Concurrency - 1
Processes and Threads

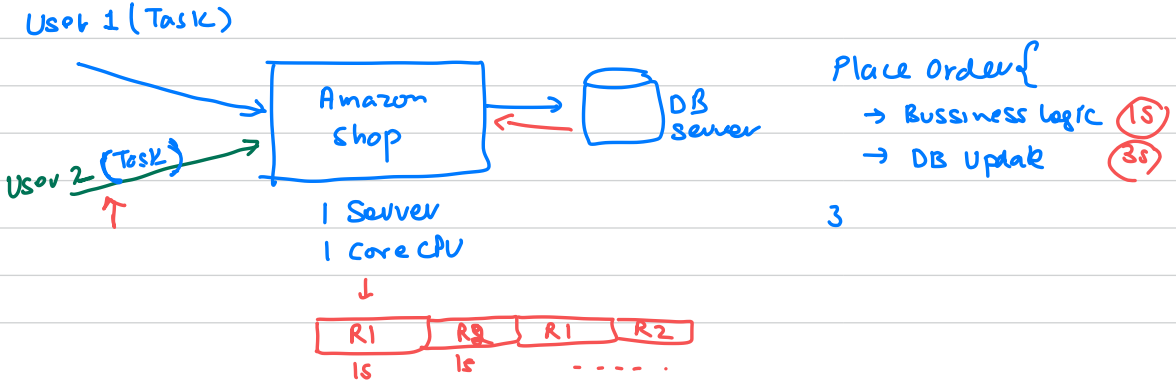
Application :

① Responsive

- you press some button, you should get response quickly

② Performance

- perform all tasks well.



can't process user2 order until user1 order is finished.

If we do multi-tasking we can many users at same time.

↳ making burgers on multiple tasks at same time.

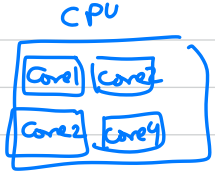
⇒ 1-core machine

↓

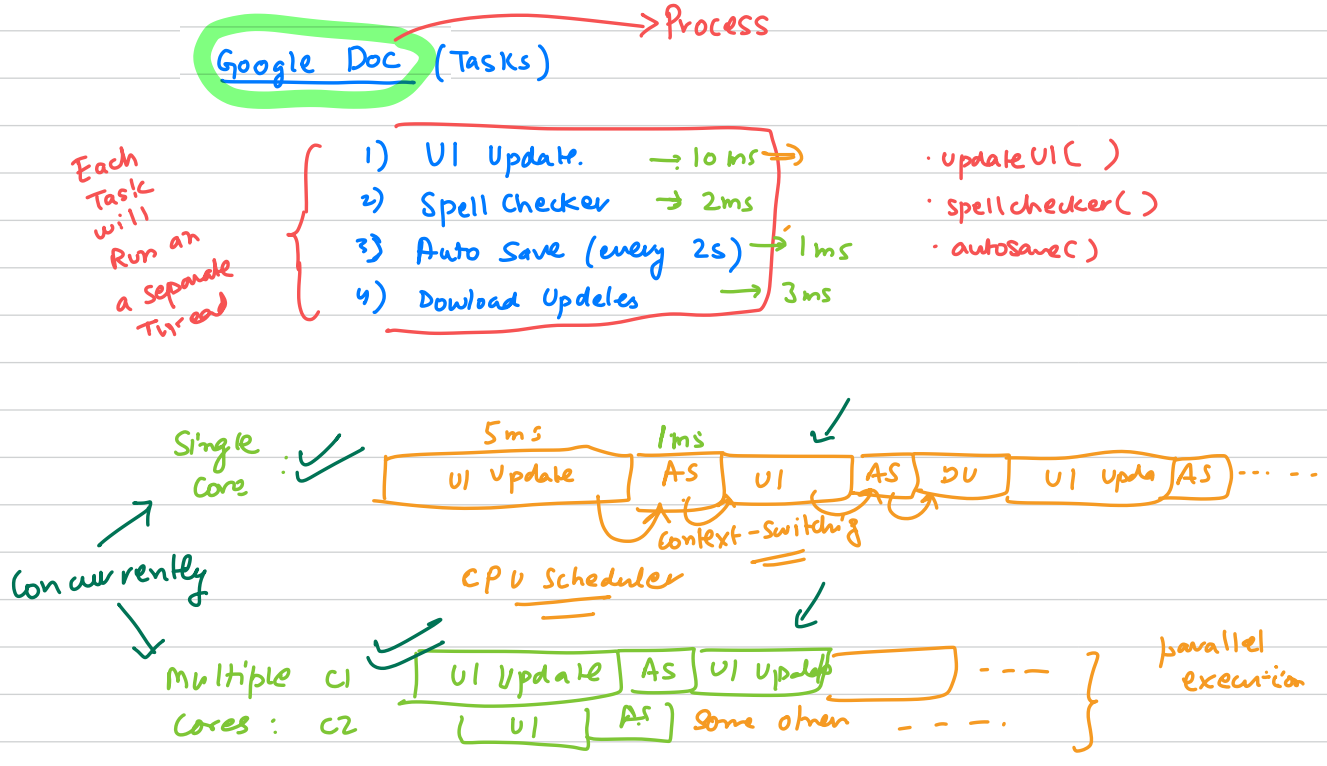
we can do multi-tasking

but we are creating an illusion of running tasks in parallel.

Multi-tasking Concurrency



Responsiveness is critical in applications where UI is involved.



Multi-threaded programming

With multiple cores, we can have improved performance

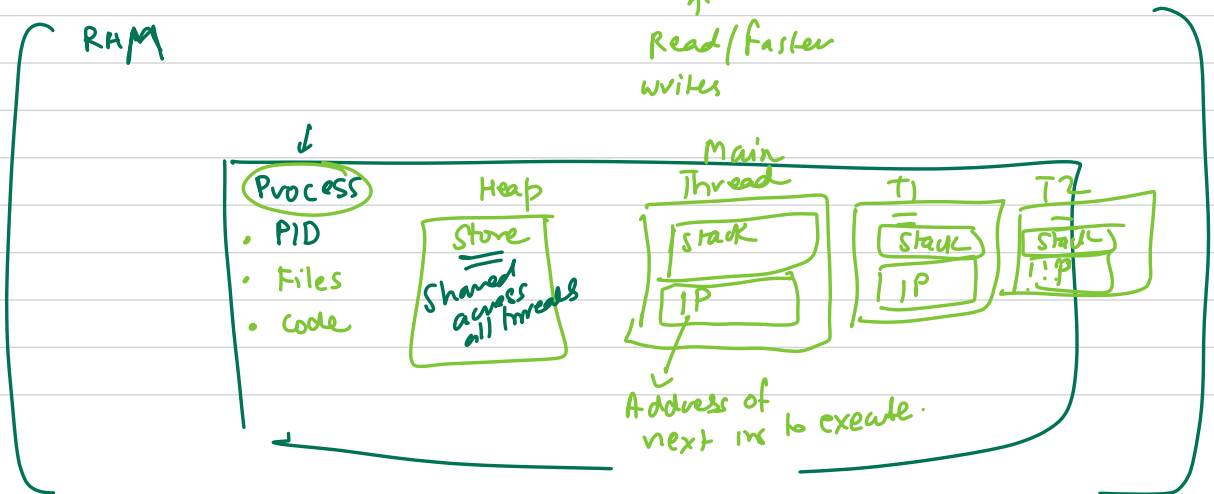
- complete a complex task much faster
- finish more work in same period of time
- build high scale services

Launch Process of an application:

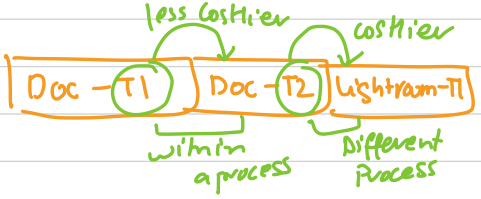
Install Software → Disk

Lauch App / Software

- creates an instance in memory, called as Process.



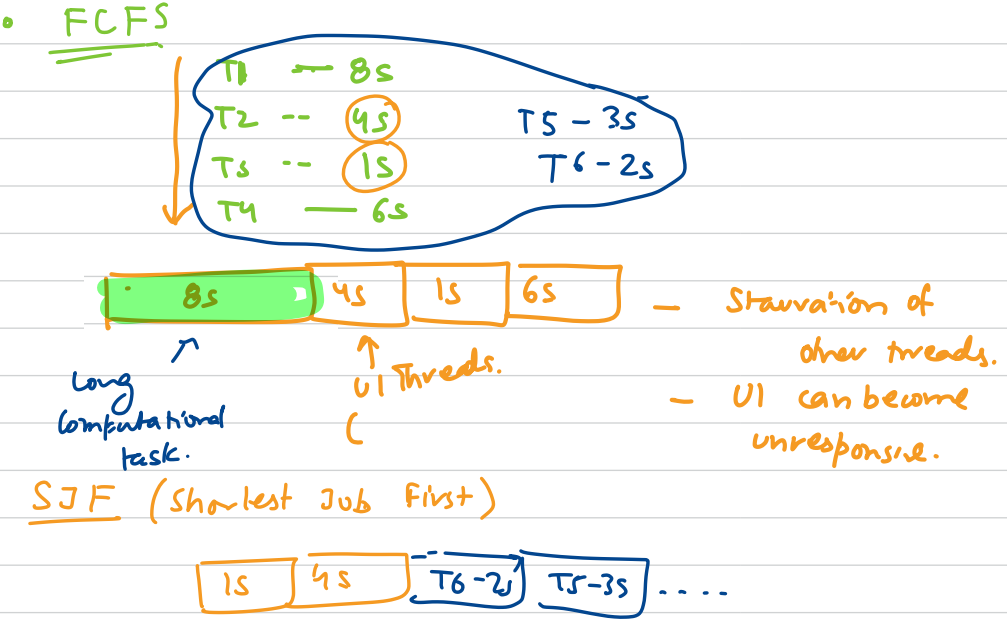
1. Each applications is one process (in general)
2. Each process can have multiple threads, each thread is like a light-weight process
3. In general more threads running/waiting than CPU cores. Threads have a state.
4. Context Switching: Time to switch from one thread to another,

1. 

Thrashing: Too many context switching, can cause thrashing - CPU is spending more time in switching the threads than doing actual work.

CPU Scheduling

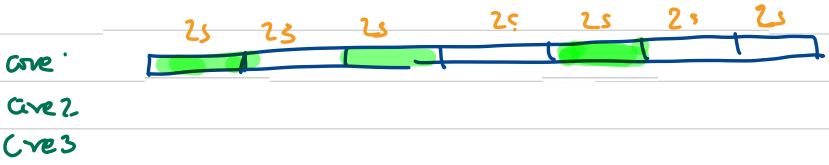
- FCFS



- SJF (Shortest Job First)

Longer jobs may never get a chance to execute.

Modern CPU



Divide the time into fixed size intervals called Epoch

Dynamic scheduling = Static Priority + Bonus (+ve -ve)

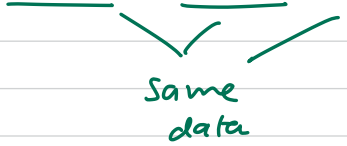
↓ set by developer

↓ OS gives bonus to more active threads, computational threads.

Multi-threaded Architecture:

- Prefer if tasks share a lot of data

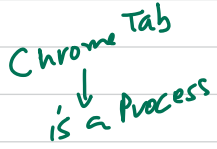
  Google Doc: UI update, AutoSave, Spell-Check



- Creating threads is faster, destroying threads is faster than processes
- Switching between threads of same process is faster

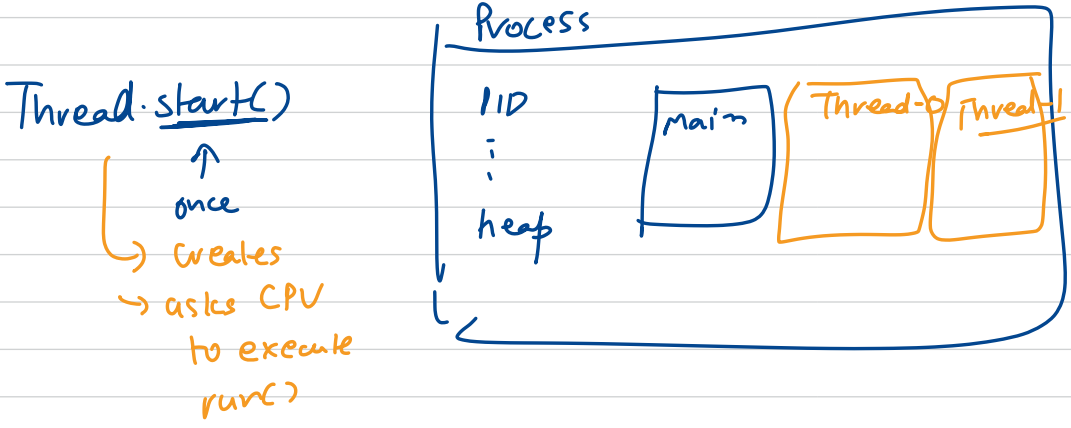
Multi-processes Architecture:

- Security and stability are of higher importance
- Tasks are unrelated to each other .



10:30

Coding


Thread.start()
once
creates
asks CPU
to execute
run()
Process
PID
heap
main
Thread-0
Thread-1