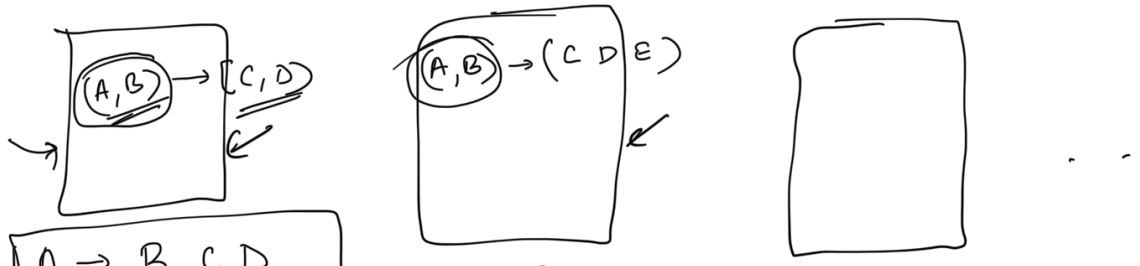
U1 U2 U3 U4

U5 U10 U15 U16

}

reduce(Key, list_of_value) {

}



(A, B) → 2

(B, C) → 3

(A, B) → [(C, D), (C, D)

(C, D)

(A, B) → [Friends of A] ← m1

[Friends of B] ← m2

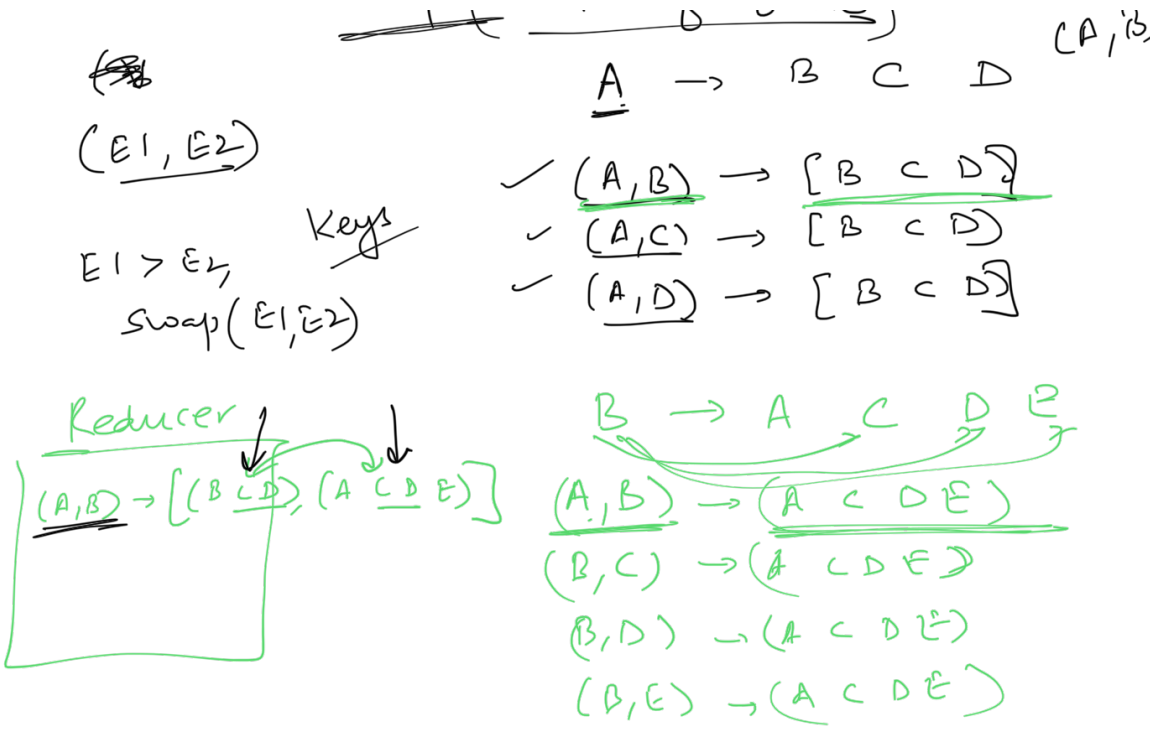
(A, B) → Friends of A

(A, B) → Friends of B

(A, B) → [(Friends of A), (Friends

reduce(key, list.)

map(line_of_text) : (A, B)



reducer (key, list_of_value) :

l1 → list_of_value[0]

l2 → list_of_val[1]

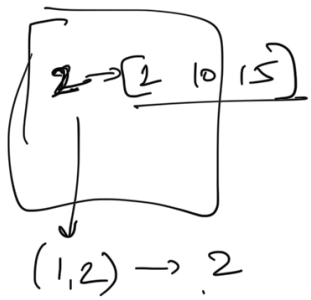
return key, intersection_of (l1, l2)

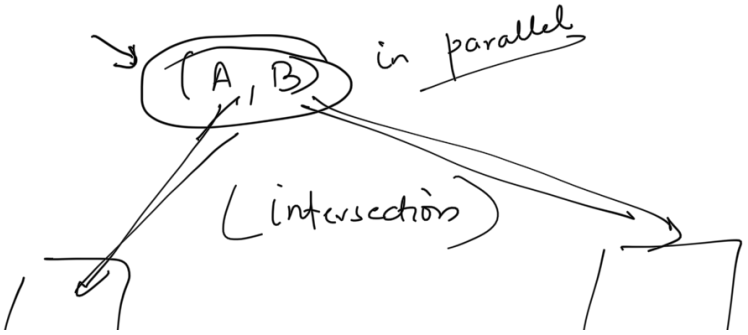
---

(1) → (2 3 4)

key (1,2) → 2

(2)





① More time

② Framework

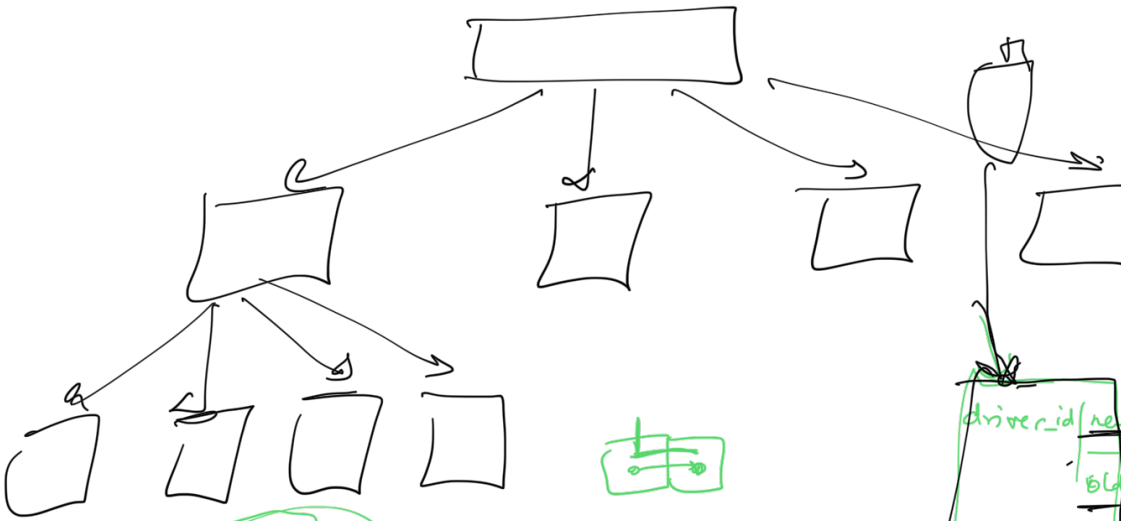
Quad Trees

find nearest neighbor

places were static

UBER/OLA → User → nearest taxis

moving



once every 5 mins

30 mins

No update

tile → taxi

Batch updates

500 taxis

0s

5mi

① Driver changes

expensive

① Deletio

① Additio

tile → [taxi1, ~~taxi2~~,

tile2 → [taxi2,

Sharding based on city

50,000

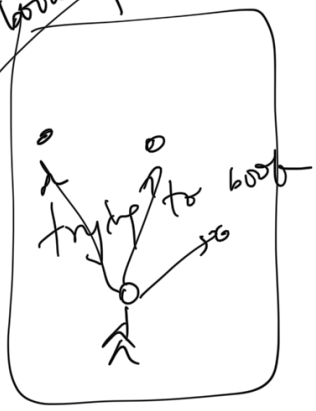
BLR 1 hour

① Given my location → find nearest 10 taxi

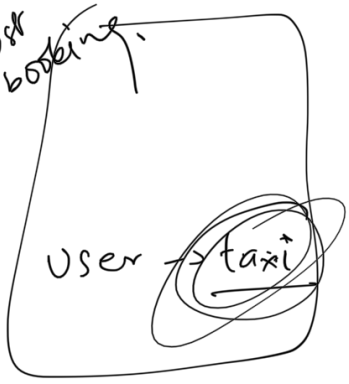
↳ Step #1: Find 100 candidates for nearest 10 → tile → taxi's

Step #2: Iterate on candidates and find 10 nearest
↳ taxi's real location

Pre-booking

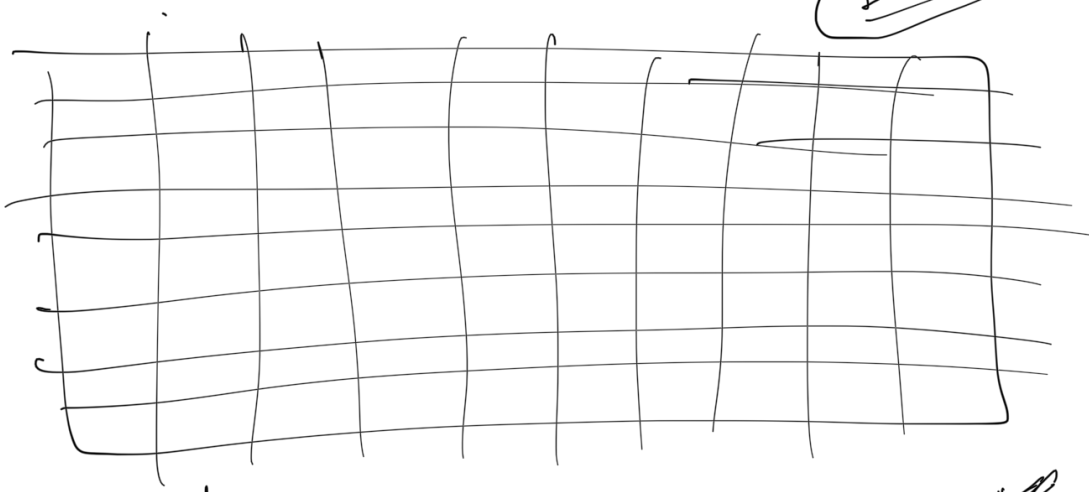


For booking

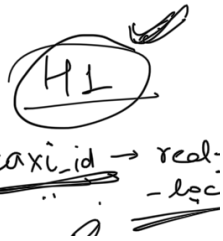


taxi_id
↓
current_loc

1 million grids



(stale)
Quad tree

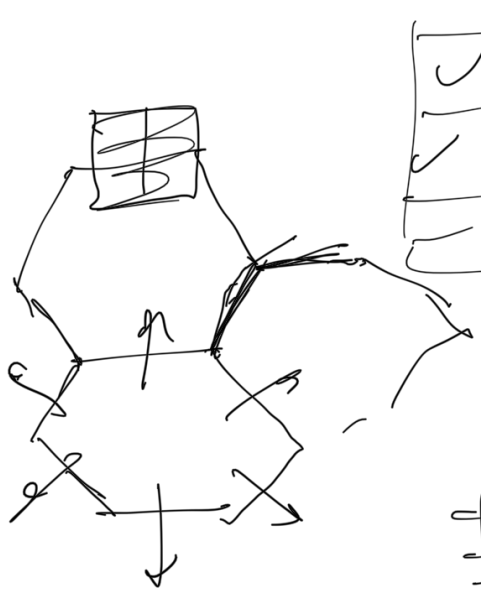


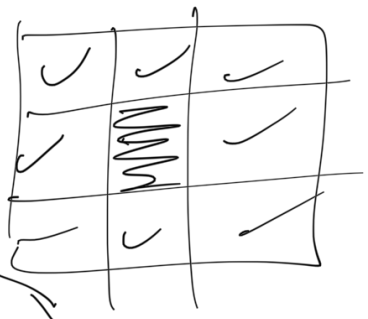
tile → taxi_id

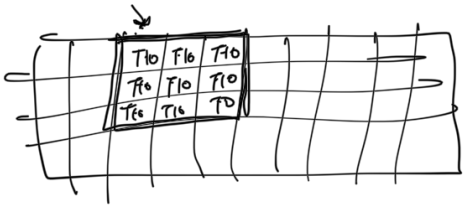
tile2 → (- - - taxi Candidate

tile0

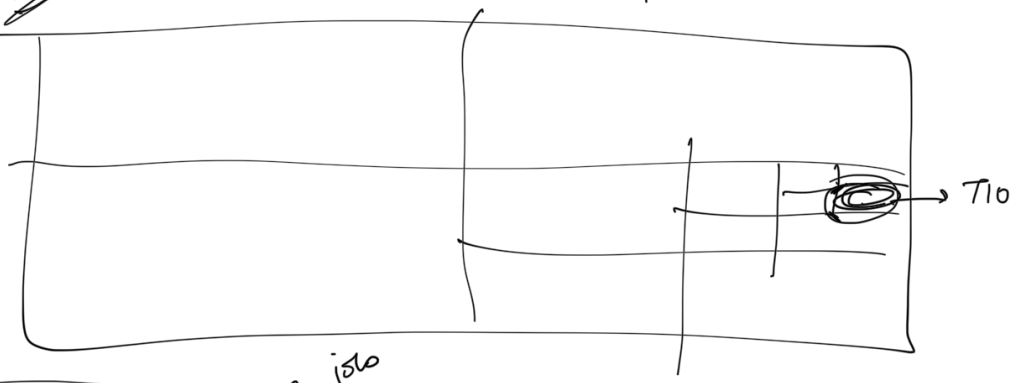
{ 100 taxis.
50 taxis }







key value
grid → Tile



L1 → tileno

L2 → tileno

Async job

H1

Driver → L2

UBER

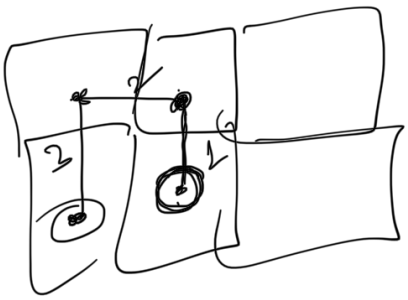
Driver

22.1053

21.1023 44

1 million 3s.

L2 new_doc



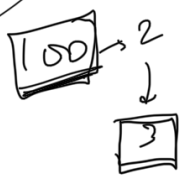
HL

tile1 → [

tile2 → [~~100~~,

tile3 → [100

tile4 →



✓ ① log N → ~~leaf~~

→ ② list deletion

✓ ③ log N

→ ④ Adding taxi to new location

→ ~~Create~~ # of points > threshold