

Friday, 22 March 2024

# Multithreading, Concurrency & Parallel programming

## Why we even need multiple, both threads.

The two main reasons we need threads for -

- Responsiveness
- Performance

## Responsiveness

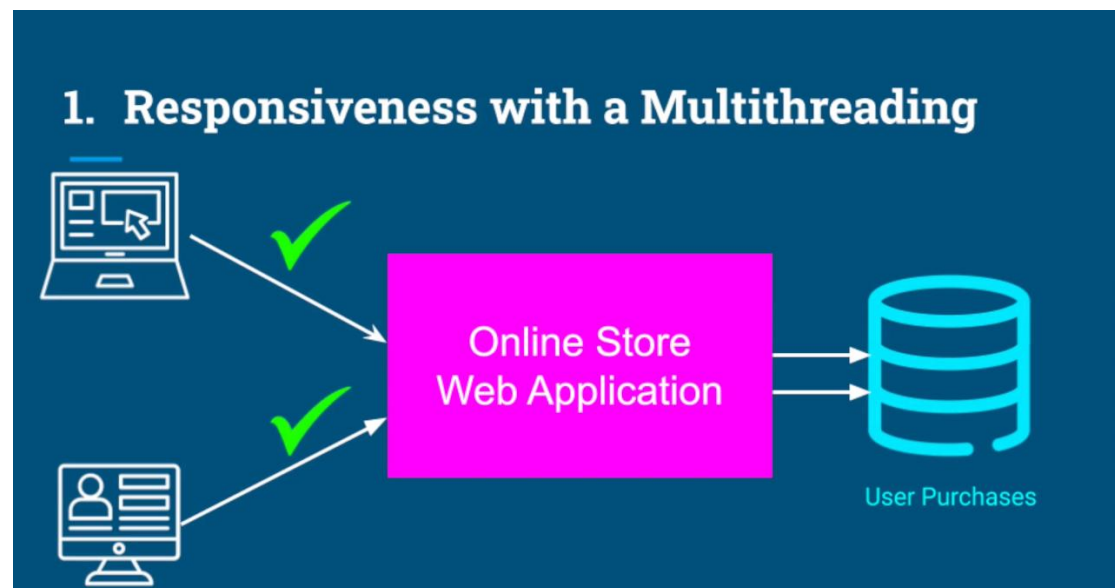
### 1. Examples of Poor Responsiveness

- Waiting for Customer Support
- Late response from a person
- No feedback from an application

### 1. Responsiveness with a Single Thread



With multithreading we could actually serve multiple users simultaneously, but serving each request on a different thread.



Responsiveness is particularly critical when it comes to applications with a [user interface](#). A good example for that can be a movie player application. The application is showing us images, playing the audio. And in the same time, we expect that if we move the mouse or click a button, we would get an instant feedback for our actions on the screen.

### 1. Responsiveness in User Interface

- Responsiveness is particularly critical in applications with a User Interface

The diagram features a movie player interface on the left, consisting of a video frame with a play button and a progress bar below it. To the right of the player is a red computer mouse. The entire scene is set against a dark blue background.

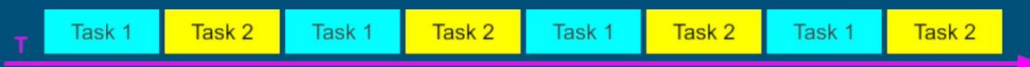
This kind of responsiveness can be achieved by using **multiple threads**, each thread for a different task.

By multitasking quickly between different threads, Our computer can create an illusion that all those tasks are actually happening in the same time.

The term we use for this kind of multitasking is **concurrency**.

## 1. Concurrency - Multitasking

- Achieved by multi-tasking between threads
- Concurrency = Multitasking



we don't even need **multiple cores** to achieve concurrency. Even with **one core**, we can create responsive applications by using **multiple threads**.

## 1. Concurrency - Multitasking

- Note : We don't need multiple cores to achieve *concurrency*

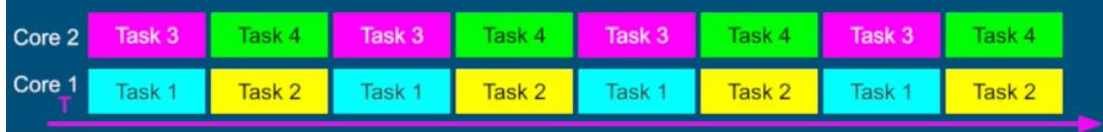


## Performance

As mentioned before using concurrency, we can create an illusion of multiple tasks running in parallel just with single core. With multiple core we can truly run multiple tasks completely in parallel.

### 2. Performance

- We can create an illusion of multiple tasks executing in parallel using just a single core
- With **multiple cores** we can truly run tasks completely in parallel



### 2. Performance - Impact

- Completing a complex task much faster
- Finish more work in the same period of time
- For high scale service -
  - Fewer machines
  - Less money spent on hardware
  - More money in your pocket

Responsiveness - Concurrency  
Performance - Parallelism

