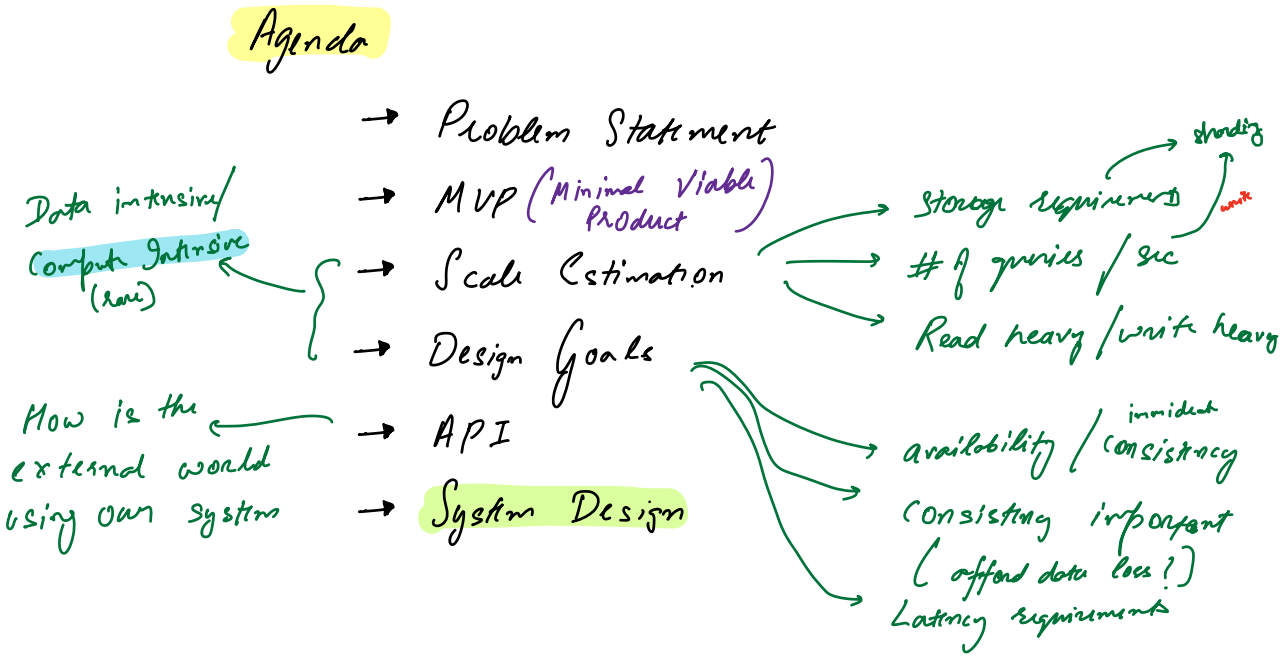
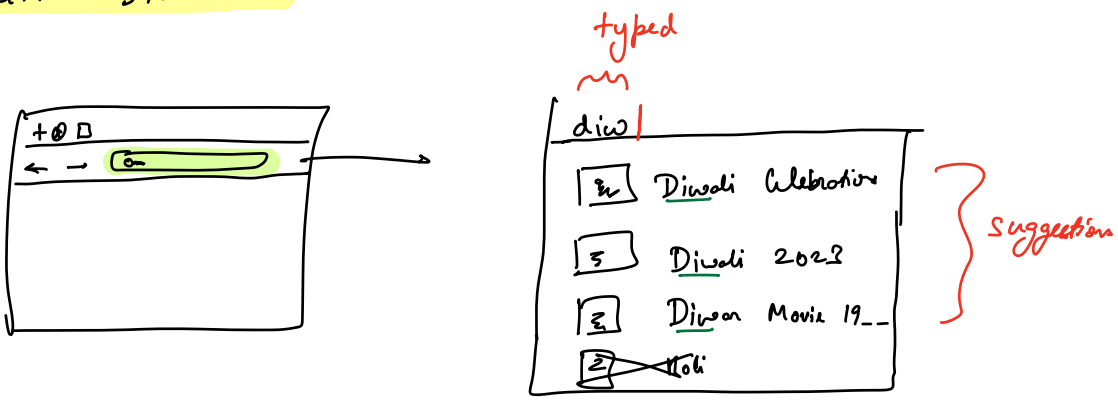


Case Study 1 → del.icio.us

Design Interviews → free flowing → discussion based

(1) Problem Statement



How do you design a type ahead System

Scale: Google

Engineering Architect @ Google → Sawan

(2) Minimal Viable Product

(1) Maximum # of suggestions

5 is more than enough

(2) What should we suggest

↳ popular searches

(3) Personalization

not important for MVP

④ If no relevant results, should we show some (irrelevant) results anyway?

No → show only relevant results

⑤ Fuzzy matches / spelling errors

→ not needed for MVP

→ strict ~~prefix match~~
(begins with)

⑥ Rank results

Popularity of queries
Search (diff from typeahead queries)

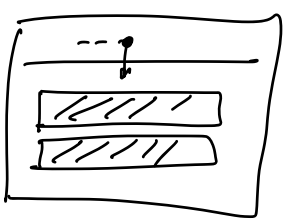
geo-personalized
Not for MVP

Show more recent results on top?
not in MVP

⑦ Min number of letters that must be typed before we show typeahead

3

⑧ Show suggestion for every keystroke



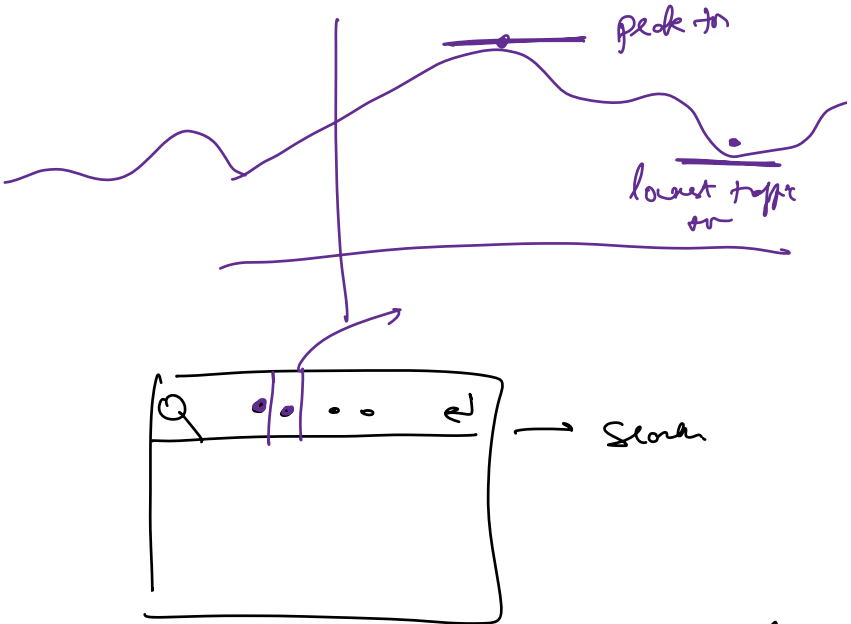
(3) Scale Estimation

Volume of queries

→ ≈ 10 Billion Search queries per day

($10^{10} / 86{,}400 \rightarrow \frac{10^{10}}{10^5} \approx 10^5$ searches/sec)

100,000



→ 7 typeahead queries for each search query.

typeahead

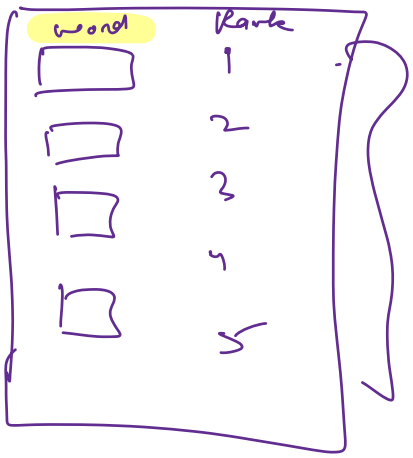
~10 letters long (avg search query)

first 3 letters no typeahead
next 7 letters → typeahead

→ 7×10^5 typeahead search queries/sec

→ 1 million typeahead/sec

Scale of data → do we need sharding

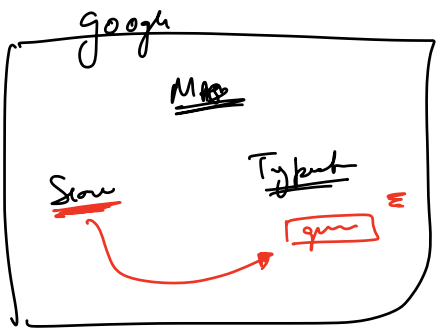


10 B qu/m

32 bits 8B

Search queries	# of hits
Dicaprio	2 Billion
Roger Fed	100 M
Holi	800M
Diwali Mov	2 Million

→ 32*2B + 8B <= 100 B per node



10 B Searches per day

↳ # of unique queries per day

↳ never been searched before

~ 15% queries are unique

ask interviewer what % of queries are unique?

80:20 rule

unique

duplicate

10 year of Post data

~~10~~ * 365 * 10 * 100 B

1.5x10^9

1.5x365x 10^12

$1.5 \times 2.65 \times 10^{14}$

$6 \times 10^{14} \rightarrow 10^{15}$ Bytes

Giga $\rightarrow 10^9$
Tera $\rightarrow 10^{12}$
Peta $\rightarrow 10^{15}$

≈ 1 PB

$\approx$ 1000 TB of data

is Sharding needed?

→ (1) data is too large
→ (2) write queries many

Yes!!

Jonathan Linus Tech Tips
100 TB server.

Read heavy / write heavy

10 B searches / day → write $\Rightarrow 10^5$ searches/sec

$\approx 10^5$ writes / sec

$\approx 10^6$ reads / sec

Both read & write heavy

(4) Design Goals

→ Availability vs Consistency

eventual consistency is okay

Hol → Holi → Holiday

→ do we need consistency at all? No!
(can we deal with data loss)

We can afford small data loss

actual → 800 M
second → 799 M hits

→ Consistency vs Latency

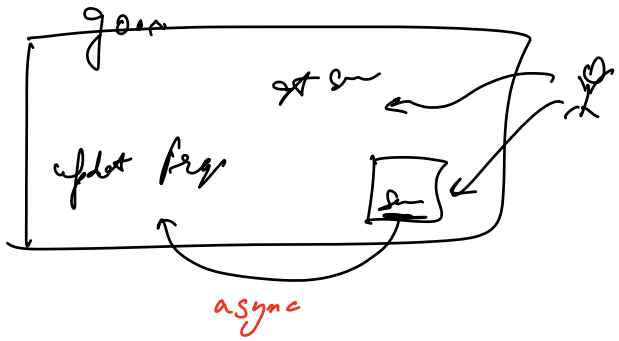
latency should be very low

(6) API → how will outside world use our system

anything except our service → users
↳ other services at google

(1) get Typeahead Suggestion (partial_query)
user ↳ upto 5 results

(2) update frequency (search_query)
google's internal service



10:42 → 10:50

Designing Typeahead

Updat freq (search-term)

Search servic → async → Pipeli- → Consum

updat freq

Key - Value

Maintain

read →

Search Q	#Hits

X won't work for read queries

getTypeaheadSuggestions (prefix) → top 5 results

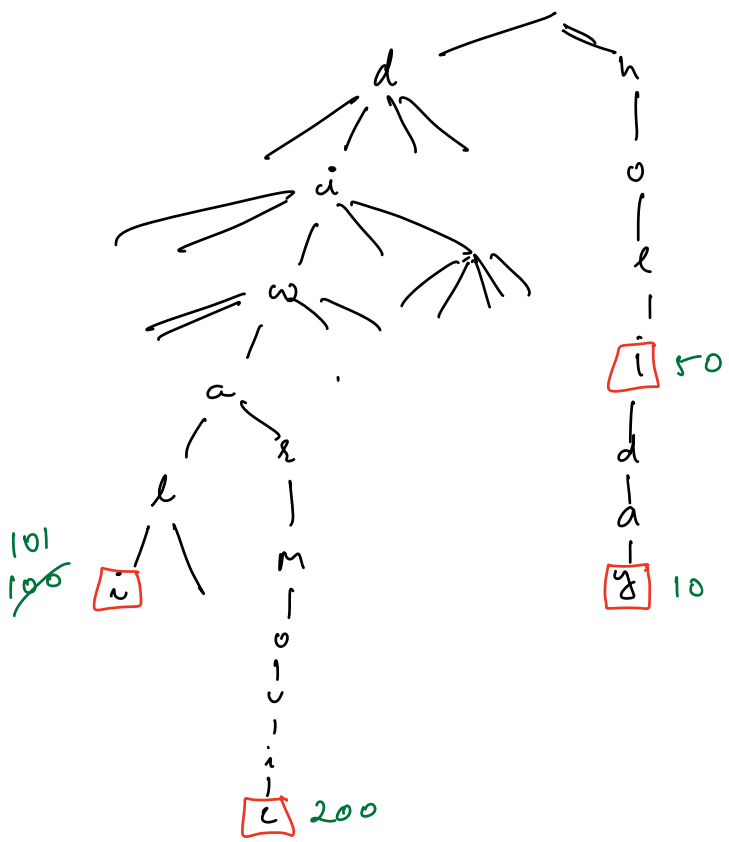
avg 10 letters

$(26+10)^{10}$

36^{10}

$36 * 36 * 36 \ldots$ (10)

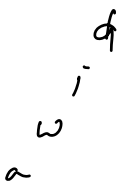
Trie Approach — Construct a trie of search queries



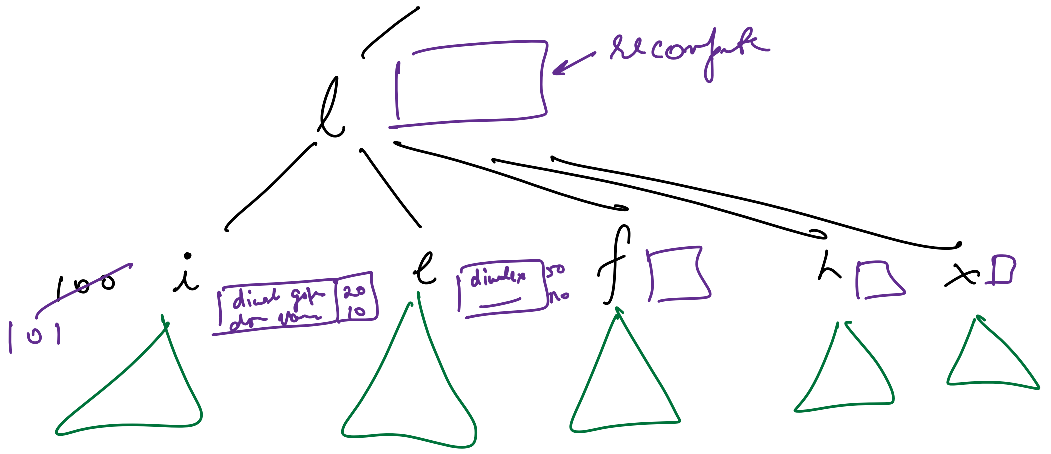
node:
children [36]
is terminal
hits
top 5 results with counts

getTypeaheadSuggestions (partial)
diw

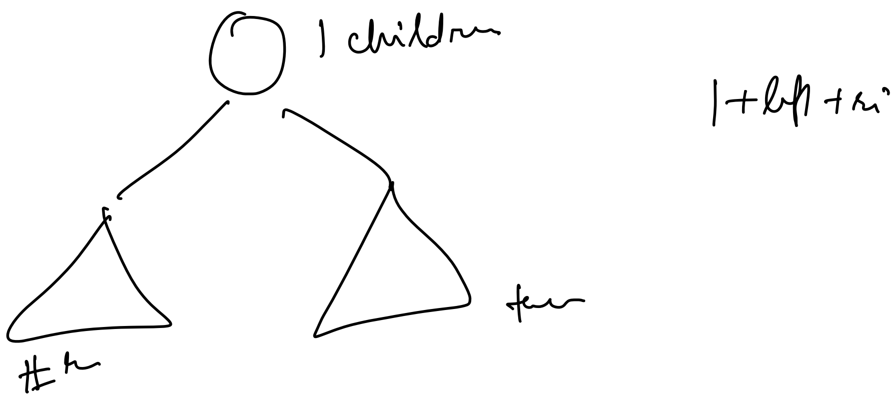
updateFreq (search-term)
diwali
32 digits



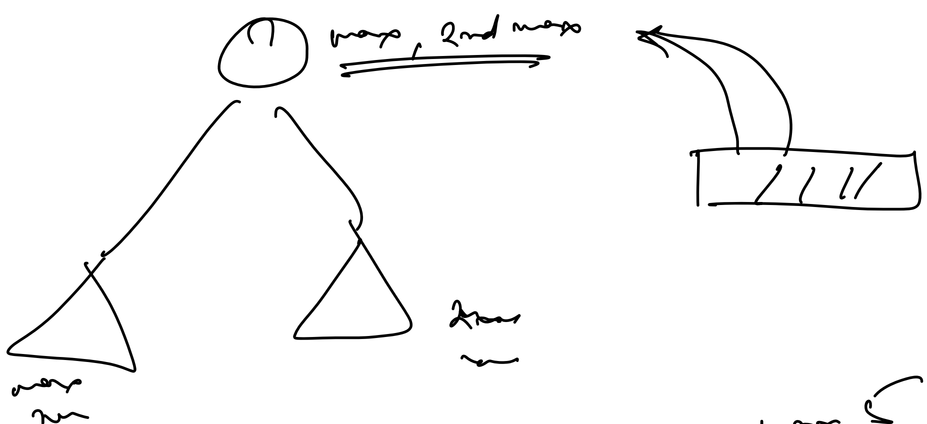
diwali



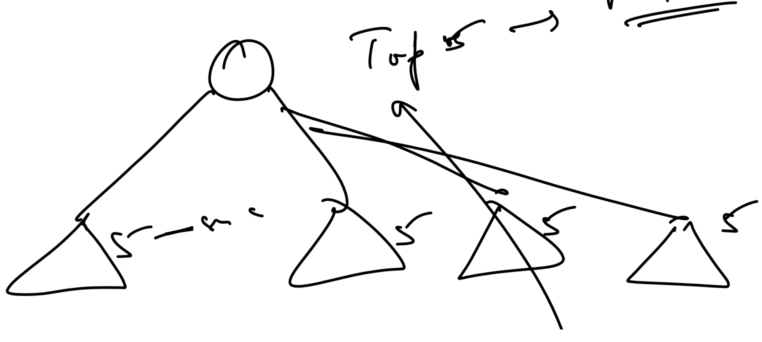
recompute
101
20
10
50
110



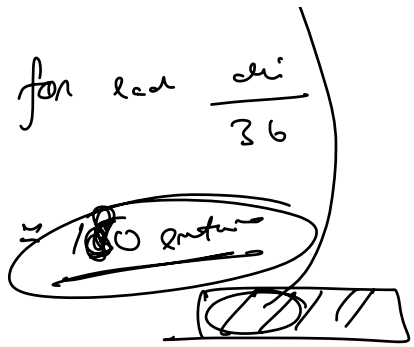
1 children
1+left+ri



max, 2nd max



Top 5 → max 5

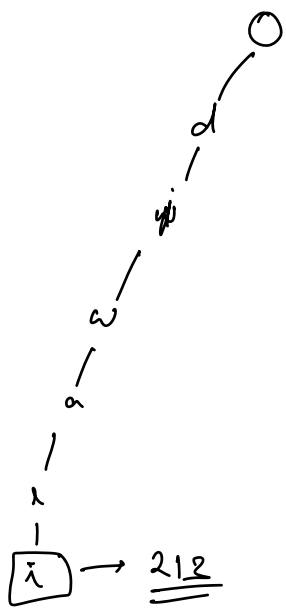


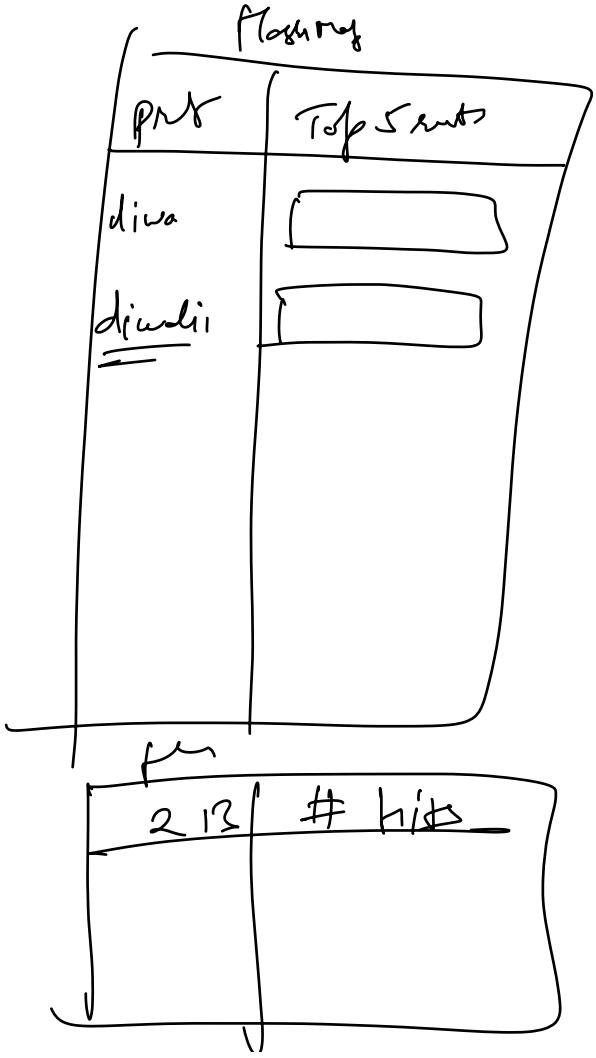
top 5 hits

i

diwali gifts, 100
diwali vocab, 50
diwali ca, 10

Trie + HashMap





HashMap Approach

H1

partial query	top 5 hits
d	[dim — , — —]
di	[— — —]
dim	[— —]
diva	[— —]
diwali	
Ro	

H2

Search query	# Hits
diwali	100/101
diwali ...	50
Rom...	20
⋮	

getTypeahead Suggs (prefix) → lookup in H1

updateKey (diwali) → H2 → update count
H1 → update top 5

d
di
diw
diwa
diwal
diwali

top 5 may change

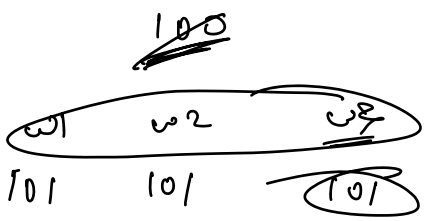
Trie → Prefix match
space optimization.

Key - Value Store is automatically sharded based on key

Optimizing Writes

10x more reads than writes
↳ updating multiple things
count
each prefix → 30 prefixes

Write → prevents reads → prevent stale reads
not important for this system
↳ prevent writes → always needed



---

(1) Sampling

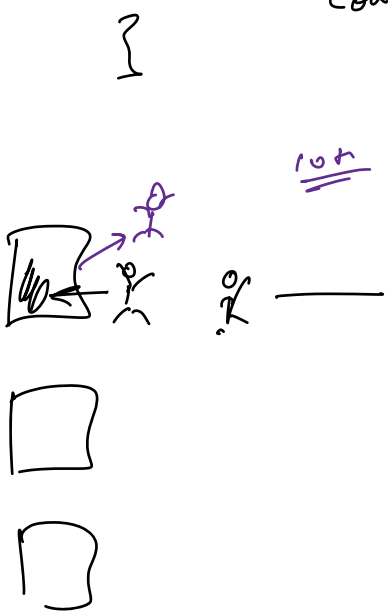
do we care about exact counts? ✗

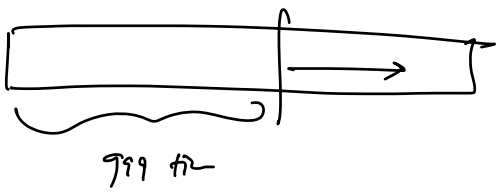
we will write only for every 100th search
1000

```
count = 0
updateFreq (searchTerm) {
    if (count == 100) {
        actually update
        count = 0
    }
}
```

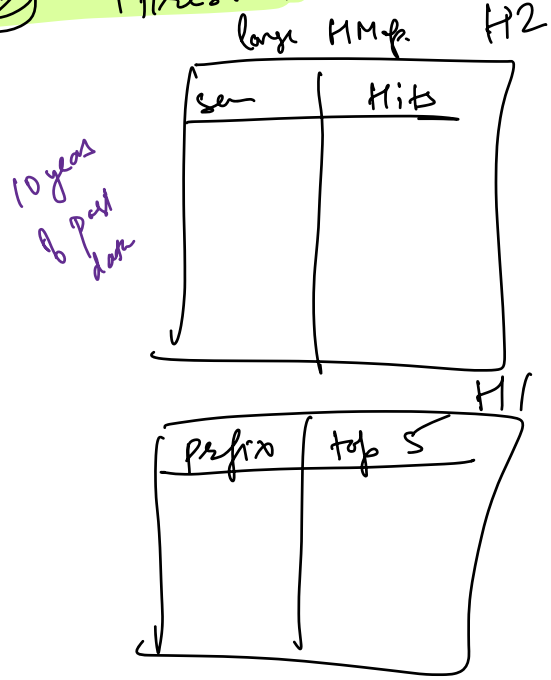
100M
1M

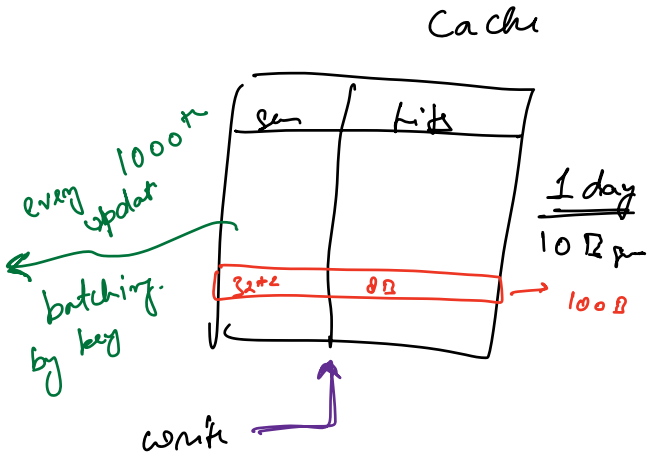
Count ++





② Threshold (batching)





100 Byte * 10^{10}

= 1 TB of data

Search diwali → diwali = 0

└→ cache → update diwali count ++

2 → upto cache

3 → ... ...

1000 → update cache → diwali count

diwali = 1000

Threshold

H1. update (diwali, 1000)

H2. update (all prefixes of diwali)

diwali = 0

Recency factor

Rodger Sanders → Top result

Rodger Federer → Wimbledon → Rodger Federer

Time Decay

Hits

↓ decay at a fixed rate

decrease hit by 10,000 after each day

diwali → 10M

diwali ... → 1M

cerebral palsy → 1000

cerebral fluid → 50,000

0

after each day decrease each count by 1% or 10%

and also by 100 → reduce for odds

diwali → 10M → 9M → 8,999,900

diwali celebrate → 1M → 900,000 → 899,900

cerebral fluid → 50,000 → 45,000 → 44,900

cerebral palsy → 10,000 → 9,000

[illegible] → 50 → 0 → 900

at day end

(1) decrease each value by 10%

(2) decrease each value by 100

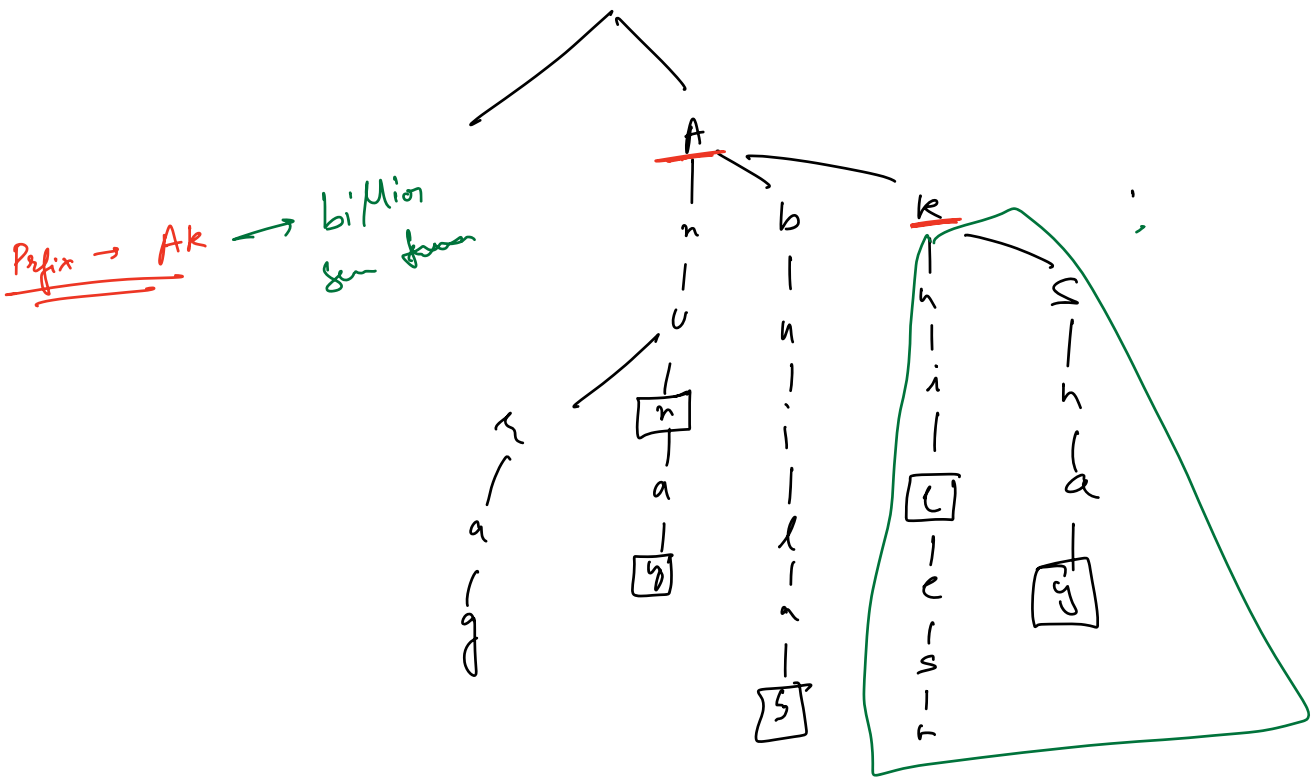
1000 reads per sec

10 words per sec

1M reads

10,000 words / sec

initially → process ≈ 1000 machines

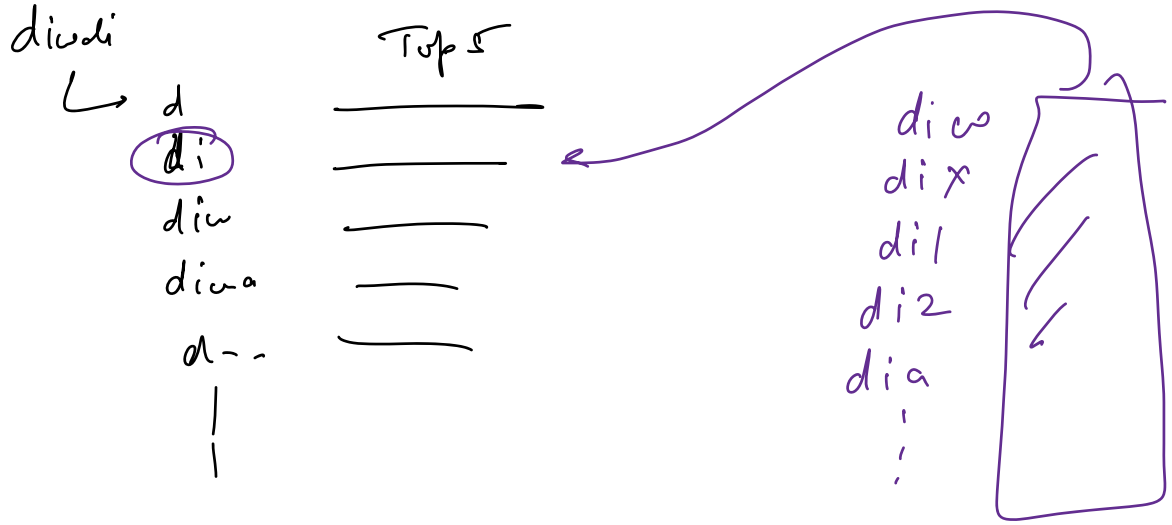


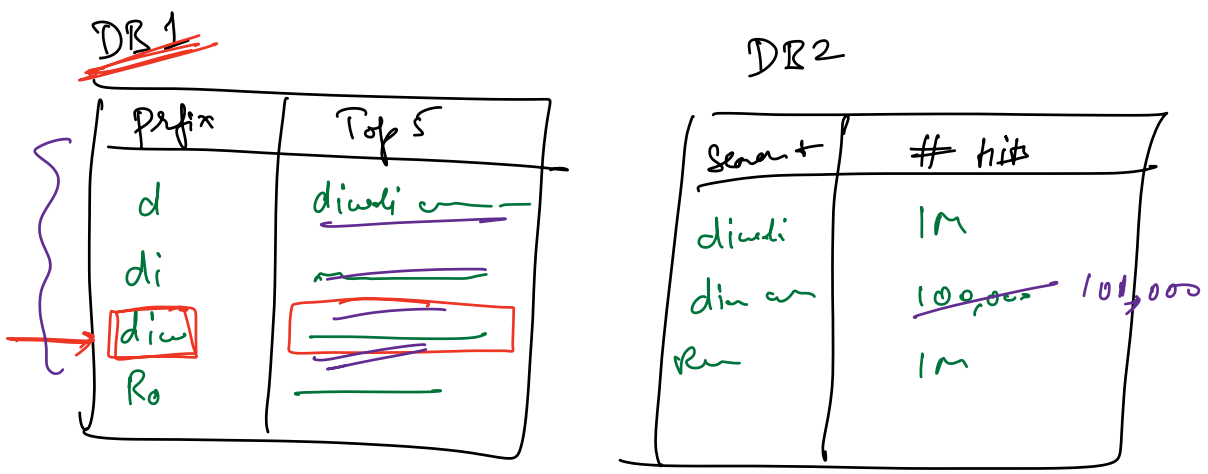
Sampling

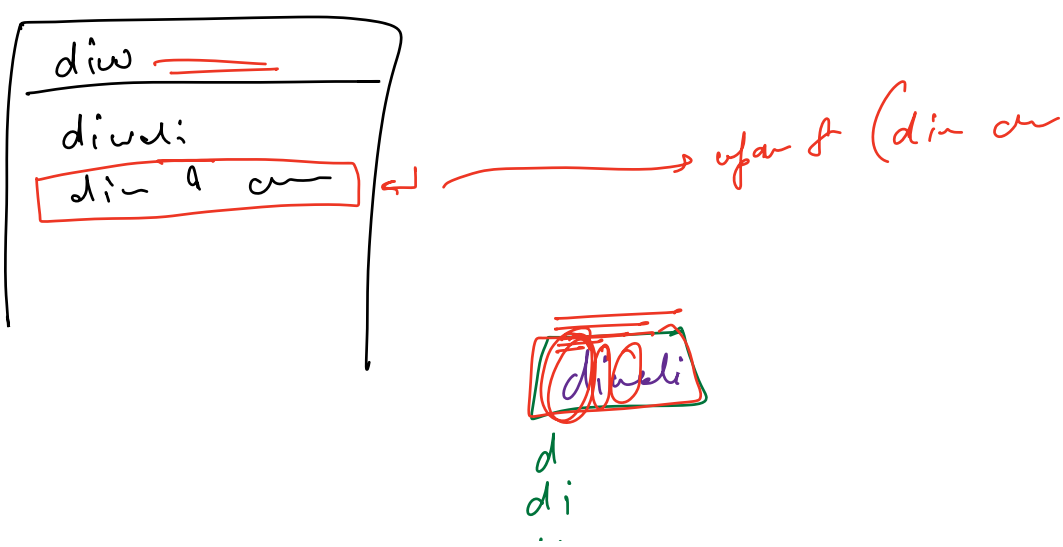
```
SAMPLE-SIZE = 1000

count = 0

updateFreq(search_term) {
    if (count == SAMPLE-SIZE) {
        sendUpdateReq(
            search_term)
        count = 0
    }
    count ++
}
```

Batching

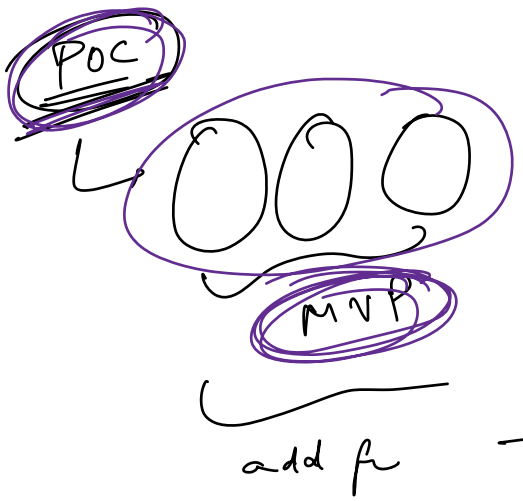
```
BATCH_SIZE

update(s_t) {
    freq[s_t] ++
    if (freq[s_t] == 1000) {
        sendUpdate(
            s_t,
            1000)
        freq[s_t] = 0
    }
}
```

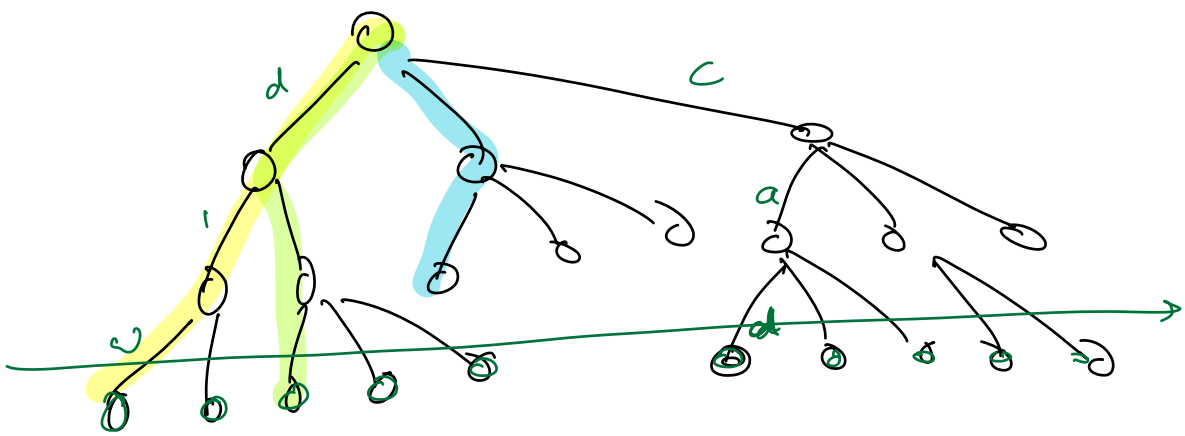

divdi
Top 5
d
di
div
diva
div
dix
dil
di2
dia


DR1
Prefix
Top 5
d
di
div
Ro
DR2
Search
hits
divdi
1M
div
100,000
101,000
Ro
1M

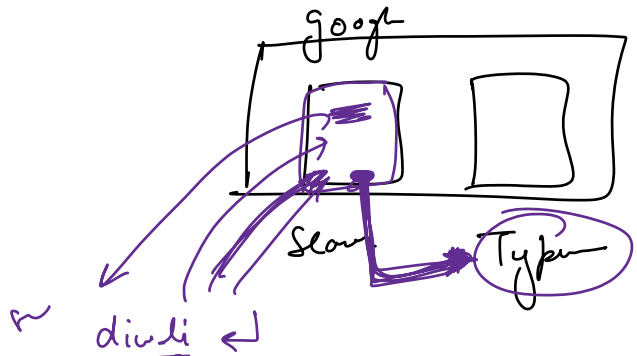

div
divdi
div a
ufar & (div
d
di

diw
diwa
diwali





Shard try



Diwa

Diwali → A
2023