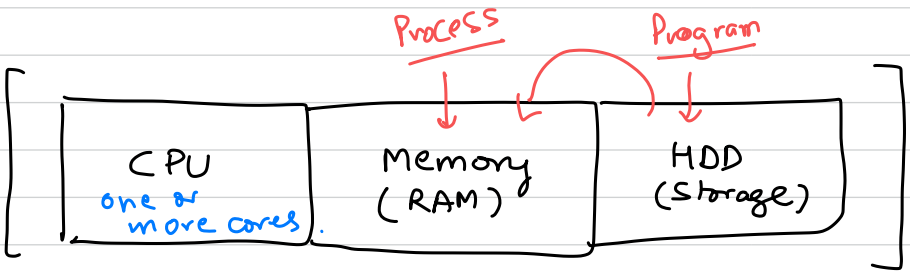
Concurrency - 1

(starting 9.05)

- How computer applications Run.
- Processes vs Threads
- Concurrent Execution, Pavallel Execution
- Multi-threading in Java
  - Thread Creation
  - Code examples

Typed Notes

I How computer applications RUN?



Core → Mini CPU

1 Core = 1 CPU
Quadcore = 4 CPU
Octacore = 8 CPU

1. Program: set of instructions

   - Google Chrome
   - MS word
   - Photoshop

2. When you launch a program, it becomes "Process".

   A process is an ~~it~~ instance of program in execution.

   ↓

take memory (RAM)

③ Context Switching

Thread. Scheduler : Dynamic scheduling

Priority

↓

more CPU time to apps getting actively used at current moment.

1 core CPU

actively used

apps

Zoom | Notes App | Browser | Light room | UI

40ms | 10ms | 40ms | 10ms | 4ms

Zoom | Notes | Zoom | Browser | LightRoom → time

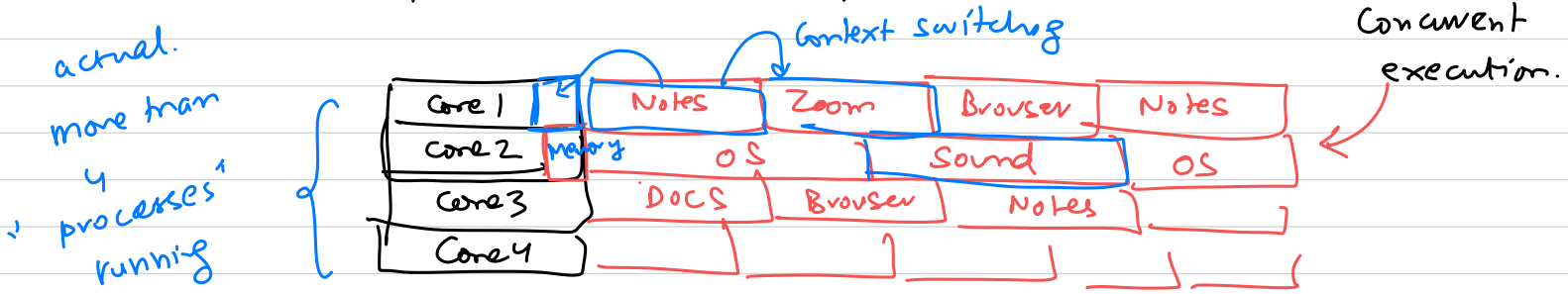
feel of "parallelism"

concurrent execution

④ Multi-tasking : Ability of comp to execute multiple processes concurrently.

(5) Parallel execution

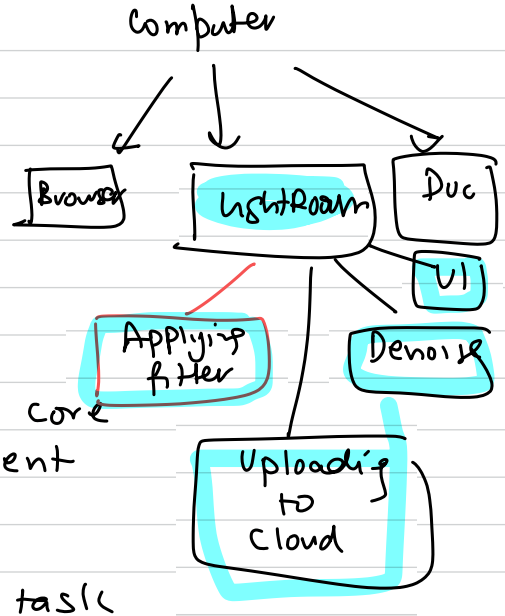
Multiple processors, true parallelism can occur.



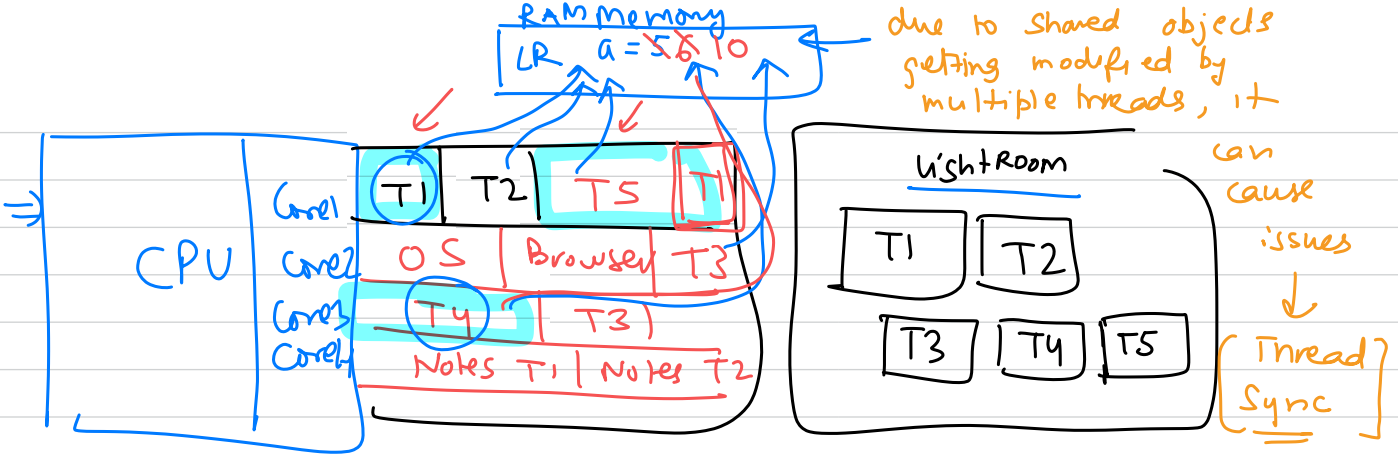
Same App → Run on multiple cores at same time?

depends whether the app is multi-threaded or not?

6 **Thread** : A process can be further divided into threads.



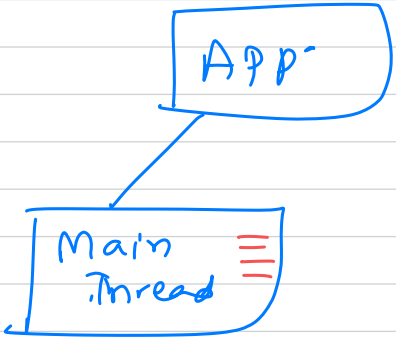
- Threads within the same process share memory resources but can execute independently.
- Each thread 'can' run on separate core to achieve 'parallelism' / concurrent execution.
- Each thread is separate indepent task within the process.



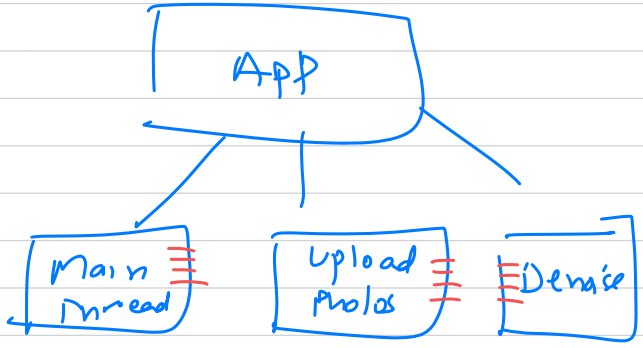
→ All threads of same process will share memory.

→ Sync Problems
↓ solutions
[Locks, Semaphores]

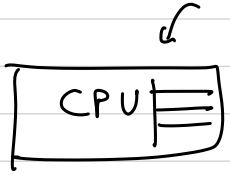
Program
↙ ↘
(1 Thread) (Multi-threaded)
Single Threaded
↓
Main Thread.
↓
Main Thread + Additional Threads



Single Threaded app.



Multi-threaded app



OS keeps track of running processes, and their threads and ensures fair access to CPU

↓

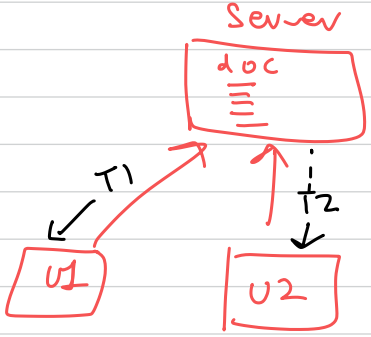
Thread schedule:

Self study (OS)

1) First Come First Serve (FCFS)
2) Shortest Job First (SJF)
3) Round Robin
4) Priority Scheduling.

II Read Applications

① Google Docs :
- ✓ Edit Text (one thread for each user)
- ✓ Collaborative editing
- ✓ Auto-correct
- ✓ UI component (UI Thread)
- ✓ auto-Saving (every 10 s)



--- T3 (ensure collaboration)

→ apple
→ app

apple

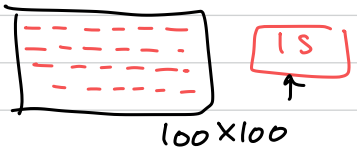
(2) Music Player

- Playback Threads: music is continuously played. uninterruped streaming. high-priority thread.
- Fetch Playlists from cloud
- UI Thread. - UI Related Actions ensuring responsive UI.

(3) Lightroom

70 threads

→ Image Processing Threads.

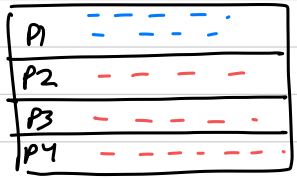


→ Apply filter using 4 Threads.

More CPU Intensive

↓

Low 'latency' to end user.

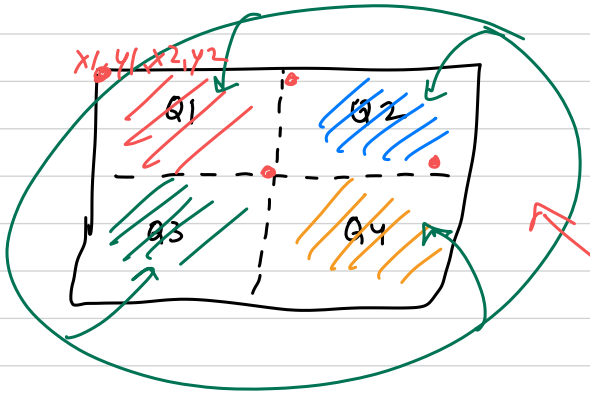


4 Threds.

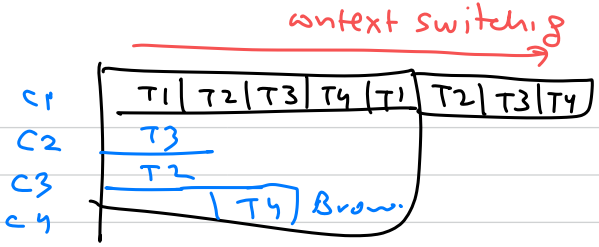
0.25s

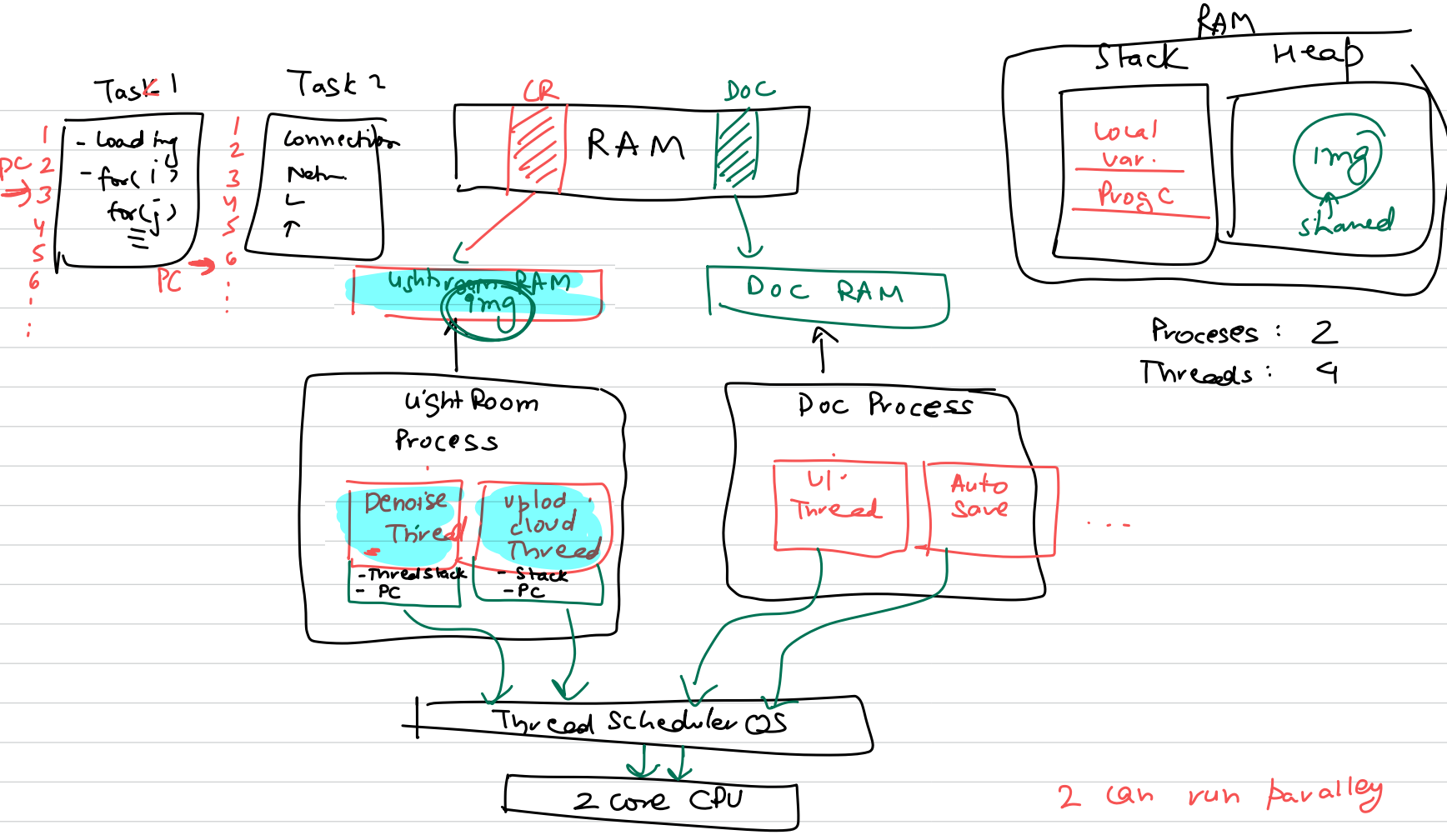
0.4s

apply Filter (img, x1, y1, x2, y2)



all 4 Threads have a quad.
1 Img
memory
which is not
shared.




Task 1
- Loading
- for(i)
for(j)
PC
Task 2
Connection
Netw
PC
CR
Doc
RAM
Light Room RAM
img
Doc RAM
Light Room Process
Denoise Thread
- Thread Stack
- PC
upload cloud Thread
- Stack
- PC
Doc Process
UI Thread
Auto Save
...
Thread Scheduler OS
2 core CPU
RAM
Stack
Heap
Local Var.
Prog C
img
shared
Processes : 2
Threads : 4
2 can run paralley

Advantages of multithreading:

1) Performance : Concurrent Executions enhances performance.
2) Responsiveness:
   Computation + UI
   UI can run a different thread, hence your app will be responsive.

3) Offload long runnings tasks to a separate thread,
   uploading photos to cloud, auto-saving doc every 5 s.

4) Web Backend - each thread can be used handle one connection.
   Multi-requests can be handled simultaneously.

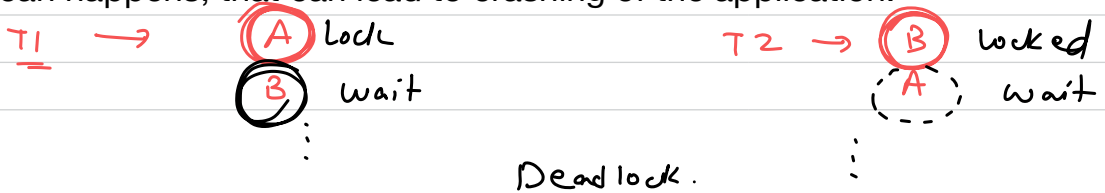
5) Since threads within the process share resources, it reduces memory overhead.

~~Disadvantages:~~
1) Put additional efforts to synchronise data, to prevent consistency issues.
2) Deadlocks can happens, that can lead to crashing of the application.

```
class Counter {
    int count;
}

Counter c = new Counter()
```

Obj live on heap and can be shared across threads



10:30 

Code in Java.