IRCTC

Thursday, 28 February 2022 15:23

① Requirements:
- search (train : src, dest, date)
- put based on time/price (fastest)
- availability per class (cheaper/AC-3)
- Payments 3rd party service
- Modify ticket : NT
- Book tickets
- Cancel tickets
- PNR status / Booking history

60 , 61 - - - -

② Scale

- 1000 - 2000 users → average QPS
- 20 trains
- 20 trains
- 8 seats/ train
- assume all trains are fully booked

③ 3000 trains × 20 × 20 × 8
= 1M × 3 × 8
= 16 M seats

20% waitlist
100
120
(101 - 120)
t1: 25 Feb

16 M / 365 days
15 years
2

920 days
120 days
older than 120 days

80M × 120 × 200
2400M

Bookings
user-id, seat-id, train-id, src, dest, date, amount,
(32) (32) (32) (32) (32) (32) (4)
average (32)
160 + 16 × 195
~ 200 bytes

Status (1)
PNR (4)

2400 × 200 bytes
480 GB
120 days

QPS
write QPS : 2M / 24 × 3600] 10pm - 6am (6hr)

- 30% booking on tatkal
- 10 mins
→ true QPS?

12 hr

30% of 20M / 10 × 60 + 70% of 20M / 12 × 3600

<3000K / 600 = 10K req/sec → 10QPS

10000 req → 1sec

= 0.1 ms

1 req → 1ms
(1sec) → 1000

multi-threading

1 core/sec
1K req/sec
threads

single-core → queueing

10 ms → 1000 req/sec
t=0
1 — 1

100 req/sec
100 threads

with QPS: 10000 → 10,000

1 thread : 1 write takes 10 ms → 1ms
: in 1sec : 100 req/sec → 10K

how many threads do I need to complete 10K req/sec
10,000 / 100 = 100 threads

CPU

QPS (N) # of instruction your CPU can handle in 1sec
→ 10^9

22×10^9 instructions per
= 1B

T1 T2 T3 T4 T5

T1 T2 T3 T4 T5
1ms 10^7

10^4 × 1sec multiple cases
problem

your memory recall

QPS
(1 ms) 1000
(5) 10,000
1000 × 1000 = 1000000
~10,000 shards

④ Design Goals
→ Consistency :
→ latency : few seconds
→ Concurrency : eight

API
- search (src, dest, date)
  ⇒ list of trains (available)
  ⇒
- Book (train_id, date, seat, user_id, src, dest, date)
- cancel (ticket-id, user-id)

Search → Availability → Book
Cancel
Inventory Book
Bookings
Payment

- get Status (ticket-id, user-id)

CQRS: Command Query Responsibility Segregation

Reads — Write
Consistency
search
Calling
M1 static : trains
M2 dynamic : Bookings/Avail
Scalability
replicated

Sharding?
① Train-id : T1 (NYM-DEL) T2 (GOA - MUMB) 10 Mbps
② seat-id : X
③ user-id : X
④ city
⑤ date
⑥ date → 11:30pm → 12:30am

QPS
storage

Replicas

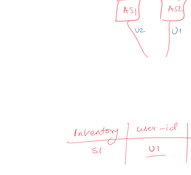
leader — writes
follower — reads

R+W > RF
RF = 3
R = 2
W = 2
Banking
3W + 1R — Yes
3R + 1W — No
2R + 2W — Comment

100 → 100,000

TTL is only for 10 mins

10,000 req
↳ persist in DB
offload actual seat allocation to a later time.
then run batch job based on their current ts.
100 start rejecting future request.



Inventory	user-id	ts
S1	U1	?

Bookings		user-id	time-created	is-paid
S1	May 1	U1	t=0	
S2	May 1	U2	t=5	
S3	"			
S4	"			
S5	"	U2	10	

t=0
t=5
t=10

5 mins

t = 600
t = 300
900

600 - 7
= 593
7300

5 min
run cron job