NEWSTORE

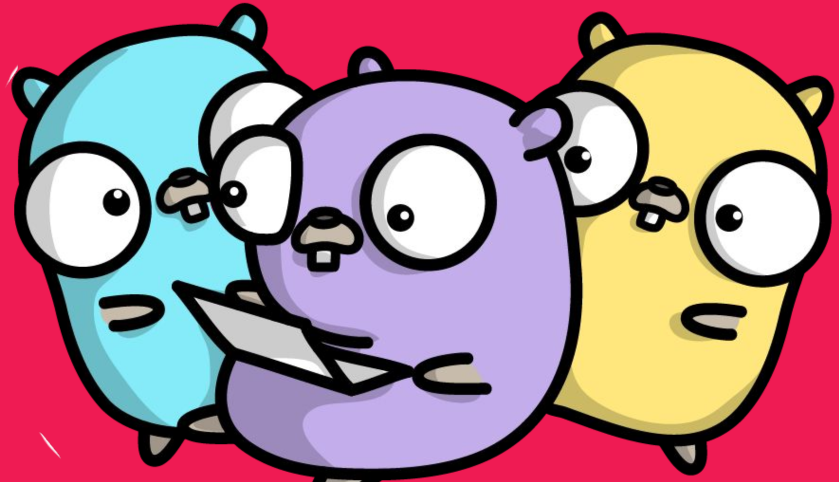
Concurrency Patterns in Go

Serdar Tahir Kabaoglu

Codes & Presentation available @
https://github.com/seredot/go-concurrency

Agenda

-> Concurrency vs Parallelism
-> CPU vs IO Bound Workloads
-> Goroutines
-> `WaitGroup`
<- Demo: WaitGroup
-> Channels
<- Demo: Throttle
-> `sync` Package
<- Demo: Worker pool
-> `select`
<- Demo: Timeout
?: Q&A



Concurrency vs Parallelism

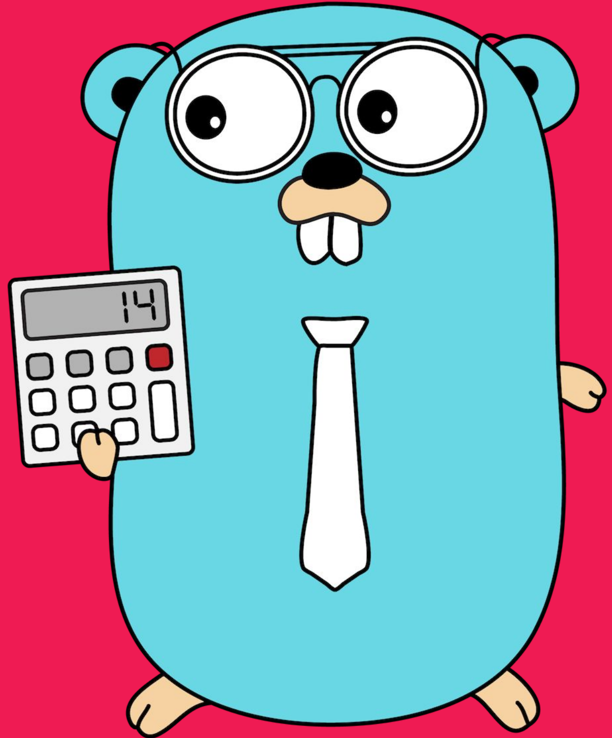
Parallelism
Execution on different processing units at the same time.

Concurrency
Execution out of order.

CPU vs IO Bound Workloads

CPU Bound
Long calculations like: finding n-th prime, compression, encoding, encryption, etc.

IO Bound
Waiting for events, network operations, database queries, file operations, locks.

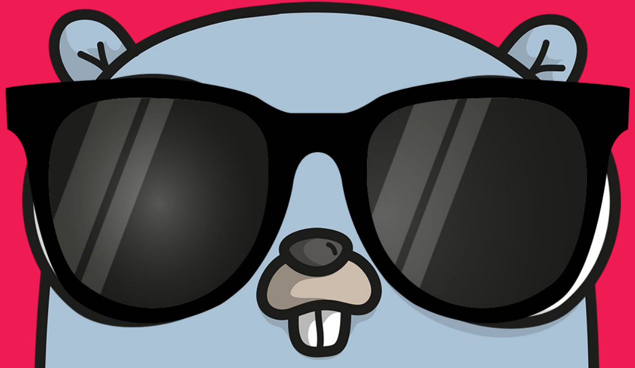


Goroutines

A Goroutine is a function that runs concurrently.

A Goroutine can be thought like a lightweight thread.

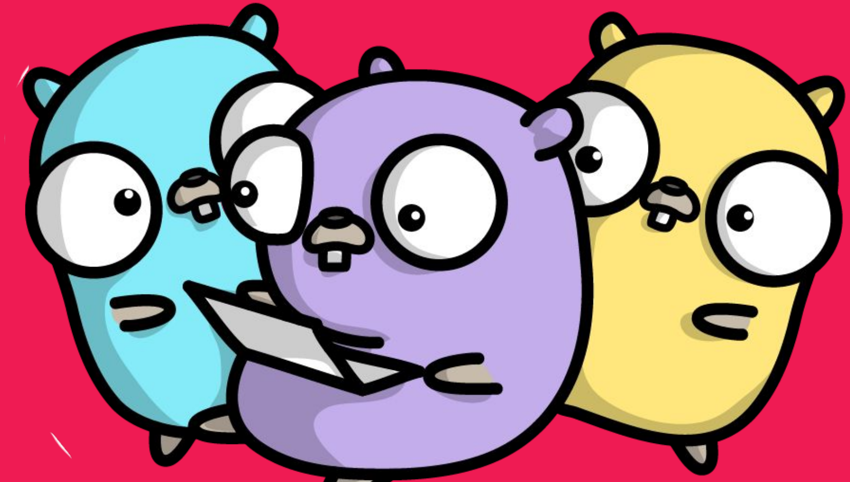
```go
go functionForSomeTask()
```



WaitGroup

A WaitGroup waits for a collection of goroutines to finish.

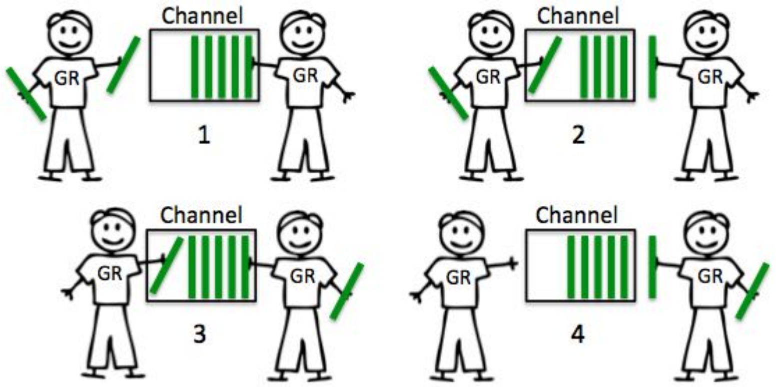
```
var wg sync.WaitGroup
wg.Add(1)
wg.Done()
wg.Wait()
```

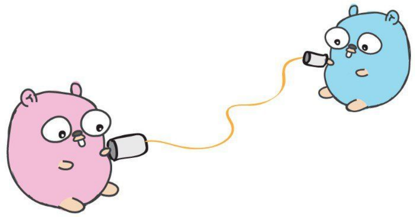


DEMO

wait-group

Channels





```go
// Create unbuffered channel.
c := make(chan string)

// Create buffered channel.
c := make(chan int, 10)

// Write to the channel.
c <- 3

// Read from the channel.
i := <-c

// Close the channel.
close(c)
```

DEMO

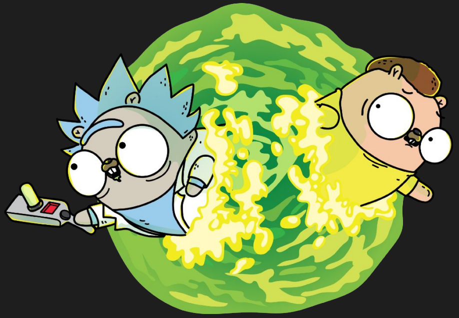
`throttle`

sync Package

```
WaitGroup
   .Add, .Done, .Wait
Mutex
   .Lock, .Unlock
RWMutex
   .RLock, RUnlock, .Lock,
.Unlock
Once
   .Do
Map
   .Load, .Store, .Delete
Cond
   .Wait, .Signal, Broadcast
```

sync/atomic Package

```
atomic.AddUint64()
atomic.LoadInt64()
atomic.StoreInt64()
atomic.SwapInt64()
atomic.CompareAddSwapint64()
atomic.Value
```



DEMO

`worker-pool`

select

Tries to read from multiple channels or write to multiple channels.

Blocks until one of the cases is available.

If multiple are available, chooses **randomly**.

```
select {
  case i := <-c1:
    fmt.Printf("Operation 1 returned %d\n", i)
  case j := <-c2:
    fmt.Printf("Operation 2 returned %d\n", j)
}
```

DEMO

`timeout`

NEWSTORE

Thank You

Concurrency Patterns in Go

Serdar Tahir Kabaoglu

Codes & Presentation available @
`https://github.com/seredot/go-concurrency`