Concurrency in Golang : http://hamait.tistory.com/934

summary : Do not communicate by sharing memory; instead, share memory by communicating.

Do not Communicate by sharing memory

- C++이나 Java 를 이용한 대규모 동시성 프로그램을 작성할 때 주로 사용되는 가장 기본적인 동시성 모델이며 뮤텍스와 같은 동시성 객체를 이용하여 쓰레드들이 서로 나누어 가질 데이터 구조/변수/메모리등을 나누어 사용하는 형태로 2 개의 쓰레드가 동시에 접근할 수 없으며 race condition, memory management, unknown exception, deadlock 등의 문제가 발생할 수 있는 문제가 있다.

Instead, share memory by communication

- Go 는 고루틴이라는 개념이 추가되어 하나의 고루틴에서 다른 고루틴로 변수에 저장되어진 값을 Communicate(send)하게 만들어준다. 가장 기본적인 로직은 데이터를 보내는 고루틴과 데이터를 받으며 도착할때까지 기다리는 고루틴이 있으며 두 고루틴들의 기다림은 고루틴들 사이에 통신이 일어날 때 더 적합한 싱크를 만들어 준다.

- Go 는 “buffered channel”이라는 것을 지원하는데 이것에 의해 전송이 완료 될 때까지 고루틴들 간에 어떤 락이나 sync 를 맞출 필요가 없다.

Goroutine

- Go 에서 제공하는 thread 와 유사하다. 이는 실제로는 thread 가 아니며 기본적으로 동일한 주소 공간에서 다른 goroutine 과 동시에 실행할 수는 기능이다. 또한 경량화된 thread 라고도 불린다.

- Goroutine 이 thread 가 아닌 이유는 항상 병행적으로 행동되지 않기 때문이다. 하지만 multiplexing 및 동기화로 인해 동시적으로 동작이 발생 가능하여 thread 와 유사하게 동작 가능하다.

- goroutine 생성 : go myfynction()

Channel

- Go 에서 동시성을 실현하는 핵심 개념이며 CSP(순차 프로세스 통신, Communicating Sequential Processes)에서 “통신” 역할을 담당한다. 즉 채널을 통해 goroutine 간에 메모리 전달이 실현된다.

- 생성 : channel_name := make(chan int64) / make(chan int64,num) = 버퍼링된 채널
- 사용 : “<-”를 이용한다.

보내는 고루틴 : mychannel <- 54 // 채널에 값을 할당
받는 고루틴 : myVar := ← mychannel // mychannel 의 값 54 를 추출하여 myVar 에 할당