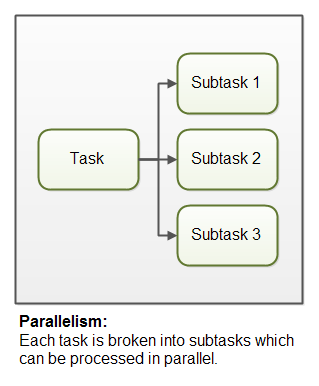
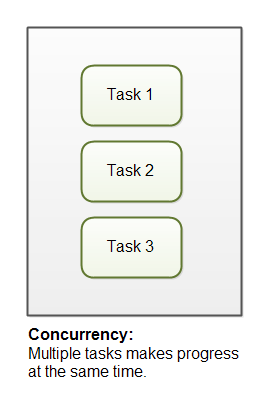
**Concurrency vs. Parallelism**

The terms *concurrency* and *parallelism* are often used in relation to multithreaded programs. But what exactly does concurrency and parallelism mean, and are they the same terms or what?

The short answer is "no". They are not the same terms, although they appear quite similar on the surface. It also took me some time to finally find and understand the difference between concurrency and parallelism. Therefore I decided to add a text about concurrency vs. parallelism to this Java concurrency tutorial.

## **Concurrency**

Concurrency means that an application is making progress on more than one task at the same time (concurrently). Well, if the computer only has one CPU the application may not make progress on more than one task at *exactly the same time*, but more than one task is being processed at a time inside the application. It does not completely finish one task before it begins the next.



## **Parallelism**

Parallelism means that an application splits its tasks up into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time.

## Concurrency vs. Parallelism In Detail

As you can see, concurrency is related to how an application handles multiple tasks it works on. An application may process one task at at time (sequentially) or work on multiple tasks at the same time (concurrently).

Parallelism on the other hand, is related to how an application handles each individual task. An application may process the task serially from start to end, or split the task up into subtasks which can be completed in parallel.

As you can see, an application can be concurrent, but not parallel. This means that it processes more than one task at the same time, but the tasks are not broken down into subtasks.

An application can also be parallel but not concurrent. This means that the application only works on one task at a time, and this task is broken down into subtasks which can be processed in parallel.

Additionally, an application can be neither concurrent nor parallel. This means that it works on only one task at a time, and the task is never broken down into subtasks for parallel execution.

Finally, an application can also be both concurrent and parallel, in that it both works on multiple tasks at the same time, and also breaks each task down into subtasks for parallel execution. However, some of the benefits of concurrency and parallelism may be lost in this scenario, as the CPUs in the computer are already kept reasonably busy with either concurrency or parallelism alone. Combining it may lead to only a small performance gain or even performance loss. Make sure you analyze and measure before you adopt a concurrent parallel model blindly.