Concurrency vs Parallelism

This lesson clarifies the common misunderstandings and confusions around concurrency and parallelism.

Introduction

Concurrency and *Parallelism* are often confused to refer to the ability of a system to run multiple distinct programs at the same time. Though the two terms are somewhat related yet they mean very different things. To clarify the concept, we'll borrow a juggler from a circus to use as an analogy. Consider the juggler to be a machine and the balls he juggles as processes.

Serial Execution

When programs are serially executed, they are scheduled one at a time on the CPU. Once a task gets completed, the next one gets a chance to run. Each task is run from the beginning to the end without interruption. The analogy for serial execution is a circus juggler who can only juggle one ball at a time. Definitely not very entertaining!

Concurrency

A concurrent program is one that can be decomposed into constituent parts and each part can be executed out of order or in partial order without affecting the final outcome. A system capable of running several distinct programs or more than one independent unit of the same program in overlapping time intervals is called a concurrent system. The execution of two programs or units of the same program may not happen simultaneously.

A concurrent system can have two programs *in progress* at the same time where *progress* doesn't imply execution. One program can be suspended while the other executes. Both programs are able to make progress as their execution is interleaved. In concurrent systems, the goal is to maximize throughput and minimize latency. For example, a browser running on a single core machine has to be responsive to user clicks but also be able to render HTML on screen as quickly as possible. Concurrent systems achieve lower latency and higher throughput when programs running on the system require frequent network or disk I/O.

The classic example of a concurrent system is that of an operating system running on a single core machine. Such an operating system is concurrent but not parallel. It can only process one task at any given point in time but all the tasks being managed by the operating system *appear* to make progress because the operating system is designed for concurrency. Each task gets a slice of the CPU time to execute and move forward.

Going back to our circus analogy, a concurrent juggler is one who can juggle several balls at the same time. However, at any one point in time, he can only have a single ball in his hand while the rest are in flight. Each ball gets a time slice during which it lands in the juggler's hand and then is thrown back up. A concurrent system is in a similar sense *juggling* several processes at the same time.

Parallelism

A parallel system is one which necessarily has the ability to execute multiple programs **at the same time.** Usually, this capability is aided by hardware in the form of multicore processors on individual machines or as computing clusters where several machines are hooked up to solve independent pieces of a problem simultaneously. Remember an individual problem has to be concurrent in nature, that is portions of it can be worked on independently without affecting the final outcome before it can be executed in parallel.

In parallel systems the emphasis is on increasing throughput and optimizing usage of hardware resources. The goal is to extract out as much computation speedup as possible. Example problems include matrix multiplication, 3D rendering, data analysis, and particle simulation.

Revisiting our juggler analogy, a parallel system would map to at least two or more jugglers juggling one or more balls. In the case of an operating system, if it runs on a machine with say four CPUs then the operating system can execute four tasks at the same time, making execution parallel. Either a single (large) problem can be executed in parallel or distinct programs can be executed in parallel on a system supporting parallel execution.

Concurrency vs Parallelism

From the above discussion it should be apparent that a concurrent system need not be parallel, whereas a parallel system is indeed concurrent. Additionally, a system can be both concurrent and parallel e.g. a multitasking operating system running on a multicore machine.

Concurrency is about *dealing* with lots of things at once. Parallelism is about *doing* lots of things at once. Last but not the least, you'll find literature describing concurrency as a property of a program or a system whereas parallelism as a runtime behaviour of executing multiple tasks.

We end the lesson with an analogy, frequently quoted in online literature, of customers waiting in two queues to buy coffee. Single-processor concurrency is akin to alternatively serving customers from the two queues but with a single coffee machine, while parallelism is similar to serving each customer queue with a dedicated coffee machine.

← Back lesson
Program vs Process vs Thread

Completed Next →
Cooperative Multitasking vs Preemptive Multitasking