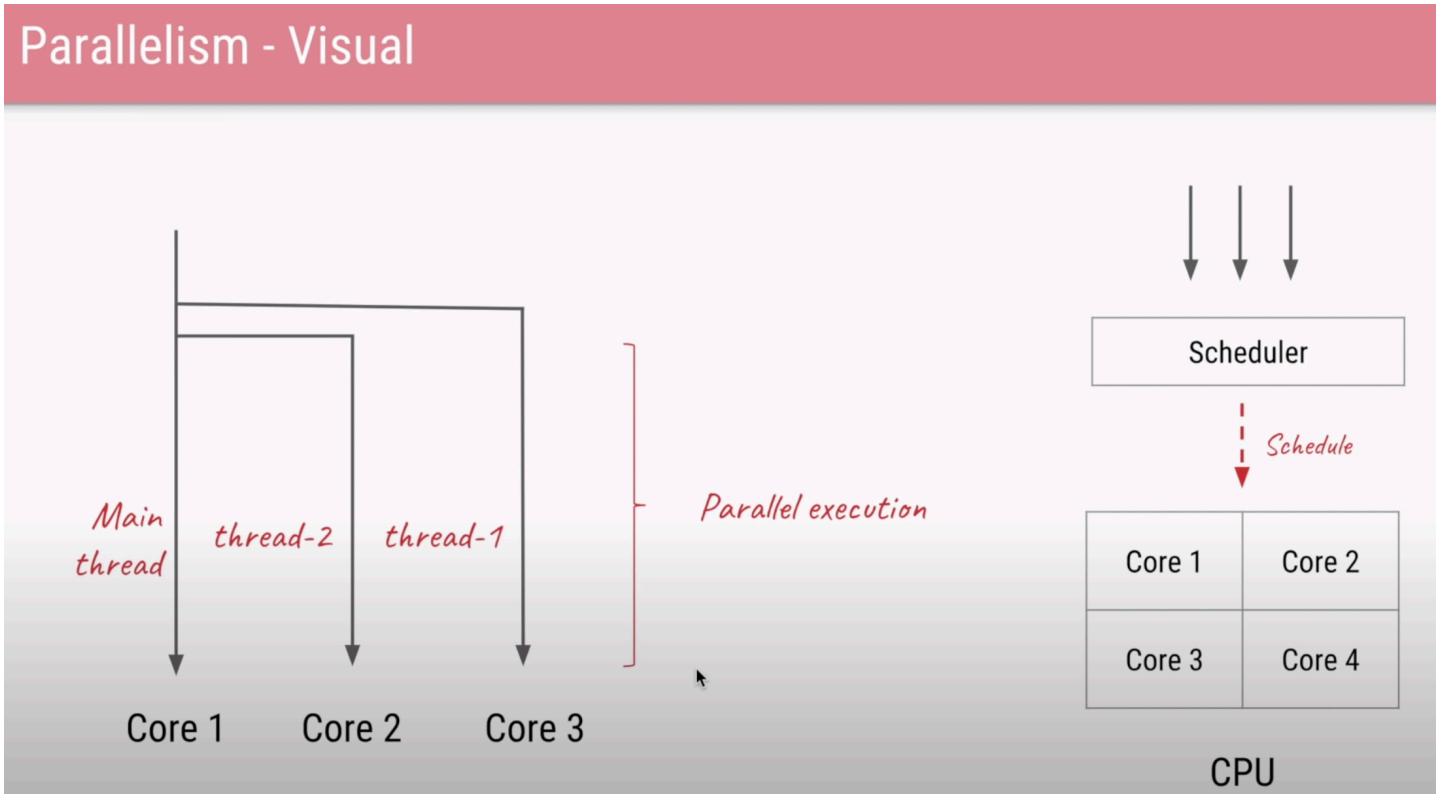
Multithreading 2 : Concurrency vs Parallelism

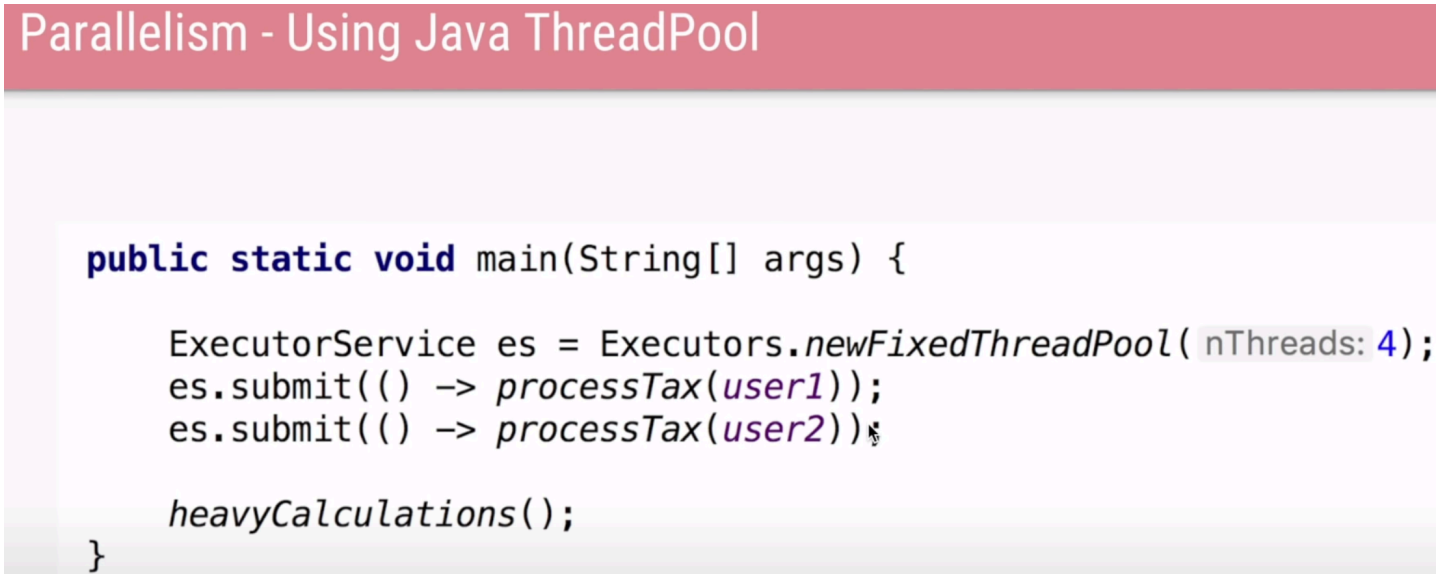
Press "/" to insert an element

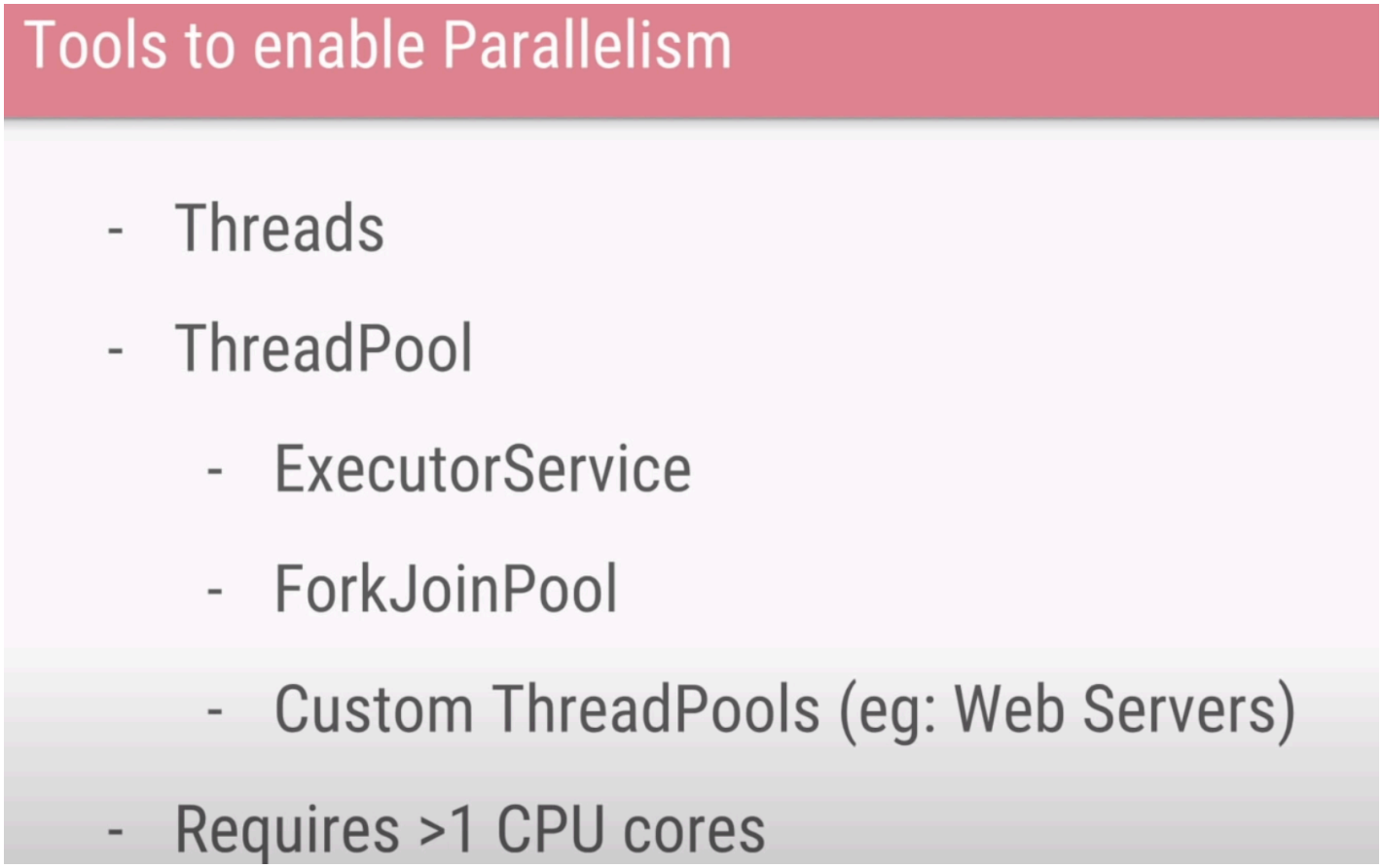
Parallelism:

Parallelism means doing a lot of things at once



As we are 4 core processors then we could execute 3-4 task parallel.





Concurrency:

Concurrency is dealing a lots of thing at once.

1. **Like shared resources are need to be updated**
2. **Multiple tasks need to be coordinated**

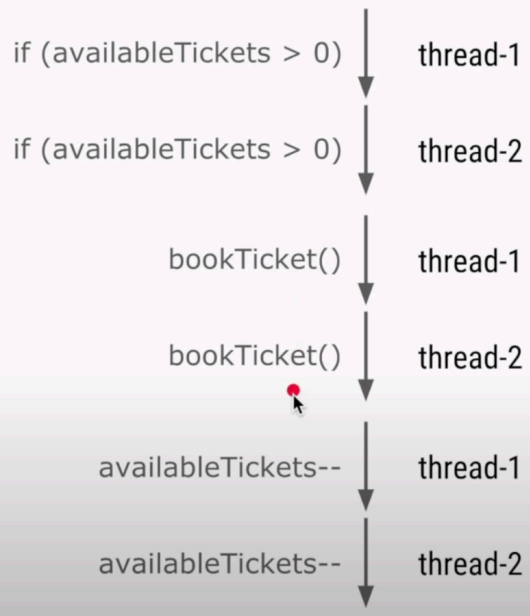
Concurrency - Code

```
public static void main(String[] args) throws InterruptedException {

    new Thread(() -> {
        if (ticketsAvailable > 0) {
            bookTicket();
            ticketsAvailable --;
        }
    }).start();

    new Thread(() ->  {
        if (ticketsAvailable > 0) {
            bookTicket();
            ticketsAvailable --;
        }
    }).start();

    Thread.sleep( millis: 5000);
}
```

Task accessing shared variable

Task accessing shared variable

Concurrency - Visual



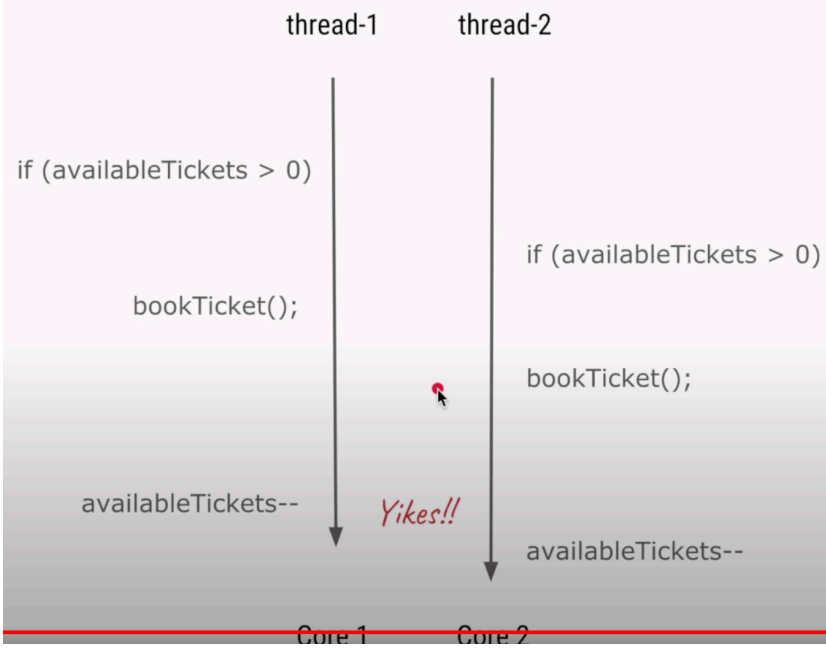
This could also happen.

Yikes! Wrong result.

For single core application thread, interleaving is there.

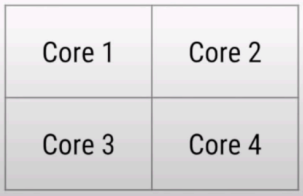
For multicore processor:

Concurrency - Visual



Thread scheduling is not in developers control.

Which instructions are being run by a thread is also non-deterministic.



Concurrency - Code - Fix

```
new Thread(() -> {
    lock.lock();
    if (ticketsAvailable > 0) {
        bookTicket();
        ticketsAvailable --;
    }
    lock.unlock();
}).start();

new Thread(() ->  {
    lock.lock();
    if (ticketsAvailable > 0) {
        bookTicket();
        ticketsAvailable --;
    }
    lock.unlock();
}).start();
```

Locks for threads to coordinate. So that they don't step on each other.

One of the possible solutions

Tools to deal with concurrency

- Locks / synchronized
- Atomic classes
- Concurrent data structures (eg: ConcurrentHashMap, BlockingQueue)
- CompletableFuture
- CountdownLatch / Phaser / CyclicBarrier / Semaphore etc.

Concurrency + Parallelism

- Split the sequential flow into independent components
- Use threads/threadpools to parallelize (& thus speed up)
- Whenever share resource is to be updated, use concurrency tools to manage state
- Whenever independent components (running on threads) need to coordinate, use concurrency tools