Go语言 并发简述

Go 从语言层面就支持并发。同时实现了自动垃圾回收机制。

Go语言的并发机制运用起来非常简便，在启动并发的方式上直接添加了语言级的关键字 `go` 就可以实现，和其他编程语言相比更加轻量。

Go 语言通过编译器运行时（runtime），从语言上支持了并发的特性。Go 语言的并发通过 **goroutine** 特性完成。goroutine 类似于线程，但是可以根据需要创建多个 goroutine 并发工作。goroutine 是由 Go 语言的运行时调度完成，而线程是由操作系统调度完成。

Go 语言还提供 **channel** 在多个 goroutine 间进行通信。goroutine 和 channel 是 Go 语言秉承的 CSP（Communicating Sequential Process）并发模式的重要实现基础。

进程/线程

进程是程序在操作系统中的一次执行过程，系统进行资源分配和调度的一个独立单位。

线程是进程的一个执行实体，是 CPU 调度和分派的基本单位，它是比进程更小的能独立运行的基本单位。

一个进程可以创建和撤销多个线程，同一个进程中的多个线程之间可以并发执行。

并发/并行

多线程程序在**单核心**的 cpu 上运行，称为**并发**；

多线程程序在**多核心**的 cpu 上运行，称为**并行**。

并发（concurrency）：把任务在不同的时间点交给处理器进行处理。在同一时间点，任务并不会同时运行。

并行（parallelism）：把每一个任务分配给每一个处理器独立完成。在同一时间点，任务一定是同时运行。

并发与并行并不相同，并发主要由切换时间片来实现“同时”运行，并行则是直接利用多核实现多线程的运行，Go程序可以设置使用核心数，以发挥多核计算机的能力。

并发不是并行。并行是让不同的代码片段同时在不同的物理处理器上执行。并行的关键是同时做很多事情，而并发是指同时管理很多事情，这些事情可能只做了一半就被暂停去做别的事情了。

协程/线程

协程：独立的栈空间，共享堆空间，调度由用户自己控制，本质上有点类似于用户级线程，这些用户级线程的调度也是自己实现的。

线程：一个线程上可以跑多个协程，**协程是轻量级的线程**。

协程（Goroutine）与线程相比，它的开销非常小。