Agenda

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

support@scaler.com

9 - 11:30 class
+10 mins
doubt

Most applications are read heavy → caching.

Challenges with Caching

- Cache is not the source of truth
  - ↳ database
- Cache data can get stale

- Small in size
- Cache miss → data doesn't exist there in Cache
  - ↳ lead to slowdown
- to load new data Cache needs to remove existing data.
  - ↳ what data to remove?

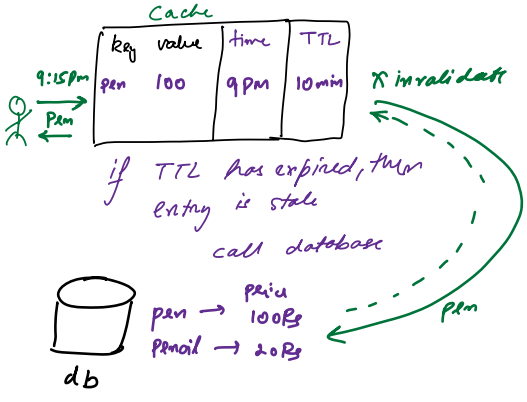
Cache Invalidation

↳ fix stale data

Time to Live (TTL)

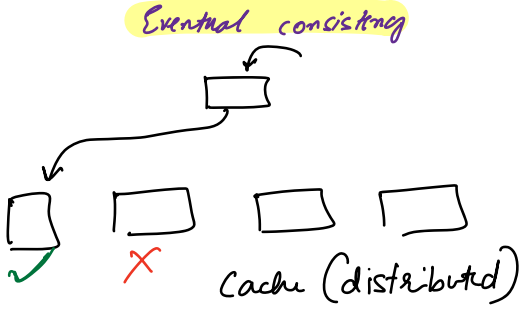
Simple / Configure TTL

you will get slightly stale data



Write Through

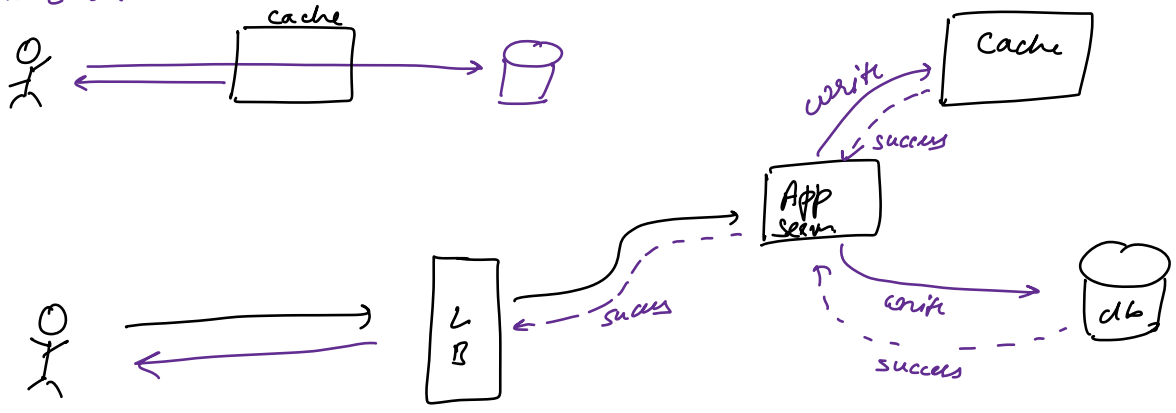
Write Back

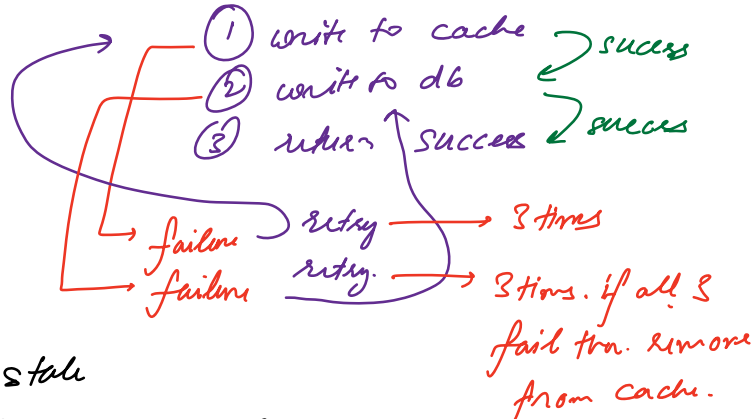


Write Around

Write Through Cache

Any updates must first be written to cache.





? Can cache data be stale in case of Write Through Cache?

Never!!

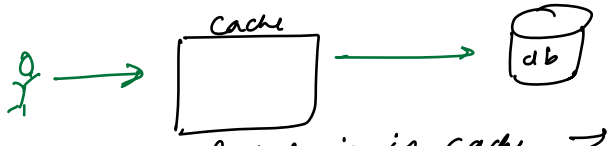
1. Always fresh data
2. Reads are super fast → every cache hit will return directly from cache
3. writes are much slower

Write through cache is good for a read-heavy system.

most req. are reads & not writes
90% of applications

Reads are fast when using a cache

↳ ∵ of how cache works.



if data is in cache → 1 ms → return from cache.

if data is not in cache → read from database ≈ 10ms

hit → 1 ms → 99% cache hit

miss → 1ms + 10ms = 11 ms → 1% miss

cache db

avg. time per query.

$0.99 * (1\text{ms}) + 0.01 * (11\text{ms})$

$= 0.99\text{ ms} + 0.11\text{ms}$

$= 1.1\text{ ms}$

Write around cache

→ write happens directly at the database

↳ the cache might be out of sync for a short while

→ cron job (10 mins) that queries recently updated data from db & updates in cache.

```
select * from product_price
where updated_at > (now() - 10 mins)
```

→ update in cache.

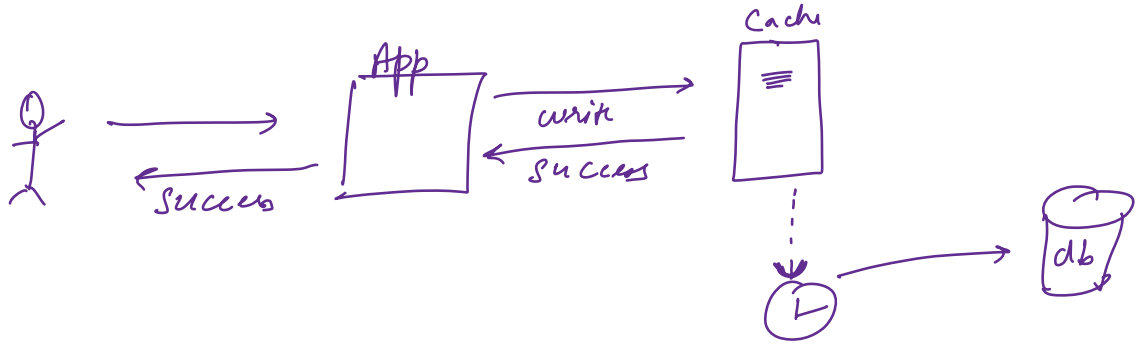
write-back cache (inconsistent)

↳ first write data in cache

↳ return success

even though data is not on db!!

↳ cron job that will periodically flush this data onto the database.



Cons

if cache server crashes we lose some data!!

Pros

Extremely high throughput

Use write-back cache in write-heavy system only if consistency is not important

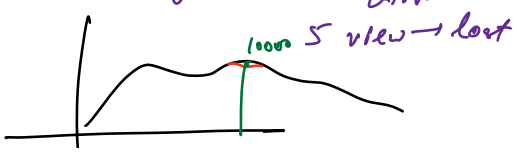
Analytics / tracking clicks on ads

↳ 10,000 views

↳ only 9995 views got written

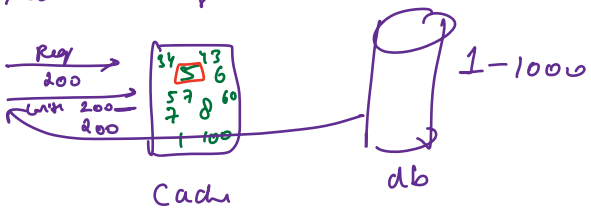
↓

friend based.



Cache Eviction

Cache is limited in size



First in First Out

queue
simple

Least Recently Used

time → Hashmap + Heap
→ Hashmap + DLL

If you see something, it is v. likely that you will see it again in near future.

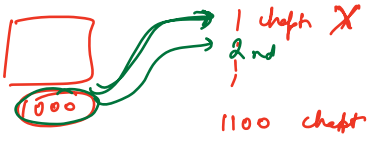
Last in First Out

→ whatever you've read recently will get removed.
H/W → find out circumstances when this is useful.

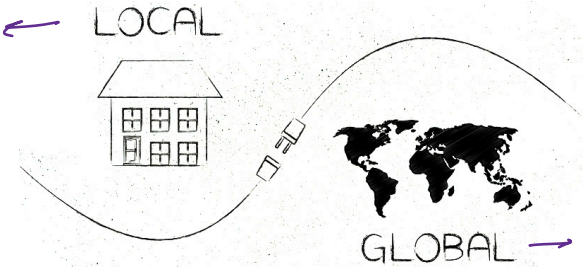
Least Frequently Used

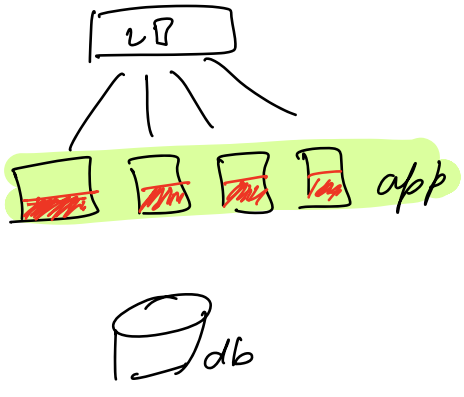
→ evicted.
↓
read counts

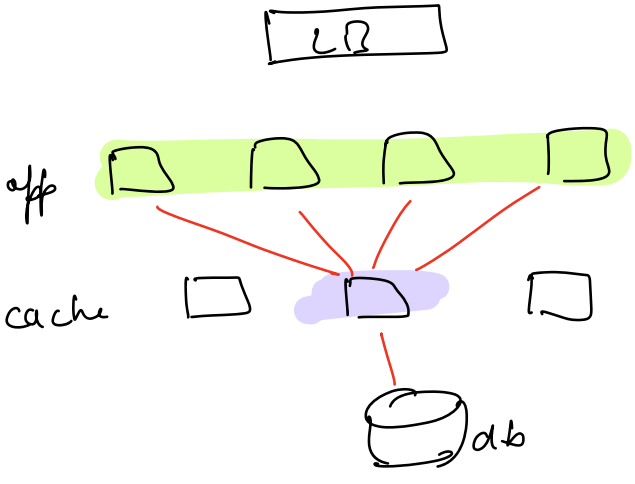
One Piece Manga Reading groups.



used by only 1 app server. ← LOCAL

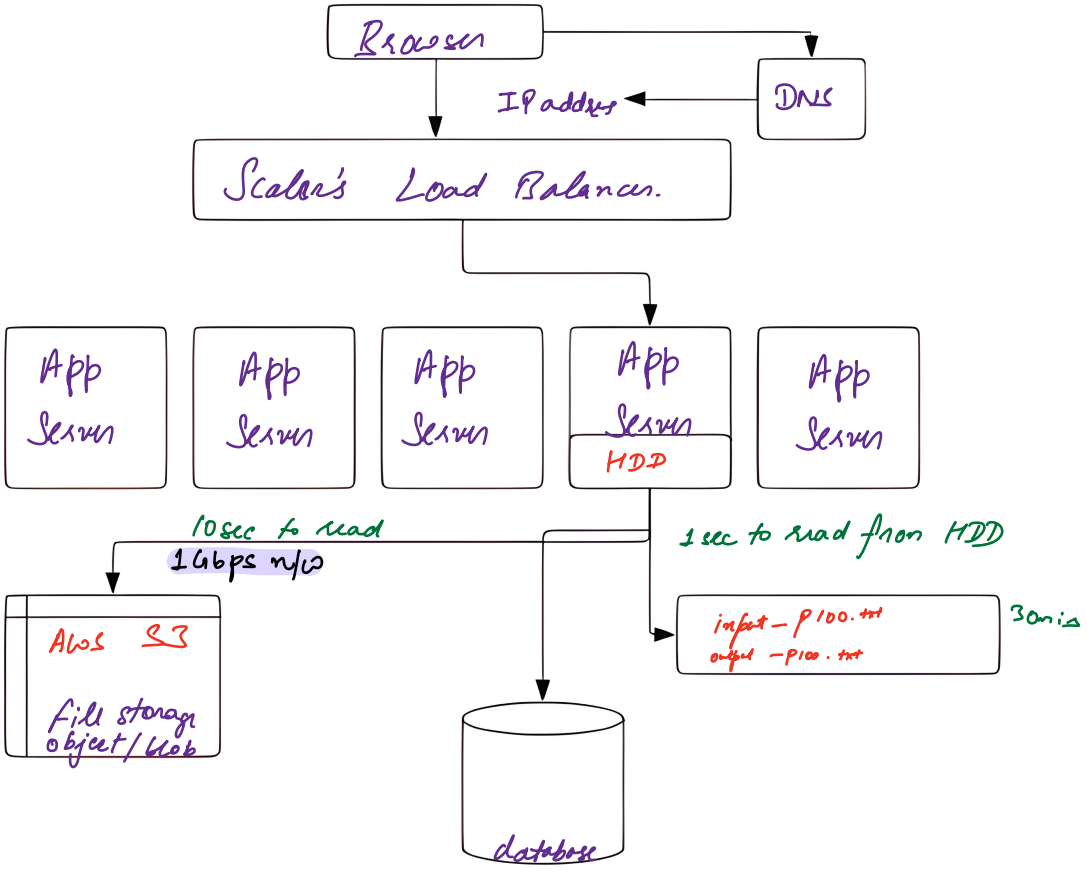






10:21 → 10:30

Online Code Judge



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

? What does the AppServer need to evaluate a solution?

problem-id problem-metadata
user-id solved-already score
code language. input testcase & output testcase

given array → sort it
O(nlgn)
≈ 10^6 size
40-50 testcases

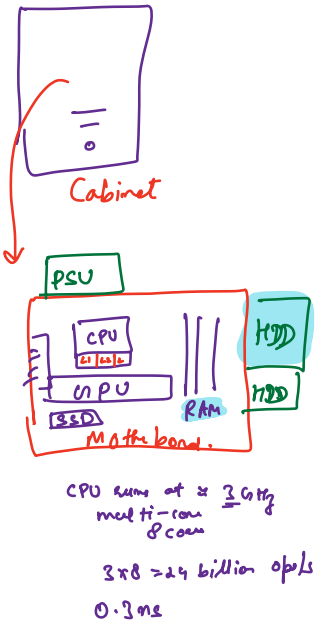
500MB input.
500MB output

? How large can this data be?

input, output } 500 MB each → 1 GB of testdata for a single problem.!

Change to the testcases

if a test case is changed → how do we update it in the local cache?



CPU runs at ≈ 3 GHz
multi-core
8 core

3 × 8 = 24 billion ops/s

0.3 ns

① at every write, push change to all appservers.

↳ write through cache

→ every app server has its own cache.

→ update every app server

→ wasteful

↳ write back cache ↳ TTL

some period during which data will be stale.

② TTL / write back cache.

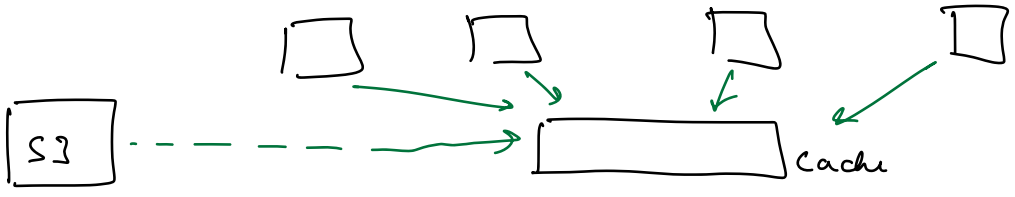
bad solution → we want immediate effect

30 min TTL → entire content gets over & test case are not updated

30 sec TTL → most req will have to go to S3

no benefit of cache

③ Global Cache

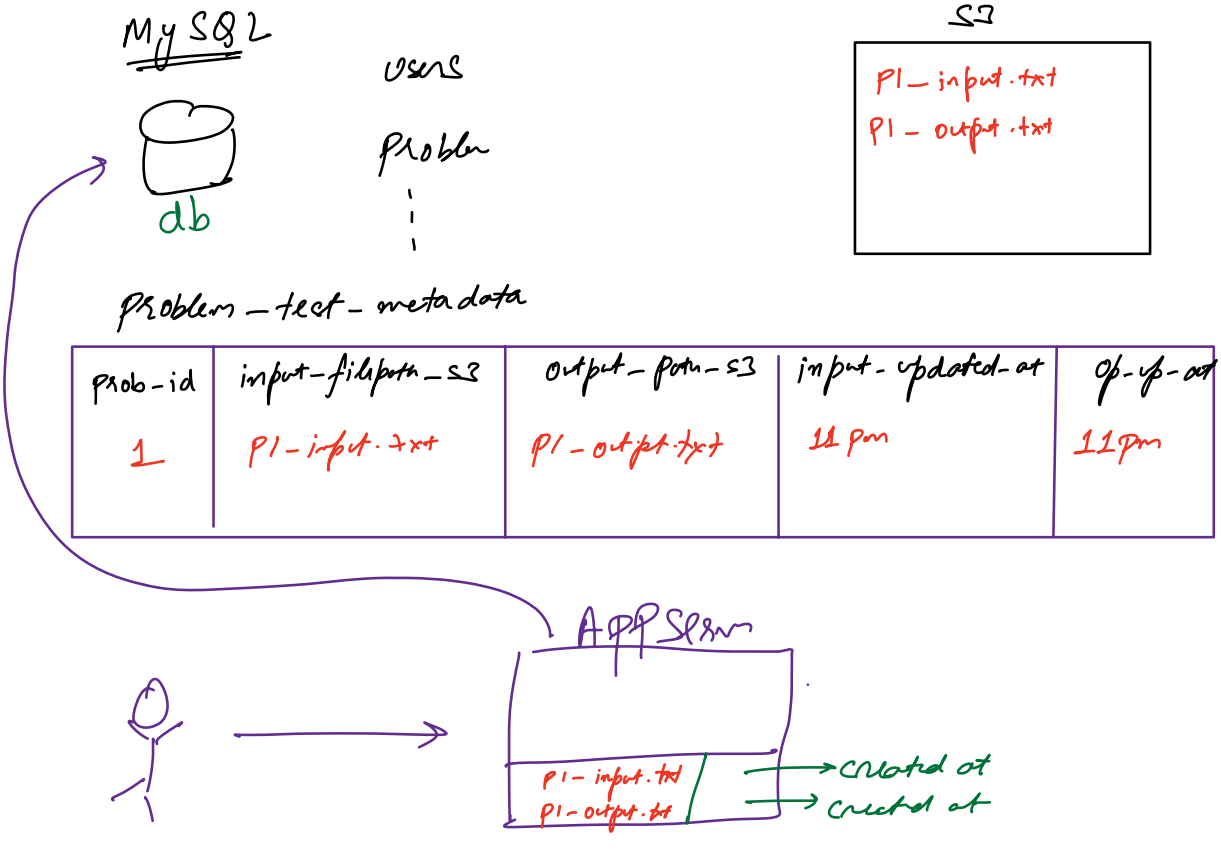


① alot of n/w traffic will come to cache

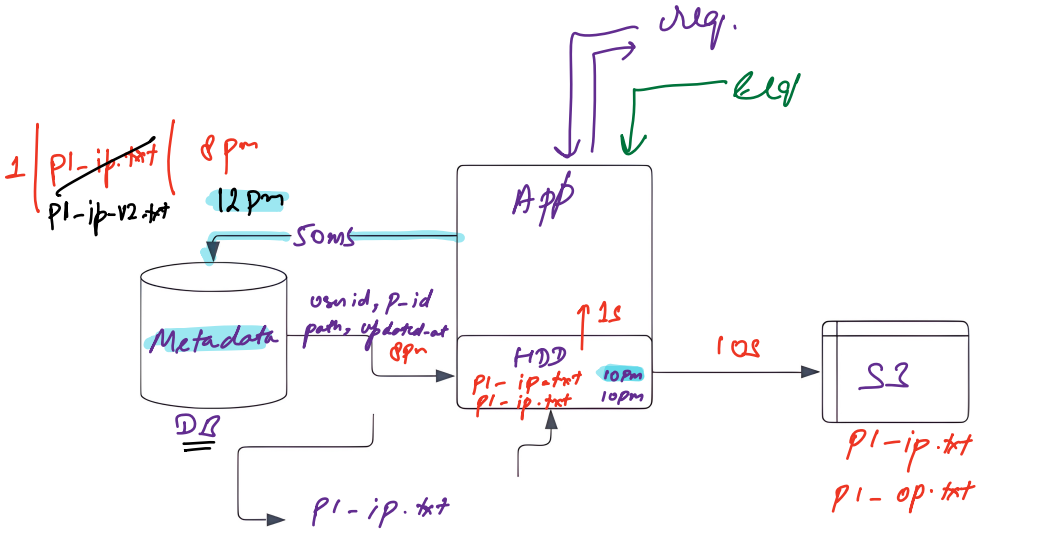
② if cache serv has 32 GB RAM
↳ 32 files
a lot of cache misses.

③ store in HDD → 3 GBps SSD | ~300 MBps HDD

File Metadata in DB



Solution



p1-ip-v2.txt
p1-op-v2.txt

Test case → change

1. update file in S3
2. update db metadata

Contest Leaderboard

How can we compute the ranklist?

How frequently does the ranklist change?

Is immediate consistency important?

Local cache

Global cache

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

Facebook News Feed

? How can we compute the News Feed?

Profile Page

News Feed

❓ Does all the user data fit on a single machine?

❓ What is my sharding key?

❓ How do I retrive data for the news feed?

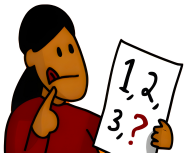
❓ How can you optimize the construction of the News Feed?

Cache `user -> news feed`

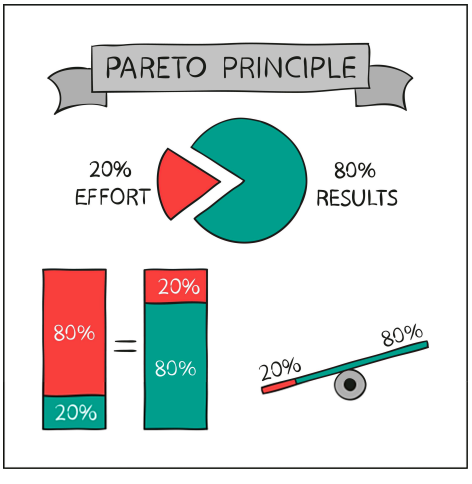
1.

2.

3.

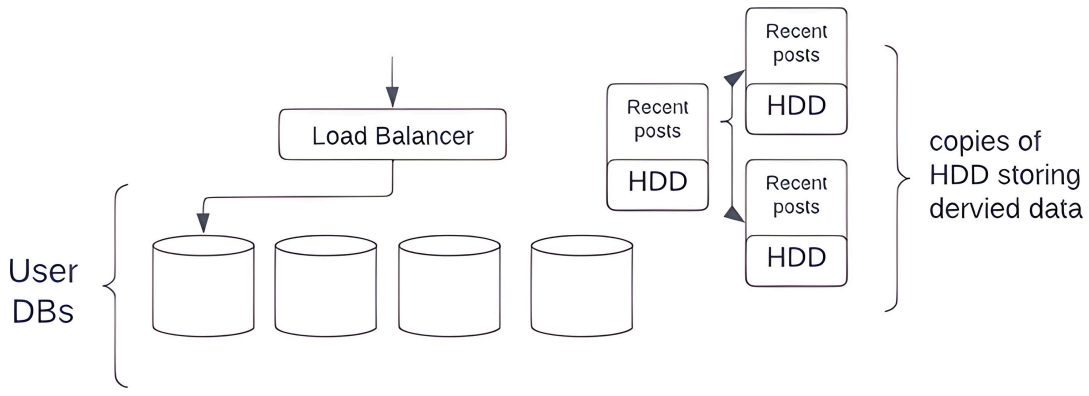


Back of the envelope estimates



? Daily post data

? Monthly post data

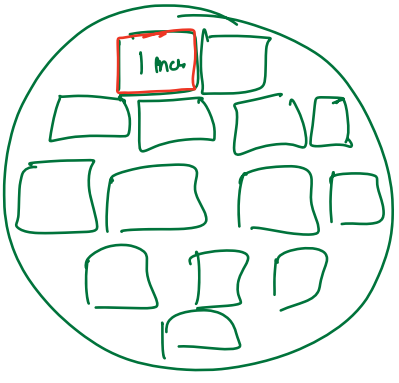

Load Balancer
User DBs
Recent posts
HDD
Recent posts
HDD
Recent posts
HDD
copies of HDD storing dervied data

CPU → Intel

clean rooms

ULSI → 3nm, 5nm, V.V.V. expensive

Ultra large scale Integration



Silicon wafer 1 million $

flip-flop → registers → 50Mb

→ cheaper but slow

L1

→ cheaper but slow

L2

⋮

---

① write through → no stale data

② write around → cron job that updates cache using db

≈ TTL → stale for some time

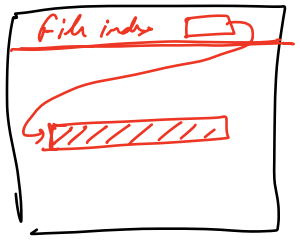
③ write back → inconsistent data

meta data → S3 → source of truth → did we have stale data

→ db metadata → No stale data!

S3 → file storage

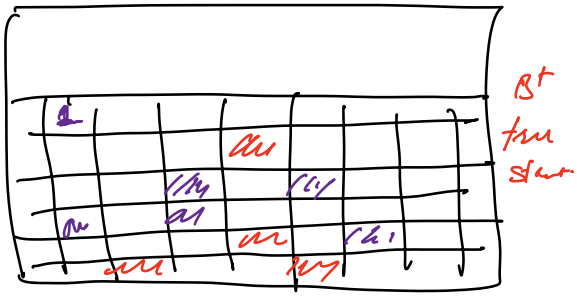
NTFS / EXT?



Continuous data access

DB

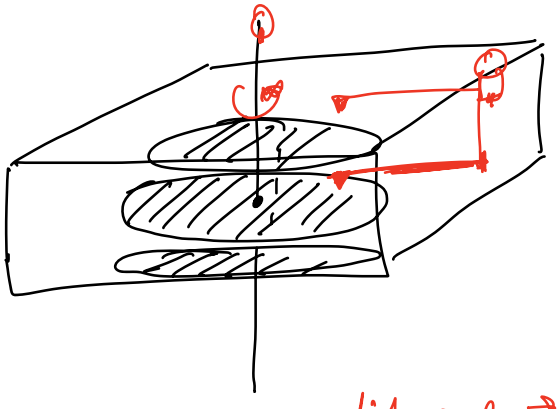
query data

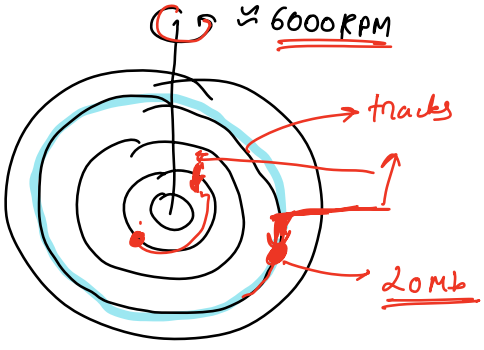


B+ tree

P_id | index 500n | opt 500n

random data access





disk seek ⇒ 100ms

$\frac{6000}{60s}$ = 1 spin = 10ms

TTL → stale data

global → high n/w usage

↳ HDD → slow

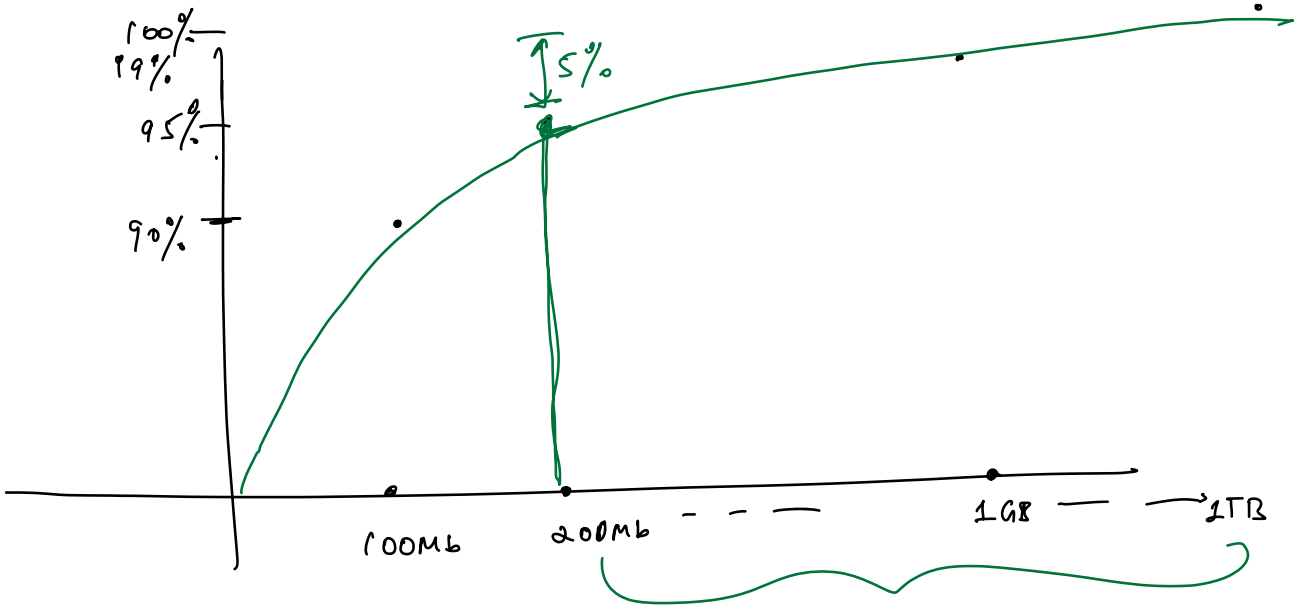
↳ RAM → small

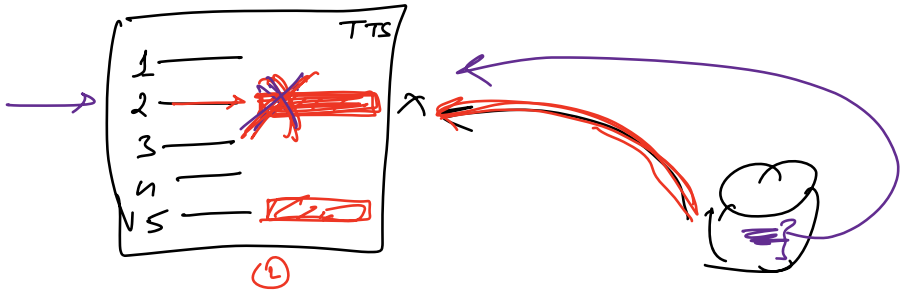
write back ✗ → inconsistent

write around → similar to TTL

DB → 1 TB

Cache → 100 Mb





(1) wait for req (2) → expired → invalidate

(2) Redis detect that (2) is stale → remove it

(3) cron → stale data → refetch fro db to update in cache