Concurrency and parallelism are two concepts related to programming languages and the execution of programs.

1. Concurrency: Concurrency refers to the ability of a program to execute multiple tasks or processes concurrently, seemingly simultaneously. In concurrent programming, tasks are divided into smaller units of work that can be executed independently and potentially in overlapping or interleaved fashion. Concurrency allows for better utilization of system resources and improved responsiveness.
2. Parallelism: Parallelism involves the actual simultaneous execution of multiple tasks or processes. It leverages multiple physical or virtual processing units (such as multiple CPU cores) to execute tasks in parallel, achieving faster execution and improved performance.

Go (Golang) is often referred to as a concurrent programming language, and it has built-in features and mechanisms to support concurrent programming. Here's why Go is considered a concurrent language:

3. Goroutines: Go introduces a lightweight concurrency construct called "goroutines." Goroutines are functions or methods that can be executed concurrently with other goroutines. They are similar to threads but more lightweight and efficient. Goroutines enable concurrent execution of tasks without the overhead associated with traditional operating system threads.
4. Channels: Go provides channels, which are communication mechanisms used for safe and synchronized communication between goroutines. Channels allow goroutines to send and receive data, facilitating coordination, synchronization, and sharing of information.
5. Select statement: Go includes a "select" statement that allows you to coordinate and synchronize operations on multiple channels. With the select statement, you can wait for multiple channels to be ready for communication, enabling efficient and controlled communication between goroutines.

6. Goroutine scheduler: Go has a built-in goroutine scheduler that manages the execution and scheduling of goroutines across available CPU cores. The scheduler automatically distributes goroutines among the available cores, optimizing resource utilization and ensuring concurrent execution.
7. Standard library support: The Go standard library provides various concurrency-related packages and utilities, such as the "sync" package for synchronization primitives, the "context" package for managing contexts and cancellation, and the "sync/atomic" package for atomic operations.

Go's focus on concurrency makes it well-suited for developing concurrent applications, network servers, web applications, and systems software. Its concurrency features allow developers to write concurrent code more easily, handle multiple requests or tasks concurrently, and build scalable and efficient systems.

It's important to note that while Go has strong support for concurrency, it also enables parallelism by utilizing multiple CPU cores. Go programs can leverage parallelism when performing computationally intensive tasks by utilizing goroutines and parallel algorithms.