LLD - Concurrency with BMS

1. Booking flow for BMS
2. Concurrency
3. Race conditions
4. Types of RC
5. Synchronisation

Process & Thread

- memory space
- register
- address
- ...

Process

new Process()

- heavy weight
- switching between processes is not

Threads

- lightweight process
- context switching

Sequential

Parallel

$$\max(T1, T2)$$

IO → Parallel

no cpus * # no of cores

2 * 4 = 8

cpu - single core

dual cue

Concurrent execution

T2

CPU handle multiple tasks

in the same time period

concurrent & parallel execution

Parallel execution vs Parallelism

T1#1
T2
T1
T1#2
T1#3
Cone 1
Cone 2
Cone 3

Bitcoin mining

get Users (1 - 100)

get (Users)

get (1 - 10)

get (10 - 20)

⋮

JAVA memory model

Thread Stack

Heap

Shared

memory

Race conditions

BMS

To allow only person to access booking the ticket at the same time

when multiple threads try to access the same resource at the same time

Types of race conditions

CHECK - THEN - ACT

Race condition

↳ Expectation

Only one thread should be able to update

while the other should be told there is an error

Tie bet will not

Race condition

be available

The process of allowing only one thread to access a resource at a time

synchronisation

Performance

↳ Database

↳ Distributed locks
mutexes

---

1. Read this. cpvote into register
(T1 A=12)(B)
2. Modified this value (1)
3. T1 & T2 tried to write it back

READ-MODIFY-WRITE

{c: 2}

c: 3

→ value = map["c"]

new_value = value + 1

map["c"] = new_value

T1	T2
1	1
2	2
2	2

How can we solve this?

Race Condition

↳ Synchronisation

CS - atomic instruction
- only one thread
at atime