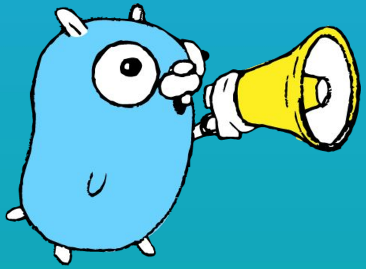
01/28/2024

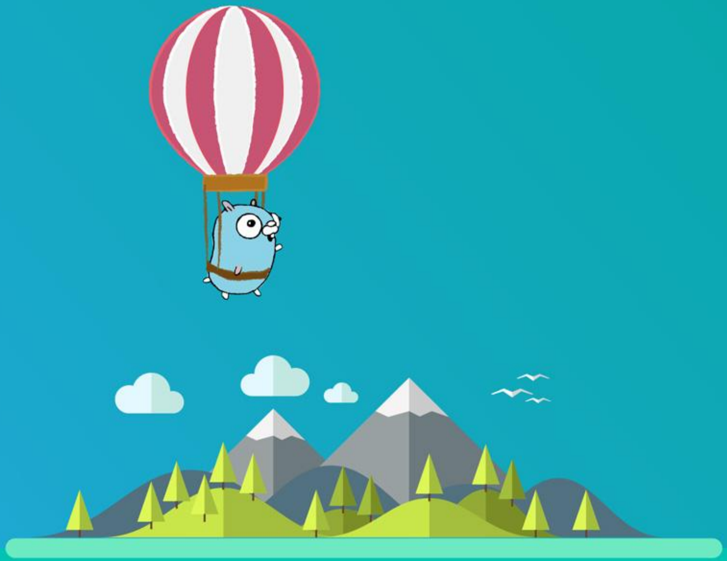
Visualization Go scheduler by gosched-simulator

sivchari

CyberAgent

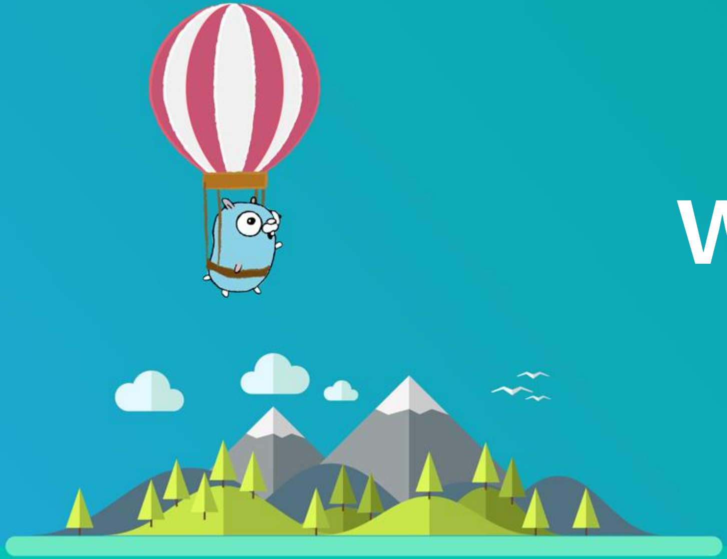
The Go gopher was designed by Renée French.





Introduction

- Takuma Shibuya
  - X/GitHub sivchari
- CIU
  - AKE (Astro Kubernetes Engine)
- CyberAgent Go Next Experts
- Go Conference main organizer



What’s Go scheduler

What’s Go scheduler

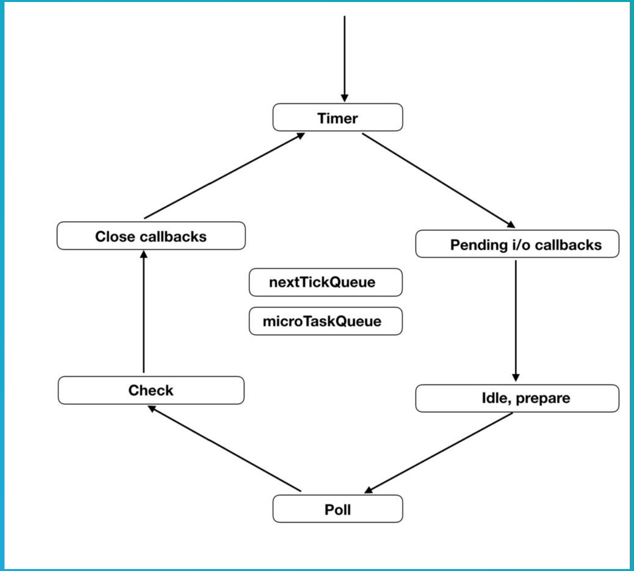
- It’s lightweight thread called by using `go` idiom
- goroutine is a minimum unit to execute Go func
  - When the program uses goroutine, the Go runtime pushes it to internal queue, then it will be executed
  - func main is a main goroutine defined in proc.go

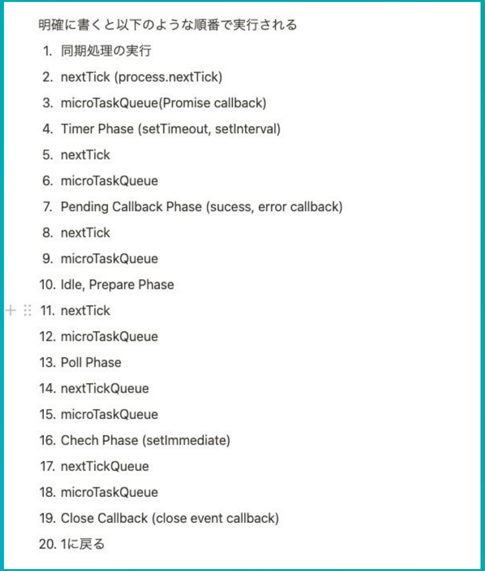
What’s Go scheduler

- goroutine is similar to coroutine
  - When the goroutine is blocking by I/O wait etc, the G runtime automatically switches it to new one
  - range-over-func uses coroutine for iteration process
    - This feature uses coro. It doesn’t have capability to execute process in parallel. coroutine can only start, switch and exit.

What’s Go scheduler

- **goroutine doesn’t have execution order, too**





明確に書くと以下のような順番で実行される

1. 同期処理の実行
2. nextTick (process.nextTick)
3. microTaskQueue(Promise callback)
4. Timer Phase (setTimeout, setInterval)
5. nextTick
6. microTaskQueue
7. Pending Callback Phase (sucess, error callback)
8. nextTick
9. microTaskQueue
10. Idle, Prepare Phase
11. nextTick
12. microTaskQueue
13. Poll Phase
14. nextTickQueue
15. microTaskQueue
16. Chech Phase (setImmediate)
17. nextTickQueue
18. microTaskQueue
19. Close Callback (close event callback)
20. 1に戻る

What's Go scheduler

- Programmer can't handle to start goroutine and stop it

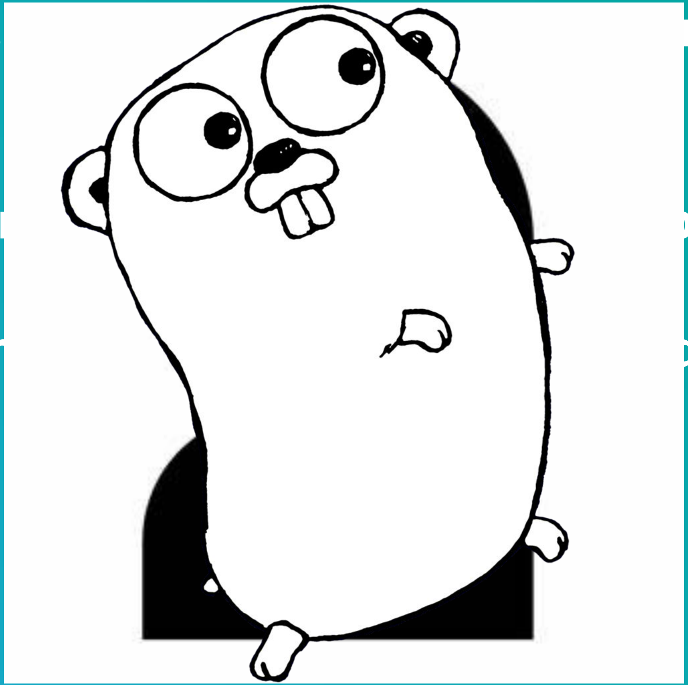
- We can only handle the synchronization point
    - sync.WaitGroup sync.Mutex sync.Cond etc

What’s Go scheduler

- Programmer ca
- ne and stop it
- We can only ha
- oint
  - sync.WaitGr
  - d etc

What’s Go scheduler

- Programmer ca ne and stop it
- We can only ha oint
  - sync.WaitGr d etc



What’s Go scheduler

- Go runtime represents goroutine, machine and processor as G, M, P
  - G represents goroutine. G has information to execute func
  - M represents machine. M has information of OS
  - P represents processor. P has information required to execute Go program

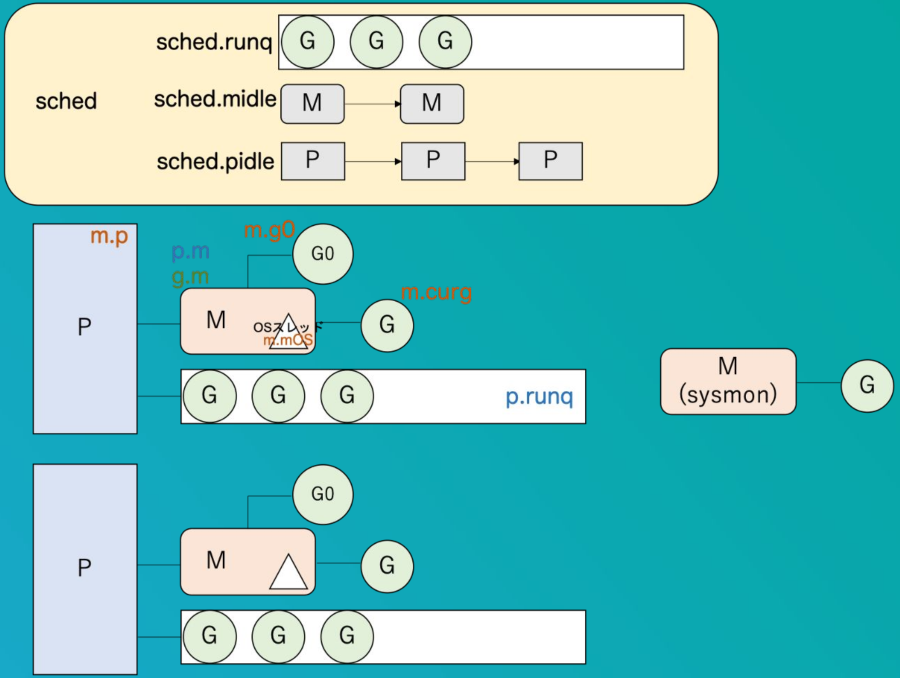
What’s Go scheduler

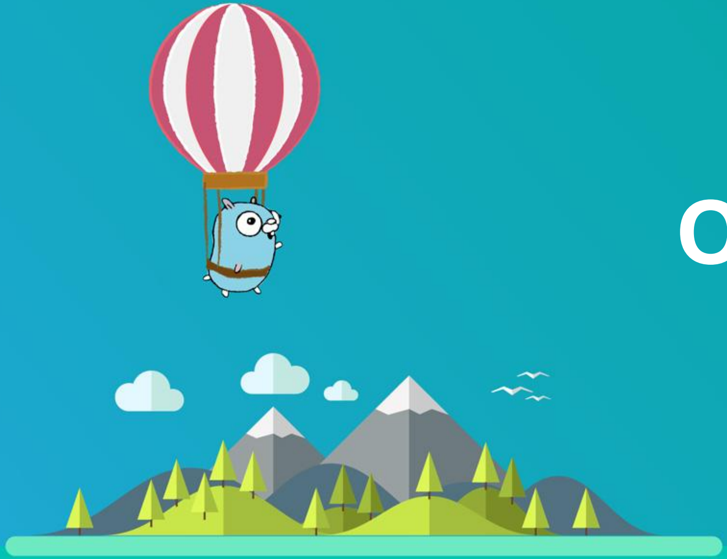
- Go scheduler manages G, M and P
    - Preemption
    - Links G, M and P
    - Push G to Queue and Pop G from Queue
- This model is called M:N model

What’s Go scheduler

- **Thread model**
  - **1:1**
    - **1 user thread links 1 kernel thread**
  - **N:1**
    - **N user thread links 1 kernel thread**
  - **M:N**
    - **M user thread links N kernel thread**

What’s Go scheduler





Overlay Go codes

Overlay Go codes

- Go’s standard library doesn’t export runtime information
  - Almost runtime package doesn’t exported
  - Other packages doesn’t return runtime information

Overlay Go codes

- The way I can get runtime information
    - Modify source code and point GOROOT to changed it
    - Use runtime/trace
    - Use scheddetail=1

Overlay Go codes

- **scheddetail=1**

```
GOMAXPROCS=2 GODEBUG=schedtrace=1000,scheddetail=1  go run main.go

SCHED 0ms: gomaxprocs=2 idleprocs=0 threads=4 spinningthreads=1 needspinning=1 idlethreads=0 runqueue=0
gcwaiting=false nmidlelocked=0 stopwait=0 sysmonwait=false
  P0: status=0 schedtick=0 syscalltick=0 m=nil runqsize=0 gfreecnt=0 timerslen=0
  P1: status=1 schedtick=2 syscalltick=0 m=2 runqsize=0 gfreecnt=0 timerslen=0
  M3: p=0 curg=nil mallocing=0 throwing=0 preemptoff= locks=1 dying=0 spinning=false blocked=false lockedg=nil
  M2: p=1 curg=3 mallocing=0 throwing=0 preemptoff= locks=4 dying=0 spinning=false blocked=false lockedg=nil
  M1: p=nil curg=nil mallocing=0 throwing=0 preemptoff= locks=2 dying=0 spinning=false blocked=false lockedg=nil
  M0: p=nil curg=nil mallocing=0 throwing=0 preemptoff= locks=1 dying=0 spinning=false blocked=false lockedg=1
  G1: status=1(chan receive) m=nil lockedm=0
  G2: status=4(force gc (idle)) m=nil lockedm=nil
  G3: status=2() m=2 lockedm=nil
  G4: status=4(GC scavenge wait) m=nil lockedm=nil
```

Overlay Go codes

- schedetail=1
  - The definition is in proc.go
  - Lock sched, then print the G, M, P information
  - Biggest tips to implement gosched-simulator
  - I want all structs, not printing

Overlay Go codes

- How
  - I don’t want to modify source codes directly
    - Difficult
    - Maintainability
    - Forward compatibility

Overlay Go codes

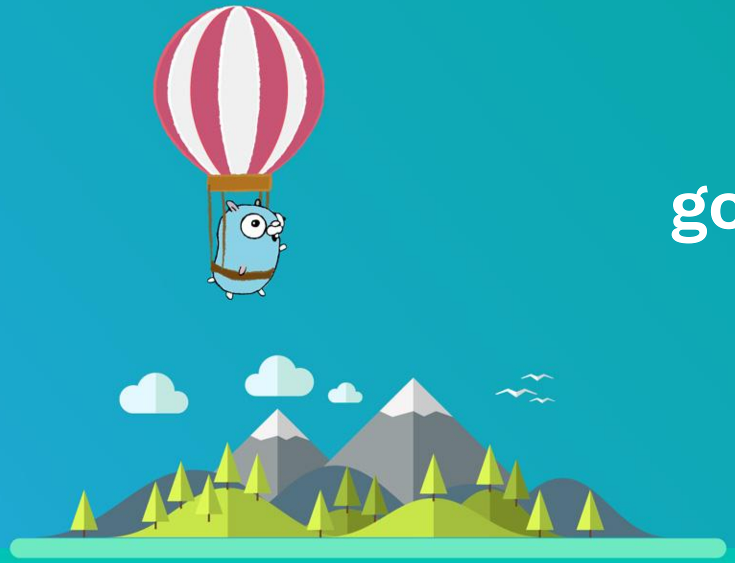
- overlay
    - go help overlay
        - Replace the source codes when building it
        - The setting is managed by JSON file
        - Go standard feature (little bit of a hack 😅)

Overlay Go codes

- overlay

```
{
  "Replace":{
    "${GOROOT}/src/runtime/proc.go":"./local/modifled_proc.go"
  }
}
```

```
go build -overlay overlay.json ./...
```



“

goverlay generates overlay codes

”

goverlay generates overlay codes

- overlay is so hard
  - Copy whole file
  - Update overlayed file if origin file is updated and the changes is important to run Go code
  - Largest sources that I don’t want to manage

goverlay generates overlay codes

- Thus, I want to
    - add the some definition that I want to use
    - patch the partial codes like a kustomization
    - control using AST

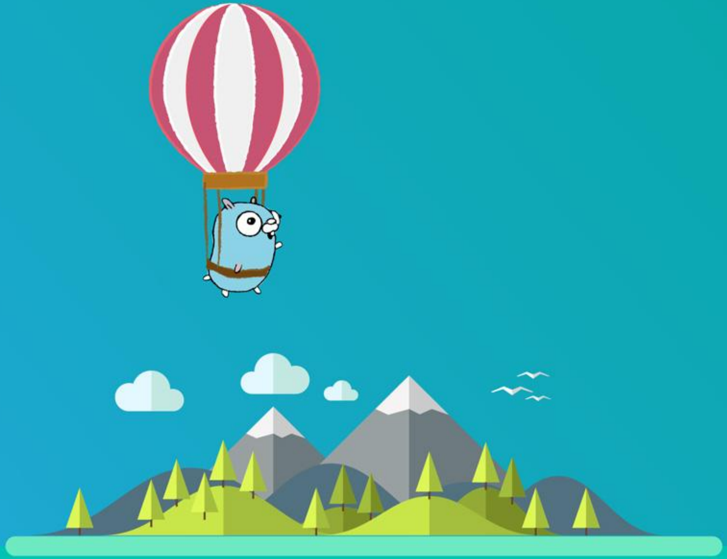
goverlay generates overlay codes

- goverlay

```
layers:
  - from: ./testdata/from.go
    patch: ./testdata/patch/patch.go
    dist: ./testdata/dist.go
    replaces:
      - kind: func
        ident: A
      - kind: struct
        ident: b
```

goverlay generates overlay codes

- goverlay can
  - add the new definition
  - patch the existing definition and replace it
    - type typ struct -> type _typ struct
    - func fn -> func _fn
  - generate overlay.json from goverlay.yaml configuration

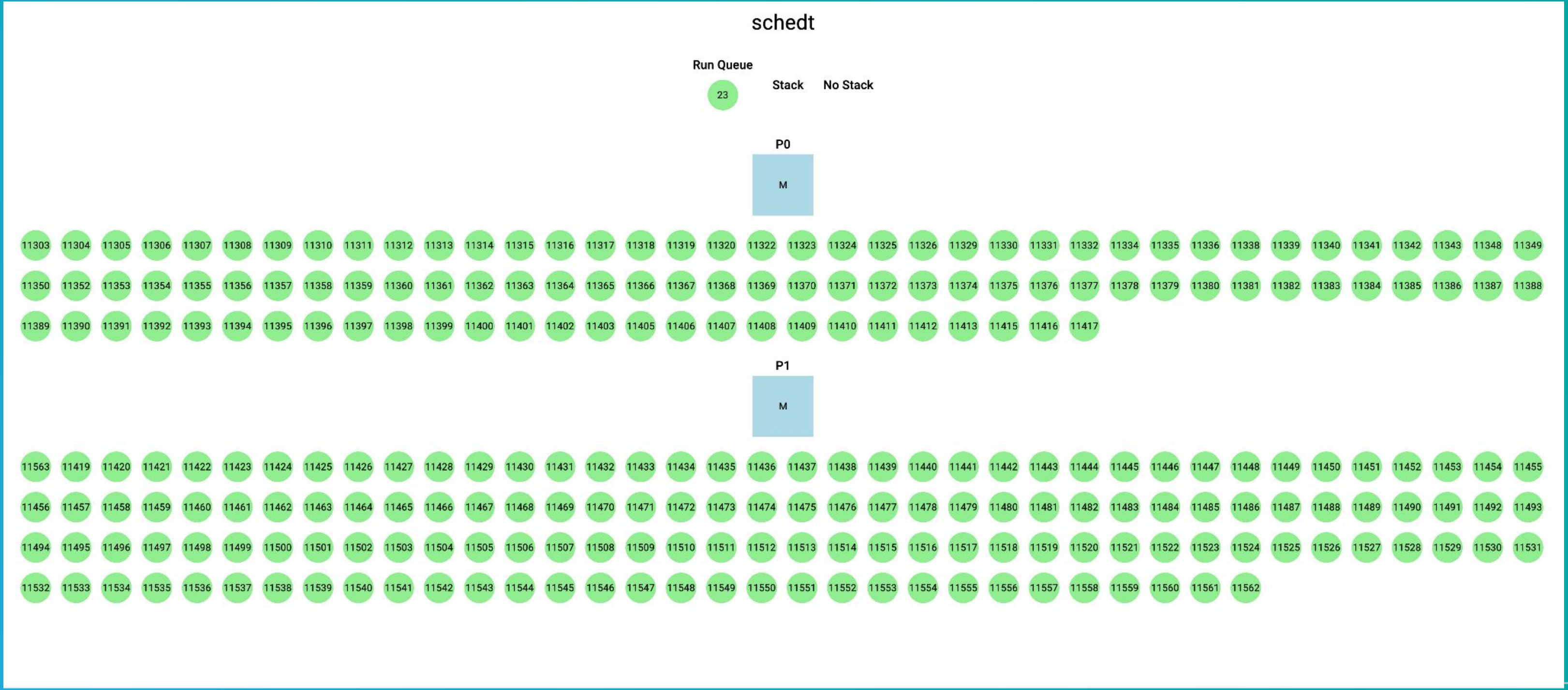


“

gosched simulator

”

gosched simulator

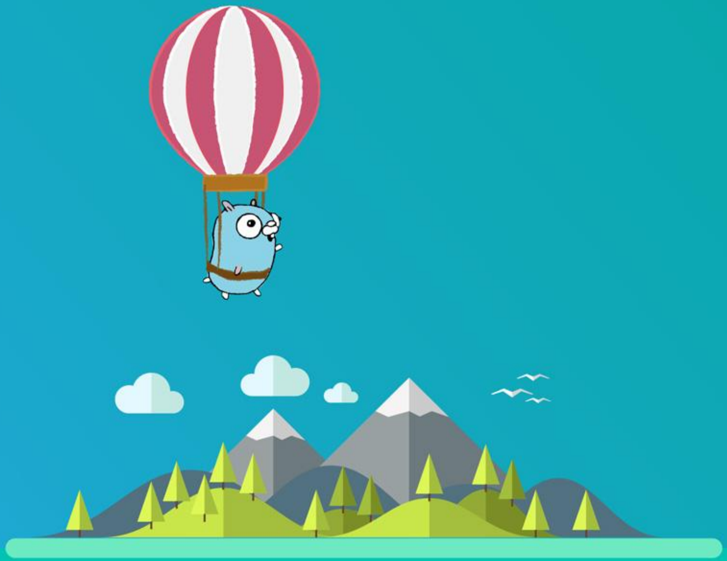


goverlay generates overlay codes

- gosched simulator
    - Visualize G, M, P and Scheduler
        - WaitReason, Status etc
    - Realtime simulation
    - Handle all G, M, P, Scheduler resources

goverlay generates overlay codes

- Expose type and convert from private type to public type
- Follow the scheddetail implementation
- No LSP, so I use acme editor or vanilla Vim
  - Of course, go tool doesn’t work 🧐



Summary

Summary

- Go scheduler is amazing, but it’s hard to visualize by right way
- overlay is magic, goverlay is magic wand for me
- Plan
    - Visualize more information
    - Improve maintainability

Please give me your feedback

Reference

- Node.js event loop
- Goでの並行処理を徹底解剖！