# Agenda 

1. Invalidation & Eviction *Recap*
2. Local Caching - Case study 1
3. Global Caching - Case study 2
4. Case study 3 - Facebook News Feed

support@scaler.com

# Challenges with Caching

Data can become stale
∵ DB is source of truth
↓
Invalidation

Small size — cache can get full
↓
Eviction

## Cache Invalidation

Immediate Consistency
Eventual
No Consistency (data loss)

### TTL

→ Eventual Consistency
→ fast Reads (10K/s)
→ fast Writes (100/s)

write → cache miss / TTL expired

### Write Through

→ Immediate Consistency → write to DB & cache in atomic manner
→ fast Read (10K/s)
→ slow write (10/s) → 2 Phase Commit (roll back / sync)

### Write Back

→ No Consistency → Data Loss!!
→ fast Reads (10K/s)
→ ultra fast Writes (10K/s) → ∵ only write to in-memory cache

### Write Around

→ Eventual Consistency
→ fast Reads (10K/s)
→ fast Writes (100/s)

→ write via cron job


# Cache Eviction

CDNs are only for client facing data

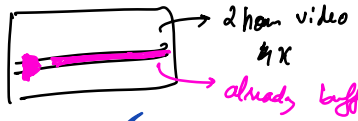
First in First Out → 1.9%

Least Recently Used → 89%

Last in First Out → 0.1%

Least Frequently Used → 9%

CDN → large companies like Akamai
Org might have their own private CDNs.




10 Gbps Ethernet. Costly

netflix → global cache for all users

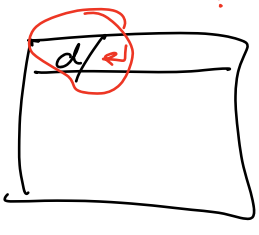
→ per user cache

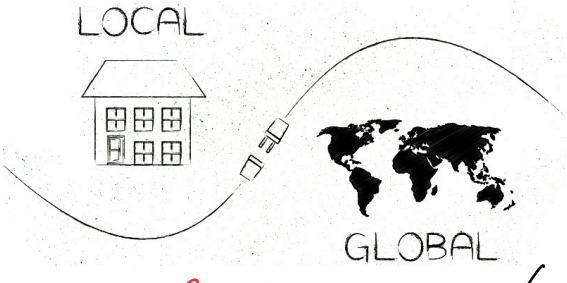
Predict what the user might watch

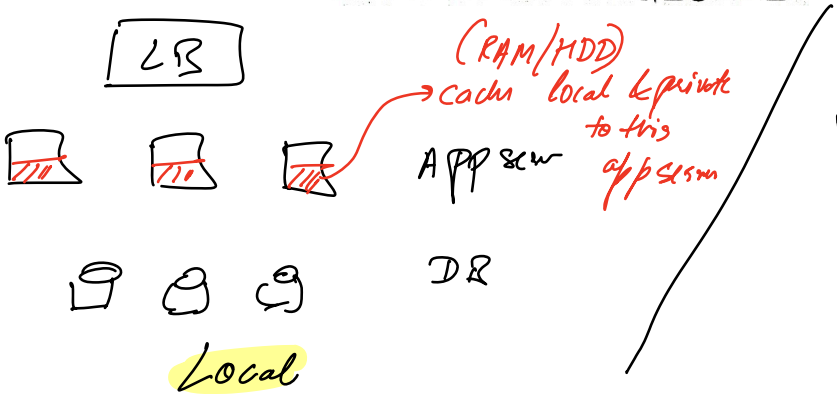
GoT → Ep 2 watched

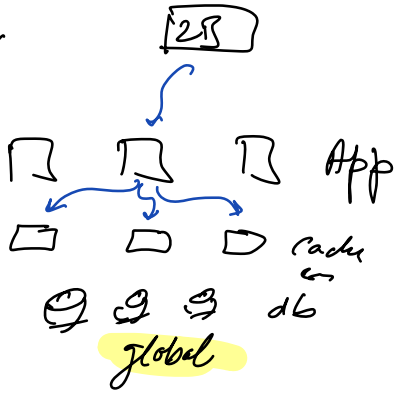
likely to watch Ep 3 next

If I watched Ep 3 last, am I likely to watch it again in near future? X NO!



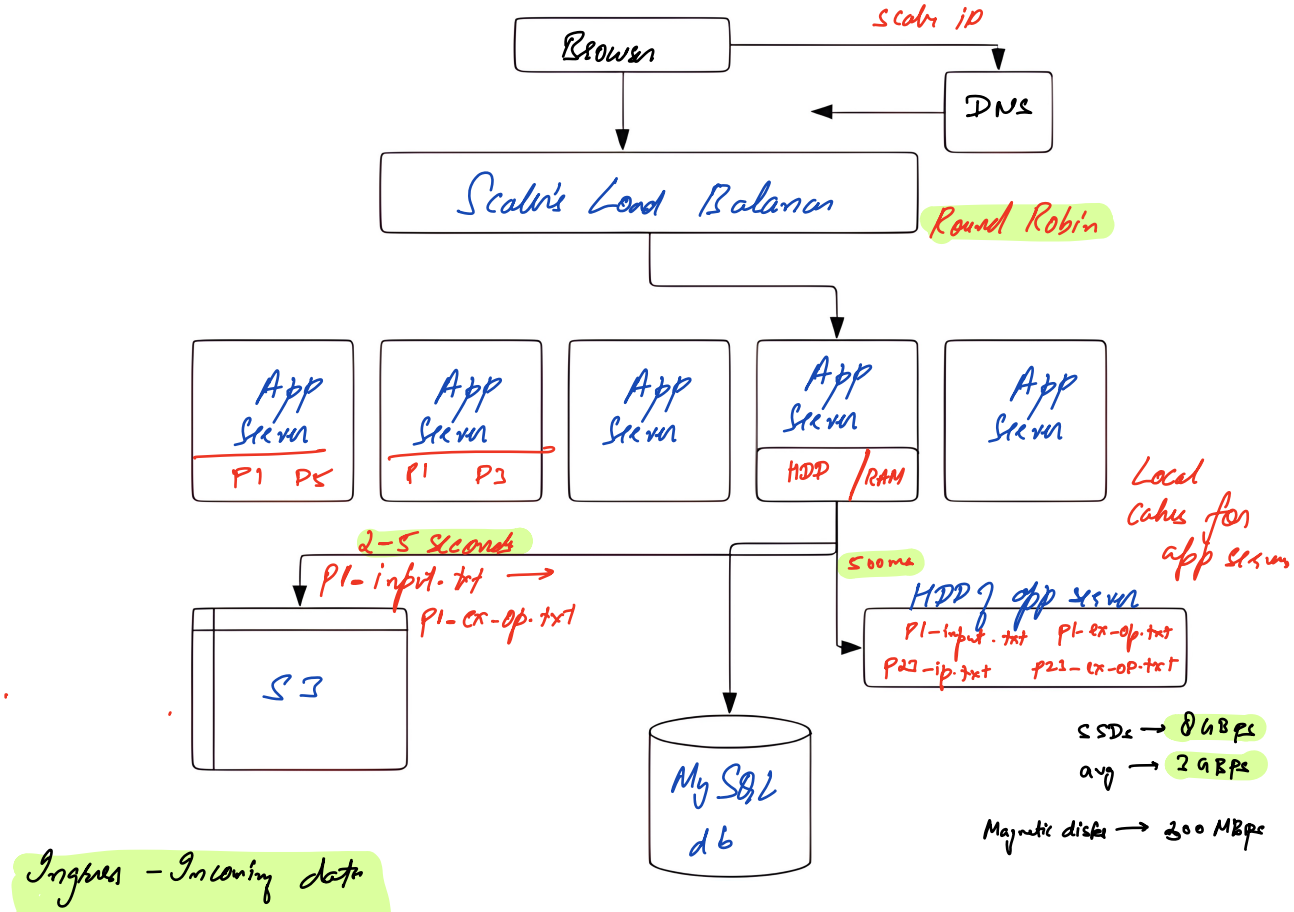




# Scaler Online Code Judge

~300,000 participants

2 hour

5 problems




Ingress - Incoming data

Egress - Outgoing data → Chargable

https://gist.github.com/jboner/2841832

? What does the AppServer need to evaluate a solution?

User-id

Problem-id

meta data about user-problem

Problem-1-input.txt

Problem-1-Expected-output.txt

Scan for each testcase / ML / TL

? How large can this data be?

Test Cases can be super large

Input array of $10^6$ items.

500 MB per file

To evaluate 1 submission we need to load 1 GB of data

# Current Caching Architecture

Source : S3

Each app server has a local cache on HDD to store testcase data

Why Not RAM?? ∵ we have ~150 GB of data → too large for RAM

Why cache? why not directly load from S3?

∵ S3 load takes 2 sec (N/W) HDD load takes 0.5 sec

∵ S3 load costs money (egress) free

Don't Appservers have duplicate cache?

Yes & No.

Each Appserver has only 1 ip & our file for each prob

Across appservers for our duplicates

AppServer 1

p1
p2 } cached
p3

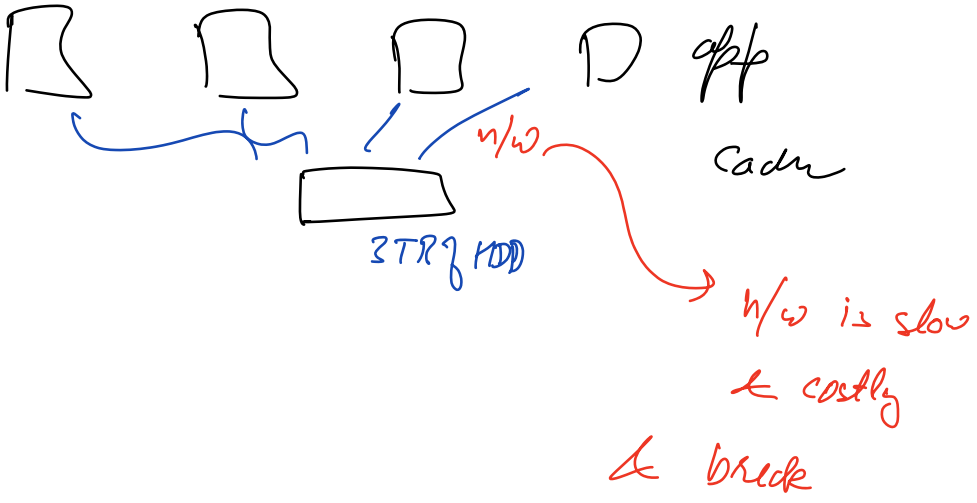
AppServer 2

p1
p2 } cached
p3

Can we cache all problem testcases?

No. ∵ I cache has 2000+ problems
≈ 3TB of data

If there is duplicate data in cache, why not a global cache?




On any day (even active contests)
≈ 100 – 150 problems are being solved

# Change to the testcases

Requirement : → able to change any problem's testcases anytime
→ low latency for updates (2 mins)

Question → Cache Invalidation

Update testcase in S3 → Cache testcase is stale

Work Around

TTL → Add cache for each app server → can store the last fetched time for testcase

X

p1_input.txt
p1_out.txt
p3_input.txt
p3_op.txt
p_10_ip.txt
} Disk

p1 : 8pm
p3 : 5pm
p5 : 3am
p10 : 9:57pm
} HashMap → in Memory

TTL ⇒ 1 hour

↓

optimal value??

High TTL → 2 hours

too stale data → context is over.

Low TTL → 1 min

every cached file is out of data very frequently

too many cache misses!!

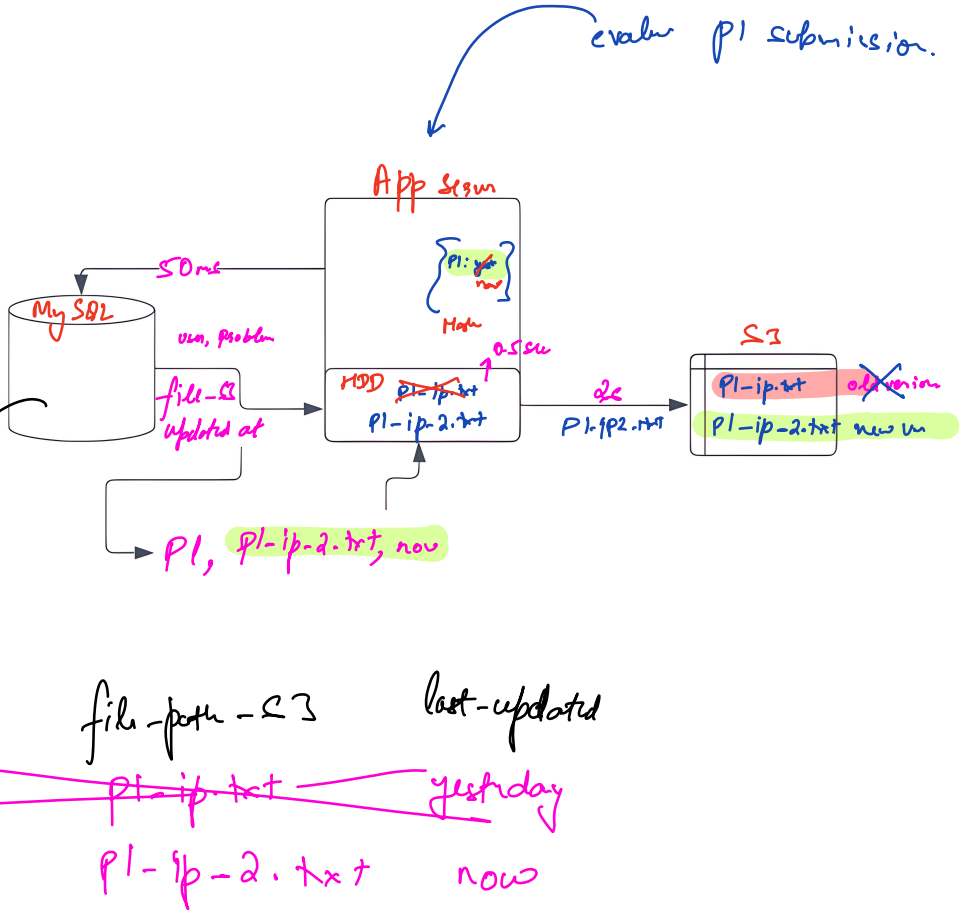
X Bad

No Good TTL

## Solution

Write-through Cache

1000 app srvrs due context




| problem | file-path-S3 | last-updated |
|---|---|---|
| ~~p1~~ | ~~p1-ip.txt~~ | ~~yesterday~~ |
| p1 | p1-ip-2.txt | now |

Yes! App srvrs must hit MySQL DB for every req. → non-issue

we need to hit DB to fetch user & problem info anyway.

do stuff in 1 db call!


# Contest Leaderboard

300,000 participants
5 problems
3 hour duration

Estimate # submissions
avg per user → 5
total = 1.5M
qps ⇒ $\frac{1.5M}{3*3600} = 140$

Each submission takes 10sec to eval
→ # users waiting = 1400

$\frac{\#\text{users}}{\text{time for one}} = \#\text{users seen}$

$\frac{1400}{10} = 140$

? How can we compute the ranklist?

User's submission → every time a submission happens, the true ranking have changed
+ve ✓
-ve → penalty

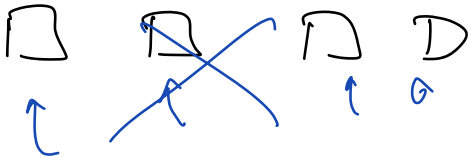
? How frequently does the ranklist change?

140 changes/sec

? Is immediate consistency important?

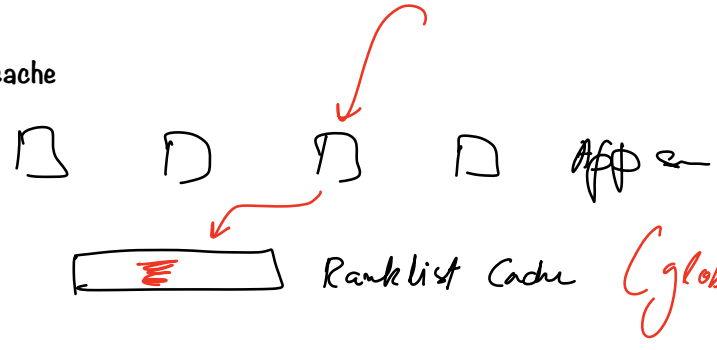
Eventual is fine
↳ max delay that will be okay (30 min @ scale)
5 – 30 mins

Local cache



TTL / work Around

Global cache



# Redis

try.redis.io
redis.io/docs/data-types
university.redis.com

100K req/sec

Key - Value

① V. powerful → adv. data structures → sets / arrays / sorted sets / bloom filter

② ultra fast → Coded in C
→ Single threaded async i/p

multi-threading
↳ locking.

③ in-memory

My Score & Rank

| Key | Value |
|---|---|
| user_id = 1 | {rank; score; P1→ – P2→ – P3} |
| user_id = 1 | {rank; score; P1→ – P2→ – P3} |
| rank = 1 | {user id = ---} |
| rank = 2 | {user id = --} |
| rank = 3 | --- |
| ⋮ | |
| rank = 300,000 | |

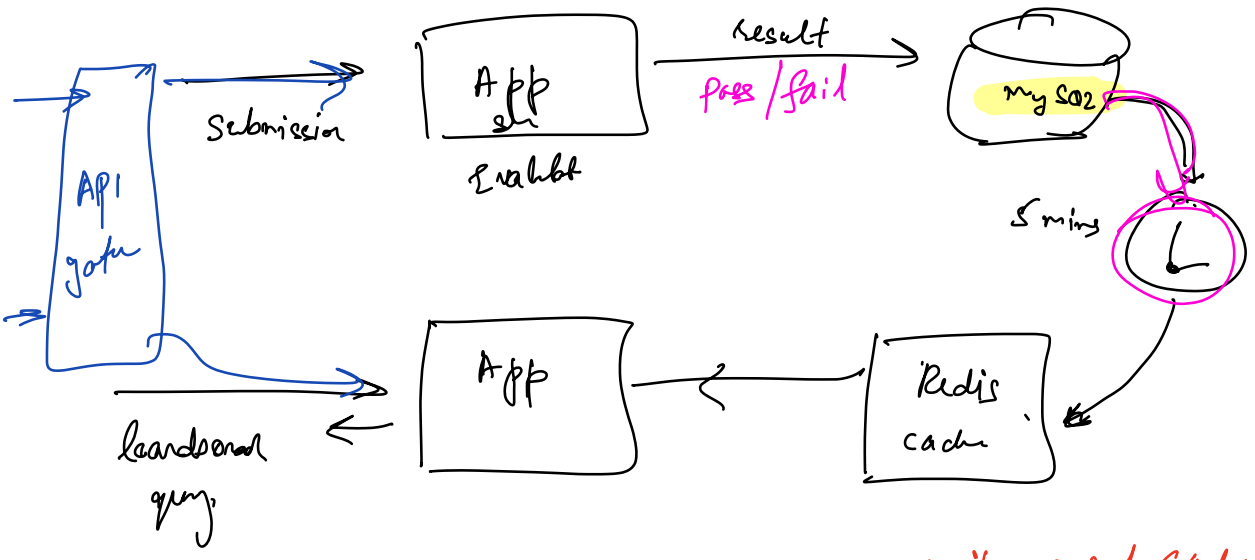
10 bytes
Score
3 bytes
100 bytes


20 entries per page

query ( all keys b/w rank-000001 to rank-000020 )
↳ Pg 1

Pg 2 → rank-000021 to rank-000040

600,000 * 110 B
= 66 MB data




write-around cache

300,000 users,
1 hour contest

On avg → each user might look at rankist ⇒ 5 times.

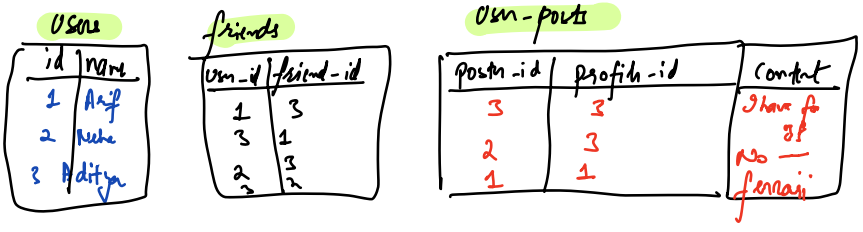
$\frac{300{,}000 * 5}{3 \text{ hour}} = 140$ qps

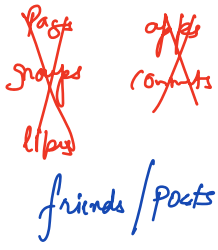
10:48 → 10:55


# Facebook News Feed

? How can we compute the News Feed?

DB schema (Relational)

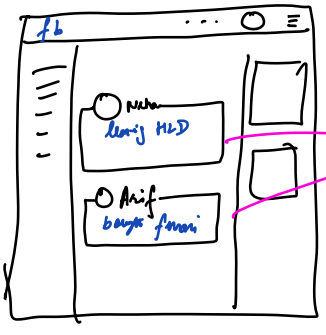



## Profile Page


any posts made by
Aditya / his friends
↳ on Aditya's profile

## News Feed




recent & popular posts made by friends of Aditya
&
on their own profiles or other people's profiles

Select * from user-posts
where profile_id = 3 → Aditya
Order by created-at desc
limit 20

① get friends list of Aditya
② join on user-post table to find any post on any friend's profile
③ score → recency
↳ popularity
↳ demography
↳ sort by user-affinity / person

? Does all the user data fit on a single machine? ← posts / friends lists

FB → 2B+ users

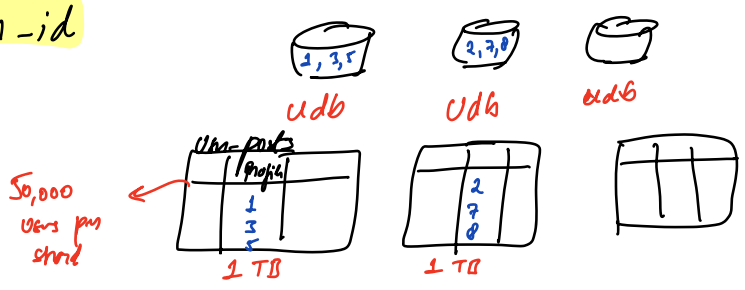
No!!

∴ Shard the data

? What is my sharding key? ← read / write

↳ ① all the frequent queries must ideally hit 1 shard ✓
2 shards ✓
n shards ✗ bad

User_id




Profile → exactly 1 shard

∴ all posts made on a given user's profile
↳ inside that user's shard.

? How do I retrive data for the news feed?

We have to hit $\frac{all}{n}$ shards!!

worst case → if Aditya has n friends we might have to hit n shards

fan-out limited by slowest shard → 

3000 req DB

1 ≤ n } fanout

? How can you optimize the construction of the News Feed?

Potential Solutions

→ diff sharding key

→ pre-calculate news feed for each user & cache it

↳ ① estimate amount of data

↳ ② how will you update the newsfeed algorithm.

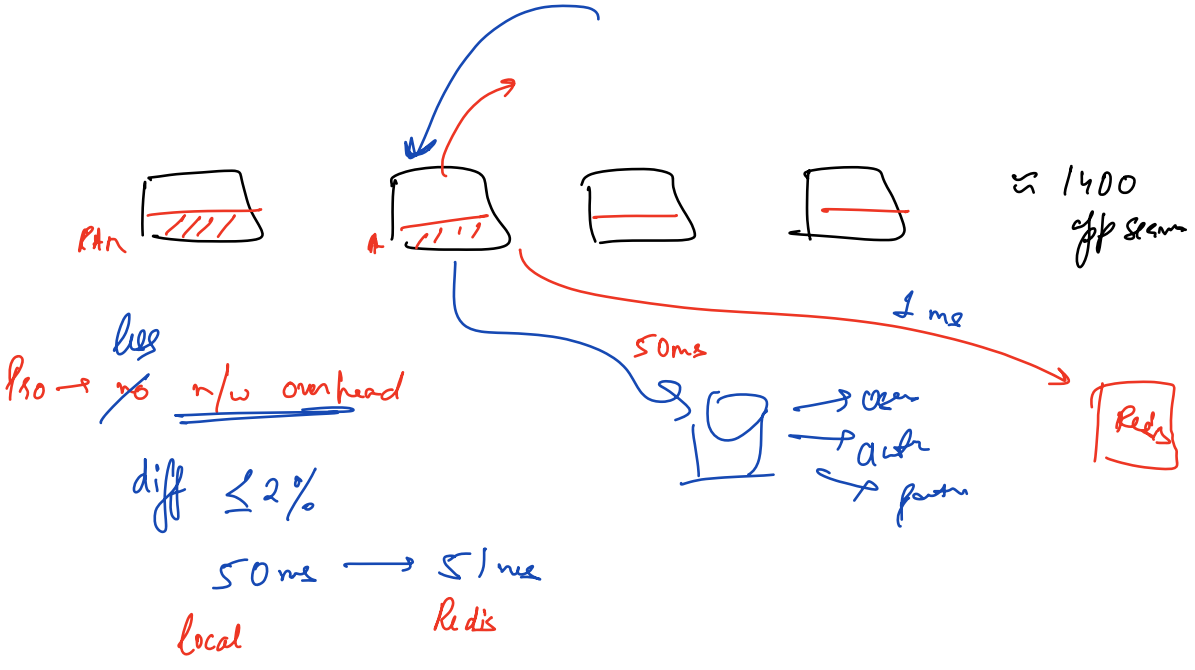
→ on each post → push to each friend's news feed

↳ fan-out writes

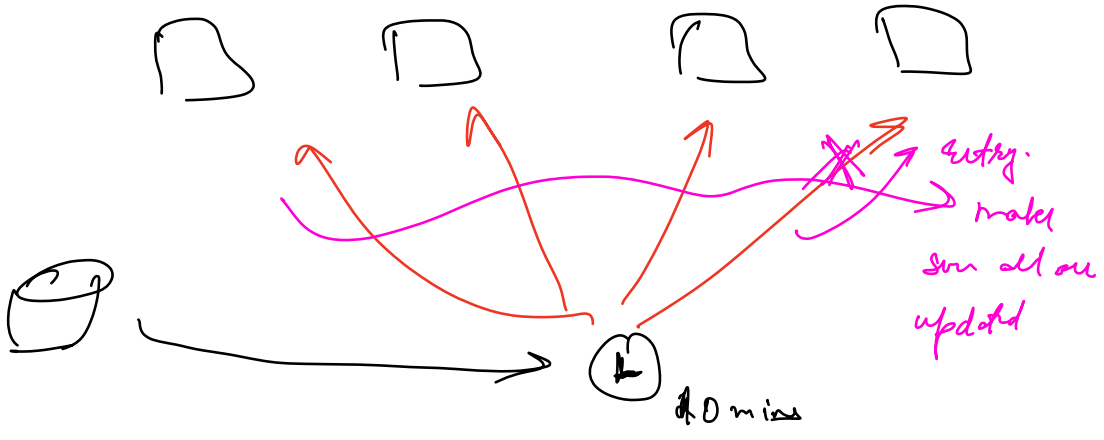
→ unless most all friends are in same shard → impossible

↳ clustering → hard problem


!
?
QnA


Pro
less
n/w overhead
diff ≤ 2%
50ms → 5ms
local
Redis
50ms
1 ms
Redis
≈ 1400
app servers

Cons


entry.
make
sure all are
updated
10 mins

Calculation

assume 1 server can handle 1 req at any given time

N servers

each req takes 10 seconds

10 secs ⟶ N requests get served.

1 sec ⟶ $\frac{N}{10}$ req

Unitary Method

10th class


⟶ 140 req

$\frac{140*10}{N} = 1 \Rightarrow N = 1400$ servers.

Evaluating a submission

I/O Bound

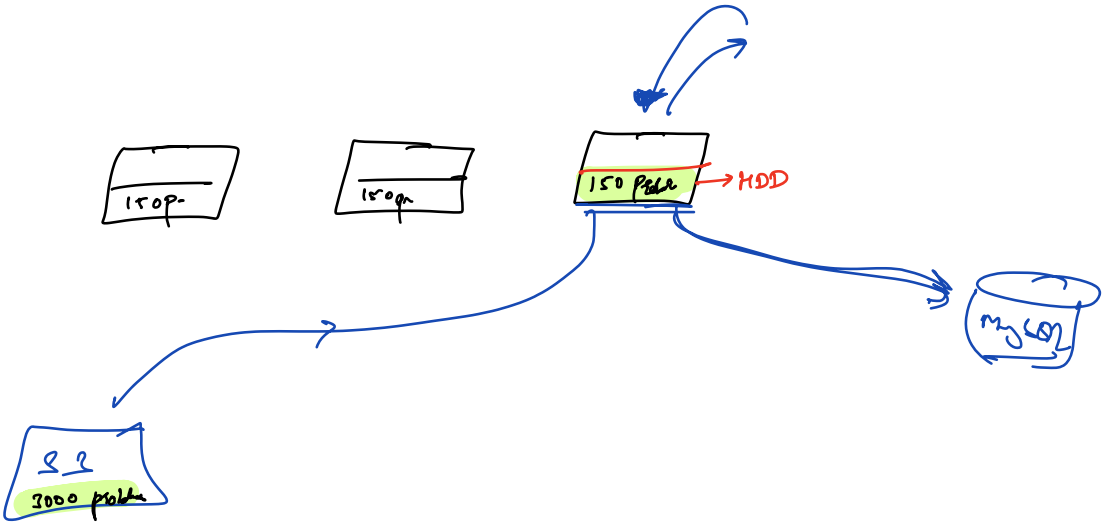
2 sec / 0.5 sec

CPU bound

10 sec ⟶ pure CPU time

you should Never multi-thread a CPU bound process ( OS lectures Threading )

leaderboard server ⟶ I/O bound




data → written once

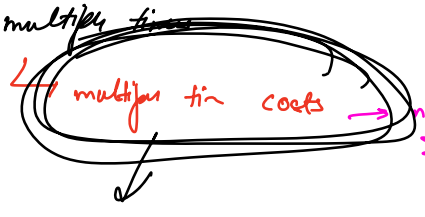
→ read multiple times

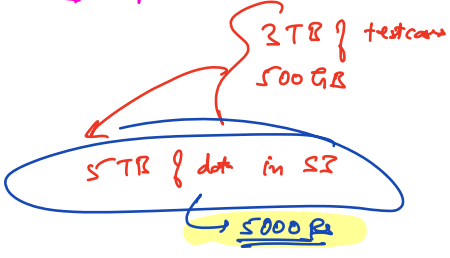
cost of n/w → 1 time

cost of storage → super cheap

5 TB for 5000 Rs

Tape drive 5TB → 2500 Rs




710 rs

---

Modern Hardware

Typical Server

200 GB ↔ 2TB HDD

2GB ↔ 64 GB RAM

2 ↔ 16 cores

100 Mbps ↔ 10 Gbps n/w

Most Expensive Server

5 PB of HDD

12TB of RAM

600 cores

100 Gbps n/w

App Server → 10GB of in-memory cache

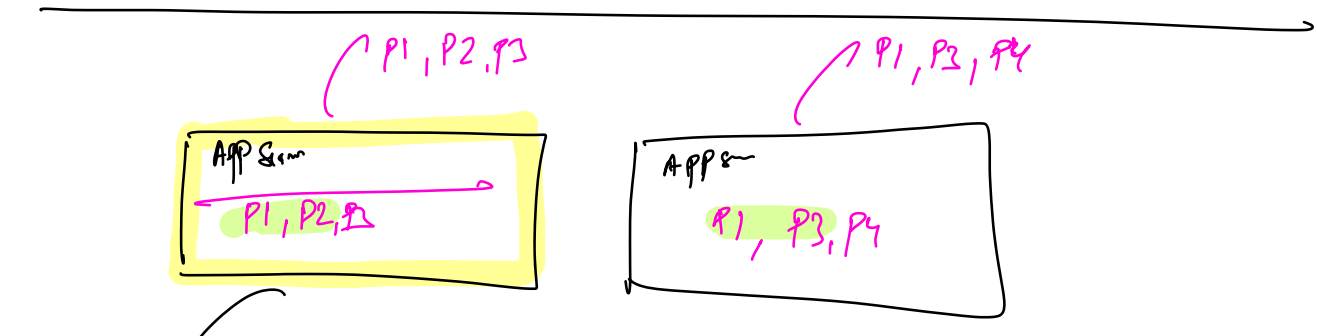
Redis Server → 60 GB → cache

4 GB → OS / - -

RAM / CPU / HDD

↓ 95% ↓ 95% ↓ 80%

5%




1. Webserver → Spring Boot / Django / Node.js
2. HDD → OS / code / files
3. RAM

POST

scale.com/api/evaluate

{ user_id : Pankaj
problem_id : —
code : " print('Hello, world') }

handle_req (req) {

Auth user

fetch user/prob data from MySQL

↳ also fetch testcase file path & upload it

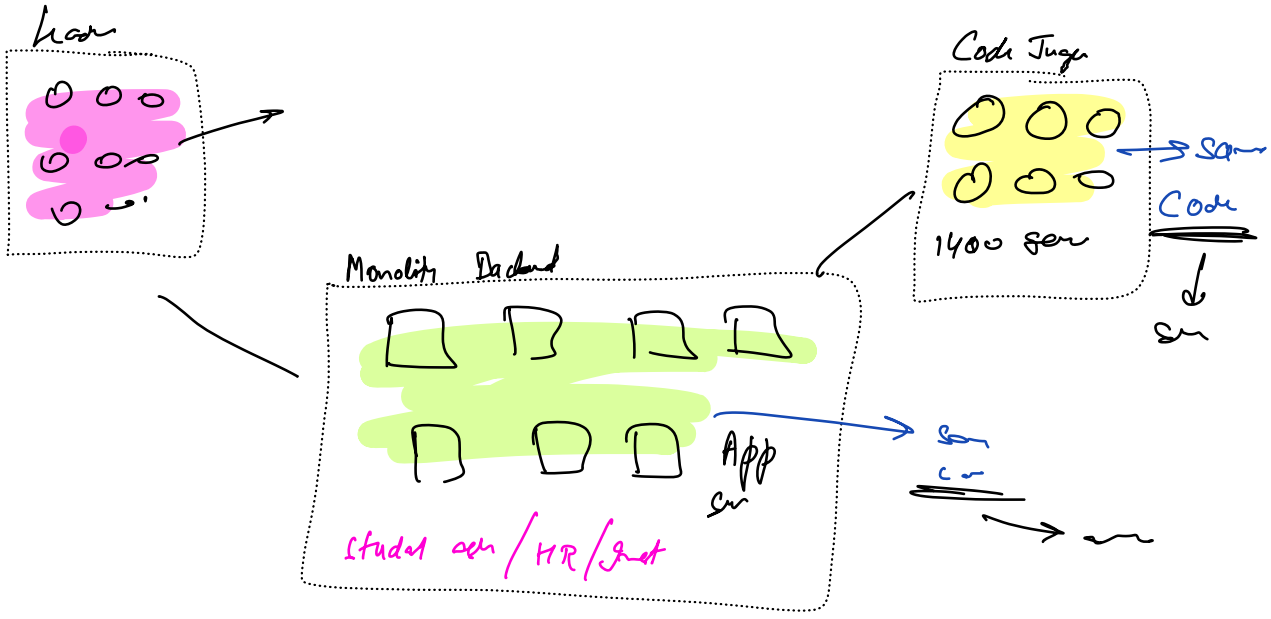
check Memcached → do we have an entry for this prob?

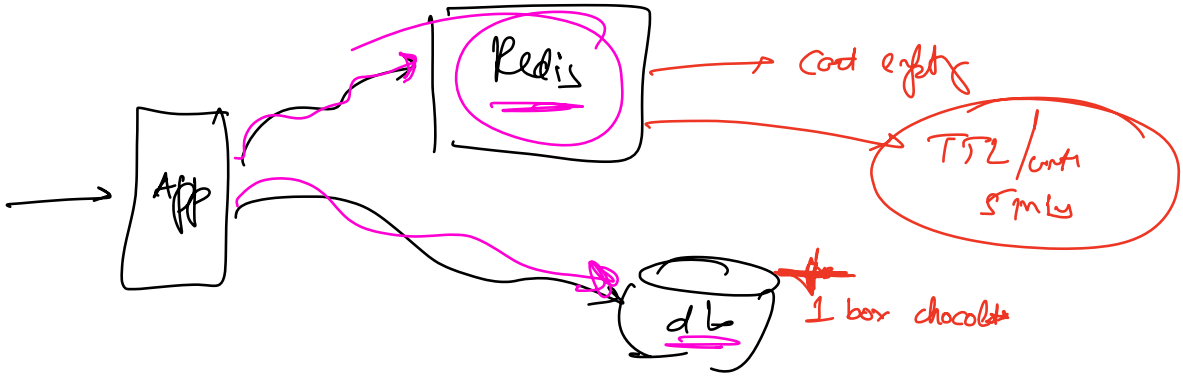
if (yes) → is it static/immo

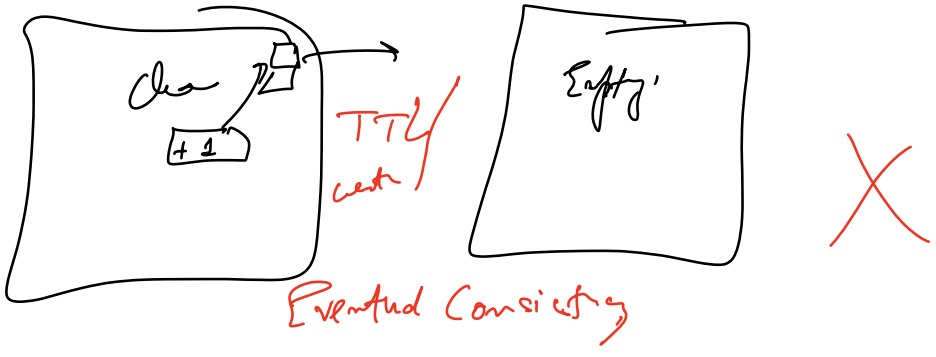
↳ fetch from S3

↳ store on HDD

↳ add Memcached




Immediate Consis → write through

Cache `user -> news feed`

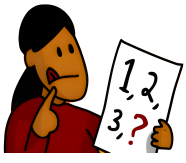
3

Next class

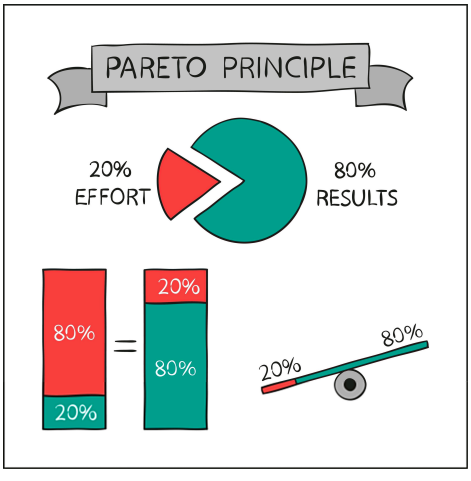
1.

2.

3.

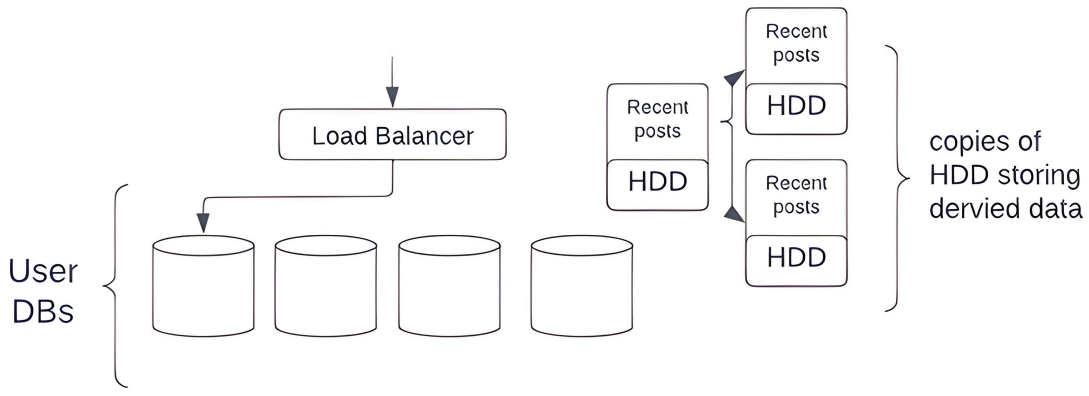



## Back of the envelope estimates




? Daily post data

? Monthly post data


Load Balancer
User DBs
Recent posts
HDD
Recent posts
HDD
Recent posts
HDD
copies of HDD storing dervied data