# Design Problems 1 : LRU Cache

Friday, 23 April 2021 6:27 PM

- Not a system design course.
  - → what is the problem
  - → collecting Requirement
  - → capacity planning and constraints
  - → what should be the function

→ Cache
LB
Purging DB ✗

⇒ School Registration System.

- Student
- Teacher
- Batch
- Department
- Staff

"is a" relationship

Employee → roles → Teacher → Subdomains
→ Principal
→ office staff
→ other roles

Animal → Dog

Students → ID
→ Batch ID
→ Department
→ Phones
→ Name

"has a"

Q → N number of songs ($10^8$) in our system.

Every song has an address. You want to query (song ID), and it should return you the memory location of that song.

→ HDD
SDD
RAM
Cache

Process, ~~cost~~ purpose?

⇒ 1 solution → Hash map

<ID, pointer>

Cache → K → 100

{free, ID}

free
1 → 1
2 → 3
3 → 2

Cache-size 4

5 → {5, x}

Two sets

Set

& HM

FIFO

LRU

Unordered

HM 1 → {S=1, address = xx} {S=2, x,y} {S=3, ~~address~~} {S=3, t=4} {t=0, S=1}

HM 2 → {S=1, t=2} {S=2, t=1}

HM 3 → {t, S}
{t=0, S=1}
{t=1, S=2}
{t=2, S=3}
{t=3, S=3}

Cache HIT

1 2 1 3 4 2

→ Is present In Cache ( song ID ) → O(1)
⇒ Is full ( ) → O(1) → O(1)
⇒ Update Cache ( ) → O(1) → O(log(N))
⇒ get from Cache ( ) → O(1) → O(1)

O(M)

Replacement Algo :- 1) FIFO
2) LIFO

{ → <id>
Hash map → <id, address>

map id → address
map id → time
set {time, id}

3 LRU → Least Recent Used

2 2 2 1 2 1 3 4

{1, 2, 3, 4}

(K=3)

time, id

t=0 → 1
t=1 → 2
t=2 → 1
t=4 → 5

Hashmap → <id, address>
heap → {time, id}

{f1, s1}
{f2, s2}
{f3, s3}

vec

Set < time, {id, address} >

3

{time, Id}

+ ✗

1 2 1 3

hashmap → <id, time>
1 = 2

t = 1
{0, 1}

Set → {time, ID}

{time++, id}

S=1

t=2, S=1

{t=0, S=1}
{t=1, 2}
{2, 1}

2, 1

t=2, S=1

⇒ Aim → Perform all the operation in O(1)

O(1)

head

tail

1 2 3

Hashmap < id, node* >

Present map

{x, y}