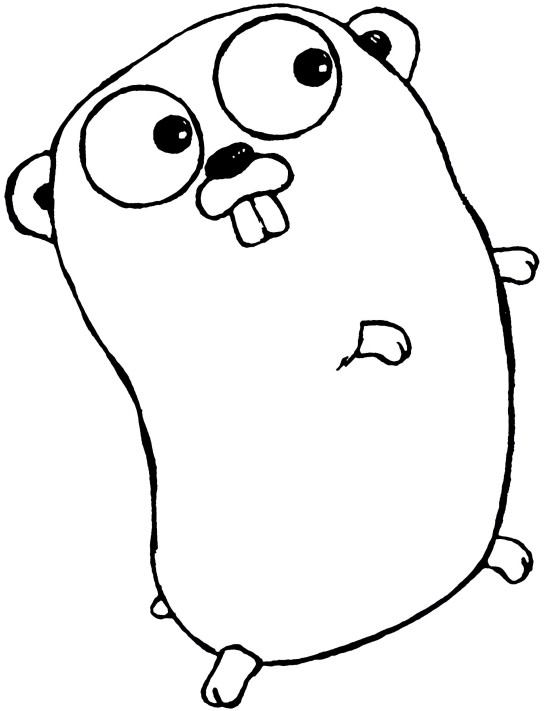


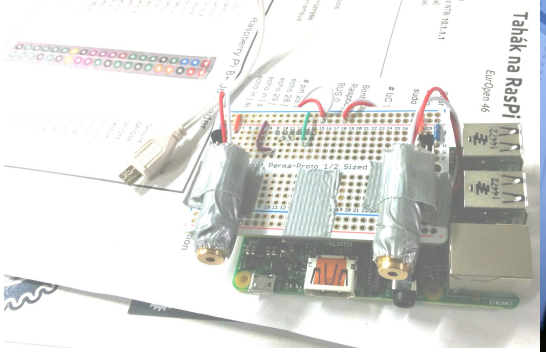
Why you want to learn Go

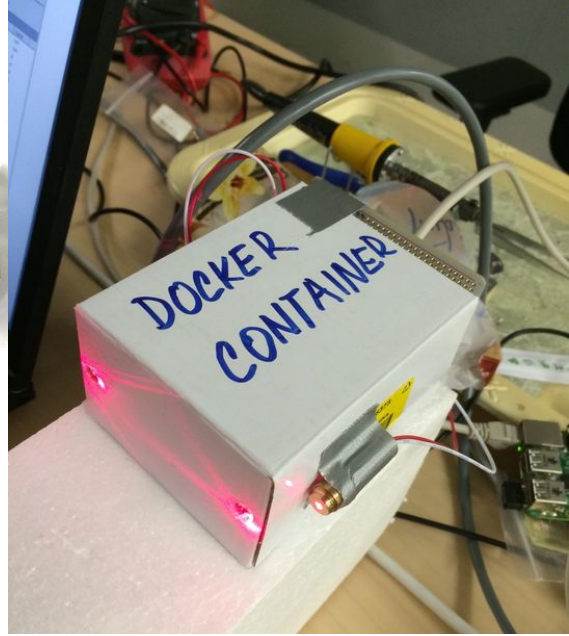
Viruslab Systems

Jan Seidl

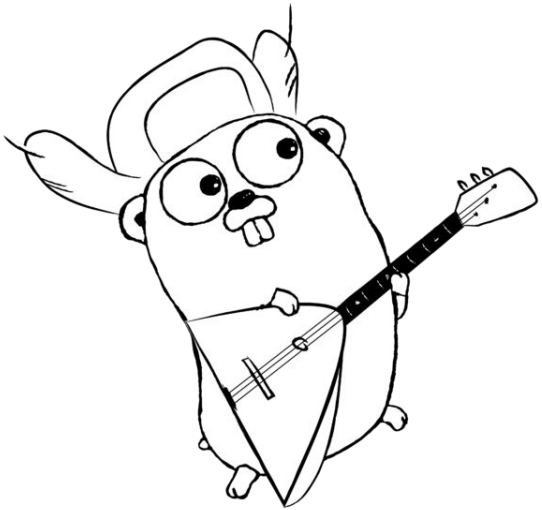
This slides link: https://goo.gl/nhXTP4

Golang and Me



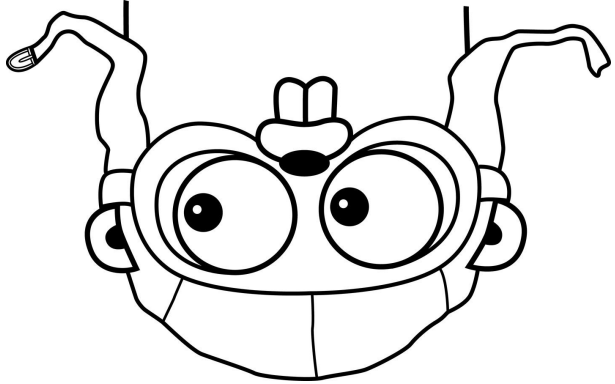


- Parkathon
- check_docker_OOMkiller, docker_container_size_check, monitor-marathon-to-statsd
- stor-client
- hashutil-go
- retry-go
- pathutil-go



- dist-go

Golang history



- Google (2007)
  - Robert Griesemer
  - Rob Pike (Unix, Plan9 from Bell Labs, Limbo, sam, acme, UTF-8,...)
  - Ken Thompson (Unix, Plan9 from Bell Labs, B, UTF-8,...)
- Go 1.0 (2012)
- Go 1.5 - go compiler and runtime fully implemented in Go (28 January 2016)
- Go 1.11 now (2 months ago) - modules

Golang history in TIOBE

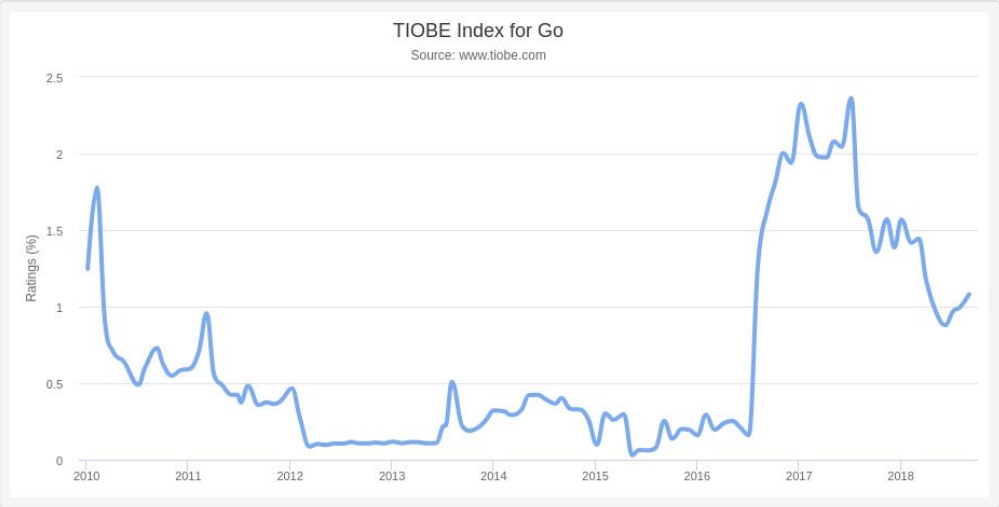
The Go Programming Language

Some information about Go:

Highest Position (since 2001): #10 in Jul 2017

Lowest Position (since 2001): #122 in May 2015

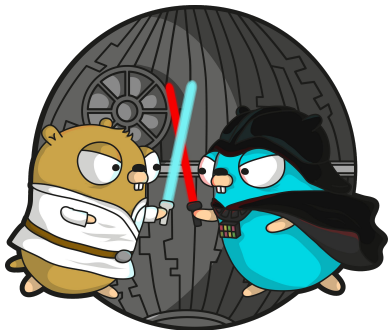
Language of the Year: 2009, 2016



https://www.tiobe.com/tiobe-index/go/

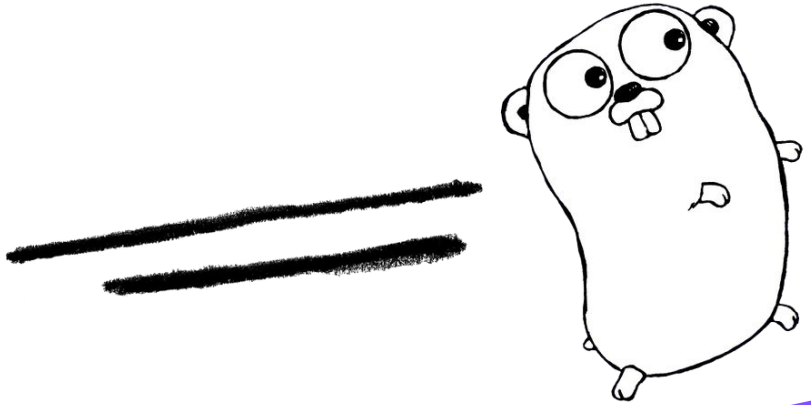
https://blog.risingstack.com/the-history-of-kubernetes/

Wait!
Go or Golang?

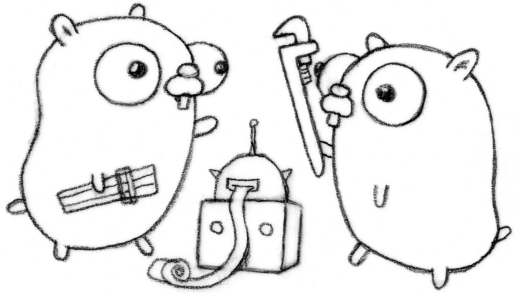


Why did google make a new language?

- scalable for large systems (java/C++)
- dislike C++ complexity
- readable (not too many keywords)
- good concurrency support
- fast

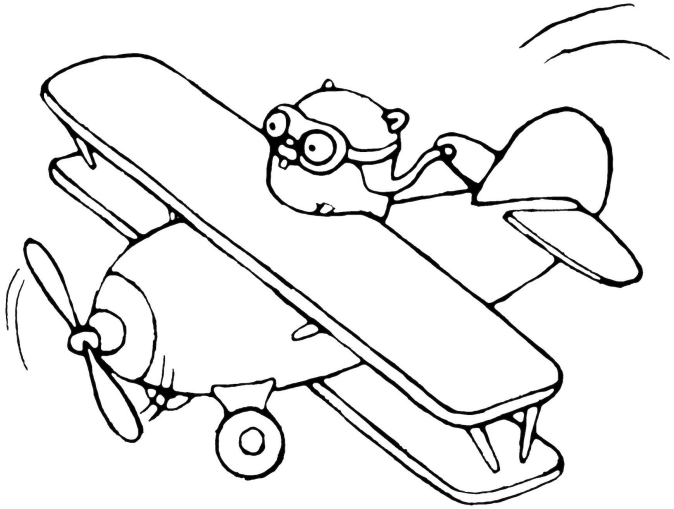


Golang features



- garbage collector
- strong typing
- memory safety (defaults)
- goroutines and channels
- type with methods (aka limited OOP)
- statically compiled
- tooling
- libraries
- cgo
- easy to read
- easy to learn
- fast compilation times
- cross compilation support
- strict formatting (go fmt)
- go workspace
- community
- no semicolons
- godoc.org
- ...

Missing in Golang

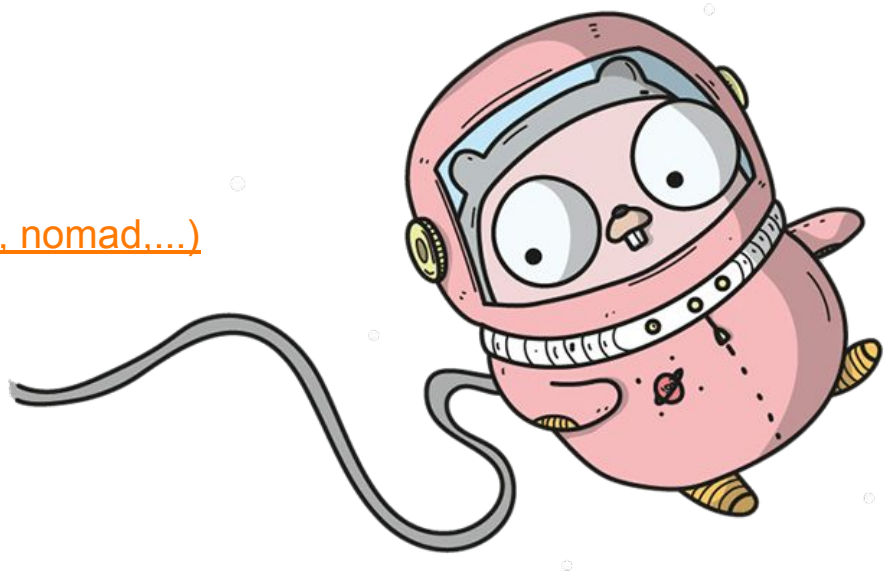


- user-defined generics
- macros
- OOP (constructors, polymorphism, inheritance,...)
- exceptions
- pointer arithmetic
- overloading
- default arguments
- functional programming (reduce, map, immutable,...)

Community

Popular projects written in golang

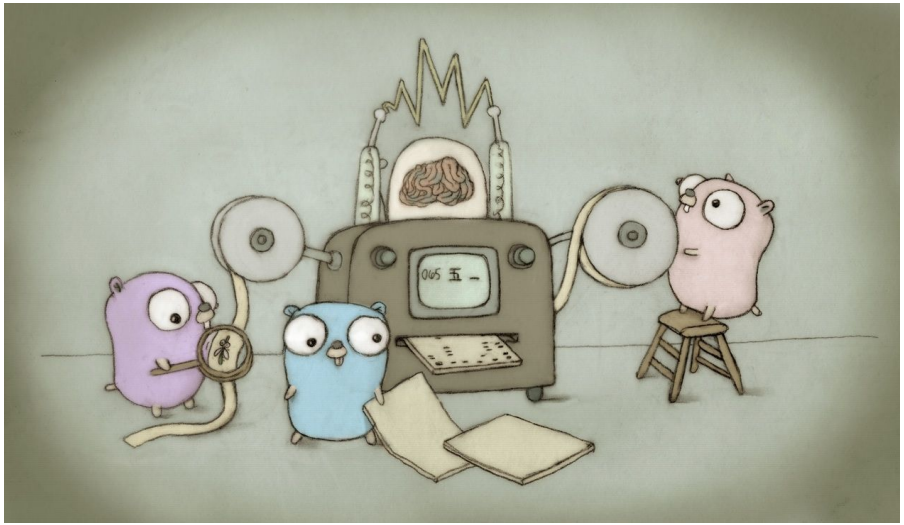
- Moby (docker)
- Kubernetes
- Prometheus
- Grafana
- HashiCorp tools (consul, vault, nomad,...)
- Etcd
- CockroachDB
- NSQ
- LimeText
- Syncthing
- Traefik
- Minio
- Influxdb
- ...



avast

Companies using golang

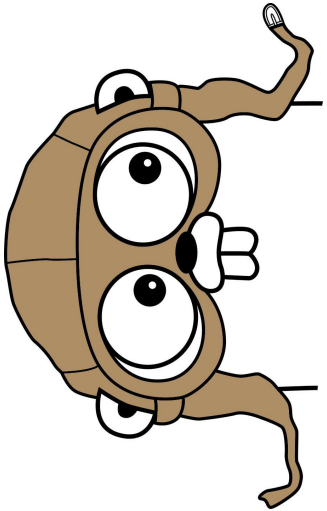
- Google
- Microsoft
- Alibaba
- Ethereum
- Mozilla
- Github
- Uber
- Netflix
- SoundCloud
- CloudFlare
- ...

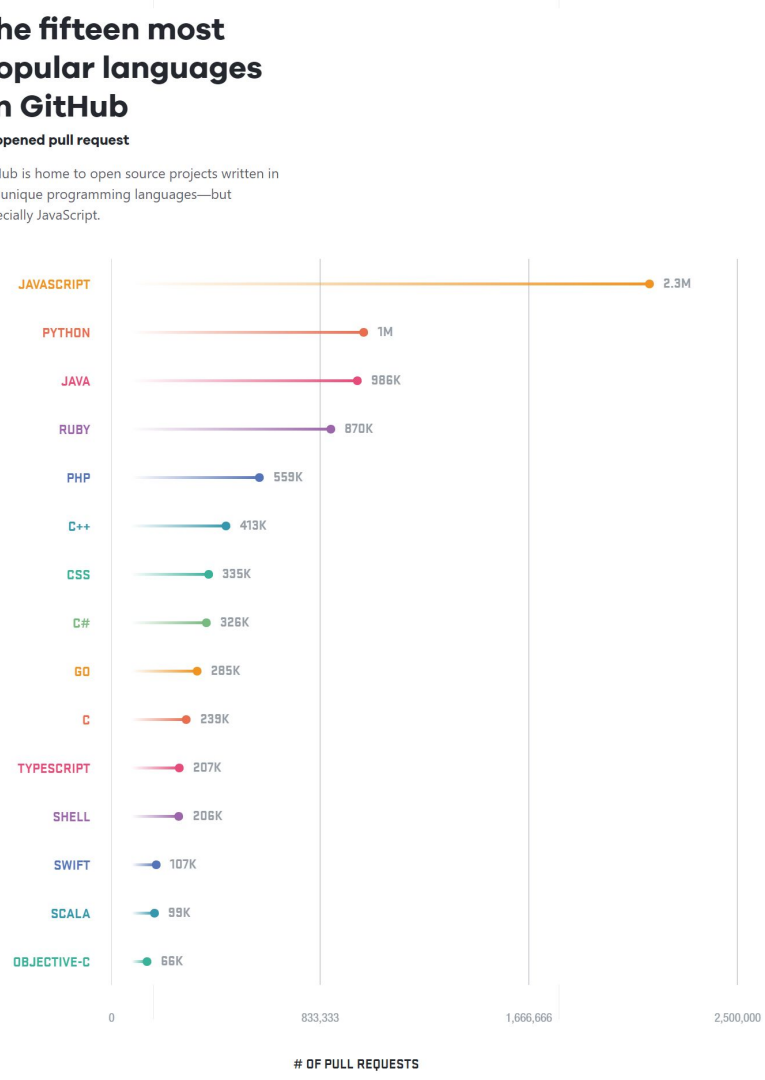


https://github.com/trending/developers/go?since=monthly

avast

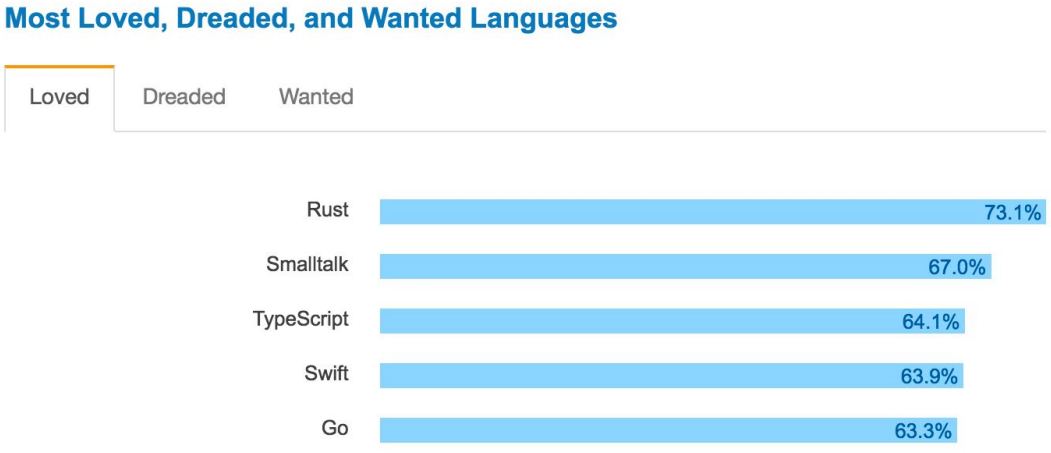
Developers love Go

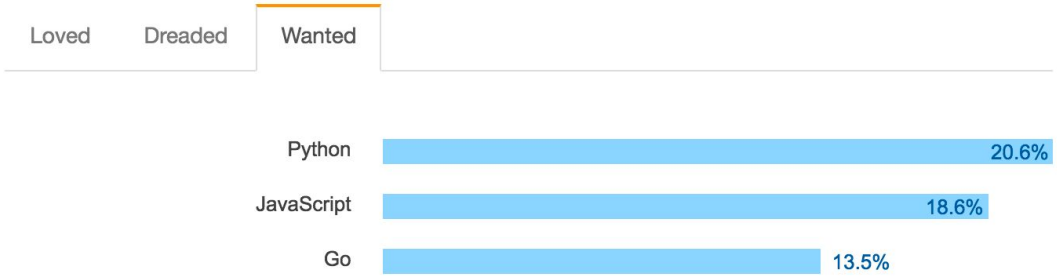




https://blog.golang.org/8years

Stack Overflow's 2017 developer survey





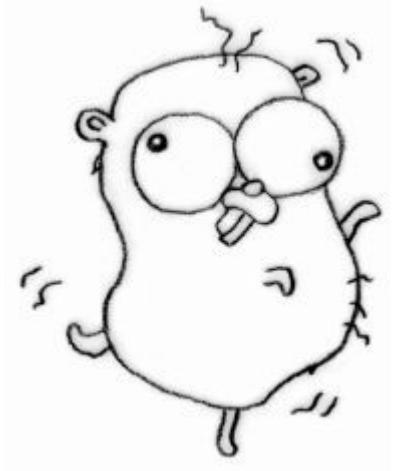
https://blog.golang.org/8years

avast

What I hate about Go

Package management (< Go 1.11)

- tool `go get`
- supports git/svn/mercurial/bazaar
- **only master**



```
package main

import (
  "github.com/avast/retry-go"
  log "github.com/sirupsen/logrus"
)

func main() {

}
```

Go workspace

$GOPATH

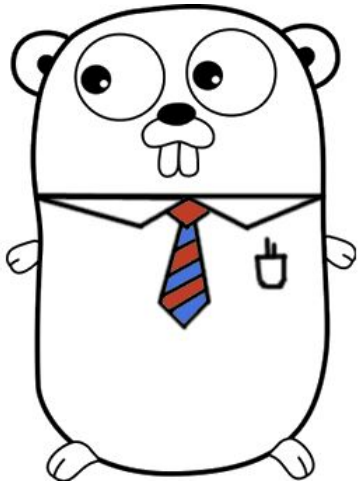
```
bin/
pkg/
src/
   github.com/golang/example/
     .git/                      # Git repository metadata
     hello/
       hello.go                 # command source
     outyet/
       main.go                  # command source
       main_test.go             # test source
   golang.org/x/image/
     .git/                      # Git repository metadata
     bmp/
       reader.go                # package source
       writer.go                # package source


    ... (many more repositories and packages omitted) ...
```

avast

Vendor

- go 1.5+
- vendor tree example

```
├── main.go
└── vendor
    ├── github.com
    │   ├── andybalholm
    │   │   └── cascadia
    │   │       ├── LICENSE
    │   │       ├── parser.go
    │   │       ├── README.md
    │   │       └── selector.go
```



https://blog.gopheracademy.com/advent-2015/vendor-folder/

avast

Package management solution

- dep
- manul
- Godep
- Govendor
- godm
- vexp
- gv
- gvt
- govend
- Glide
- Vendetta
- trash
- gsv
- gom
- Rubigo
- ....

https://github.com/golang/go/wiki/PackageManagementTools

Package management solution: dep

- `dep init`
- Gopkg.toml
- `dep ensure`
- community (~~yet~~)

```
[[constraint]]
 name = "github.com/pkg/errors"
 version = "0.8.0"

[[constraint]]
 name = "github.com/boltdb/bolt"
 version = "1.0.0"

[[constraint]]
 name = "github.com/jmank88/nuts"
 version = "0.3.0"

[[constraint]]
 name = "github.com/golang/protobuf"
 branch = "master"
```

https://github.com/golang/dep

Package management solution: vgo / Go1.11 modules

New official dependency tool?

> Go 1.11 is not shipping with "vgo support". It is shipping with "Go module support". Vgo was only ever the prototype/demo of that module support. Just like all the other features we've introduced to the go command over the past ten years, the goal for this one is to fit so well into the rest of the go command fabric that the seams are invisible - module support is just part of "go", the same way that, say, race detector support or GOPATH support is.
>
> https://groups.google.com/forum/#!topic/golang-dev/a5PqQuBljF4

- https://blog.spiralscout.com/golang-vgo-dependency-management-explained-419d143204e4
- https://golang.org/cmd/go/#hdr-Modules__module_versions__and_more
- https://godoc.org/golang.org/x/vgo

Package management solution: Go1.11 modules

~/workspace/example/go.mod:

```
module github.com/JaSei/ModuleExample

require github.com/avast/retry-go 1.0.2
```


```
go get

go: finding github.com/avast/retry-go 1.0.2
go: downloading github.com/avast/retry-go v0.0.0-20180502193734-611bd93c6d74
# github.com/JaSei/ModuleExample
```

~/workspace/example/go.mod:

```
module github.com/JaSei/ModuleExample

require github.com/avast/retry-go v0.0.0-20171124144729-5e4735226698
```

avast

Defaults

- no default arguments
- no constructor sugar

```
type defaults struct {
    a string    // ""
    b int       // 0
    c float64   // 0
    d bool      // false
    e *int      // nil
}
```

How to set attributes? (python)

```
class Client:
     def __init__(addr: str = 'localhost', port: int = 8080):
         self.addr = addr
         self.port = port
```

```
Client()
# or
Client(addr = '10.0.0.1', port = 12345)
```

How to set attributes? - example I

```go
package client

type Options struct {
    Addr string
    Port uint16
}

func New(opt Options) newObject {
    newObject := someObject{}
    if opt.Addr == "" {
        newObject.Addr = "localhost"
    }
    if opt.Port == 0 {
        newObject.Port = 8080
    }
}
```

```go
func main() {
    client.New(Options{})
    // or
    client.New(Options{Addr: "10.0.0.1", Port: 12345})
}
```

How to set attributes? - example II

```go
package client

type config struct {
    addr string
    port  uint16
}

type Option func(*config)

func Addr(addr string) Option {
    return func(c *config) { c.addr = addr }
}

func Port(port uint16) Option {
    return func(c *config) { c.port = port }
}
```

```go
func New(opts ...Option) config {
    config := &config{
        addr: "localhost",
        port:  8080,
    }

    for _, opt := range opts {
        opt(config)
    }
}
```

```go
func main() {
    client.New()
    // or
    client.New(client.Port(12345),
client.Addr("10.0.0.1"))
}
```

https://github.com/avast/retry-go

avast

Error handling

- no exceptions
- standard `error` interface
- tuples

```
type error interface {
    Error() string
}
```

```
func DoSomeOther() error {
    some, err := doA()
    if err != nil {
        return err
    }

    if err := doB(); err != nil {
        return err
    }
    return nil
}

func doA() (int, error) {
...
}

func doB() error {
...
}
```

Nil is not nil

```go
if err != nil {
   log.Println(err.Error())
}
```

```
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0xffffffff addr=0x0
pc=0xdabc6]

goroutine 1 [running]:
main.(*myError).Error(0x0, 0x1, 0x1, 0x7766)
      /tmp/sandbox001469156/main.go:11 +0x6
main.main()
      /tmp/sandbox001469156/main.go:27 +0x100
```

Nil is not nil

```
type myError string

func (e *myError) Error() string {
      return (string)(*e)
}

func main() {
      var a *myError
      fmt.Println(a == nil) //true

      var t error
      fmt.Println(t == nil) //true

      t = a
      fmt.Println(t == nil) //false

      if t != nil {
            log.Println(t.Error())
      }
}
```

https://play.golang.org/p/CiIaGrqAbPB

https://golang.org/doc/faq#nil_error

https://speakerdeck.com/campoy/understanding-nil

```
*struct { typ type; impl *ptr_t }
```

Error (nil, nil) => nil
Error (*myError, nil) => not nil

No generics

- isn't possible to do
  - own hash implementation (new interface only)
  - filter/map/reduce
  - ...

What I hate about Go

- ~~package management~~
- ~~defaults~~
- error handling
- nil is not nil
- no generics

Go 2 Draft Designs

- error handling
- error values
- generics

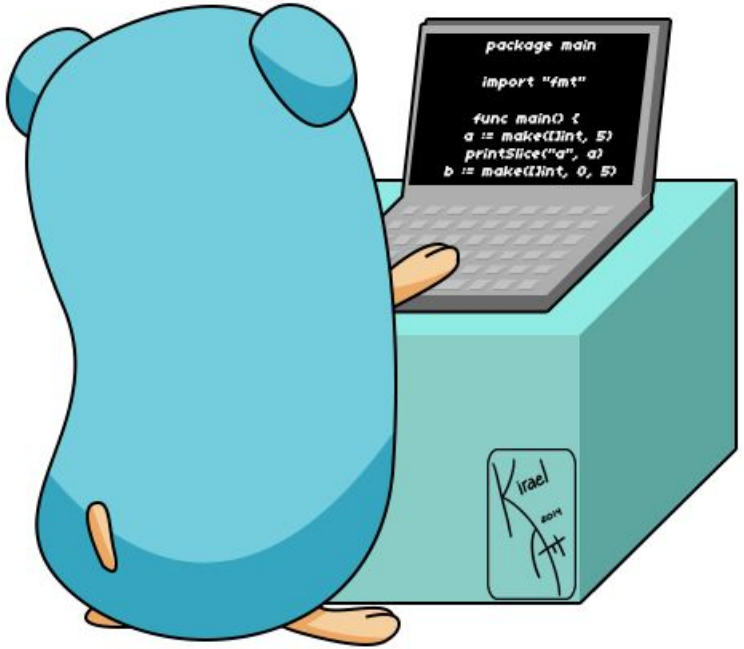
https://go.googlesource.com/proposal/+/master/design/go2draft.md

avast

What I love about Go

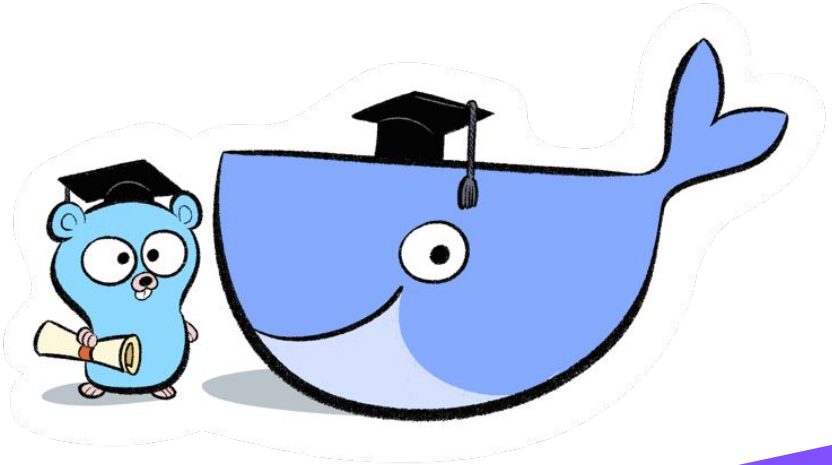
Golang is easy

- read
- (write)
- code review
- good for open source



Statically linked

- no dependency hell
- works on all target platforms
- docker is small (`FROM scratch, CGO_ENABLED=0`)
- binaries are big
  - -ldflags="-s -w"



Concurrency

Concurrency is a way to structure a program by breaking it into pieces that can be executed independently.

Communication is the means to coordinate the independent executions.

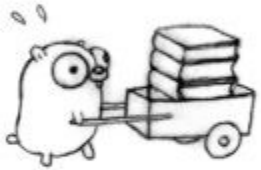
This is the Go model and (like Erlang and others) it's based on CSP:

C. A. R. Hoare: Communicating Sequential Processes (CACM 1978)

Rob Pike

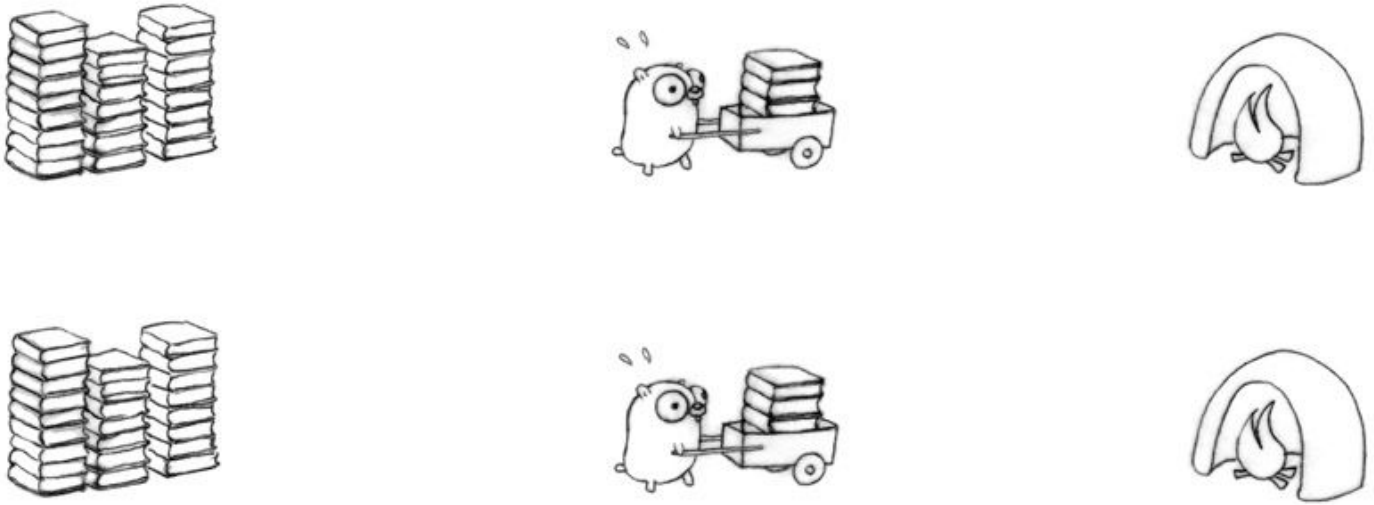
https://blog.golang.org/concurrency-is-not-parallelism

Concurrency



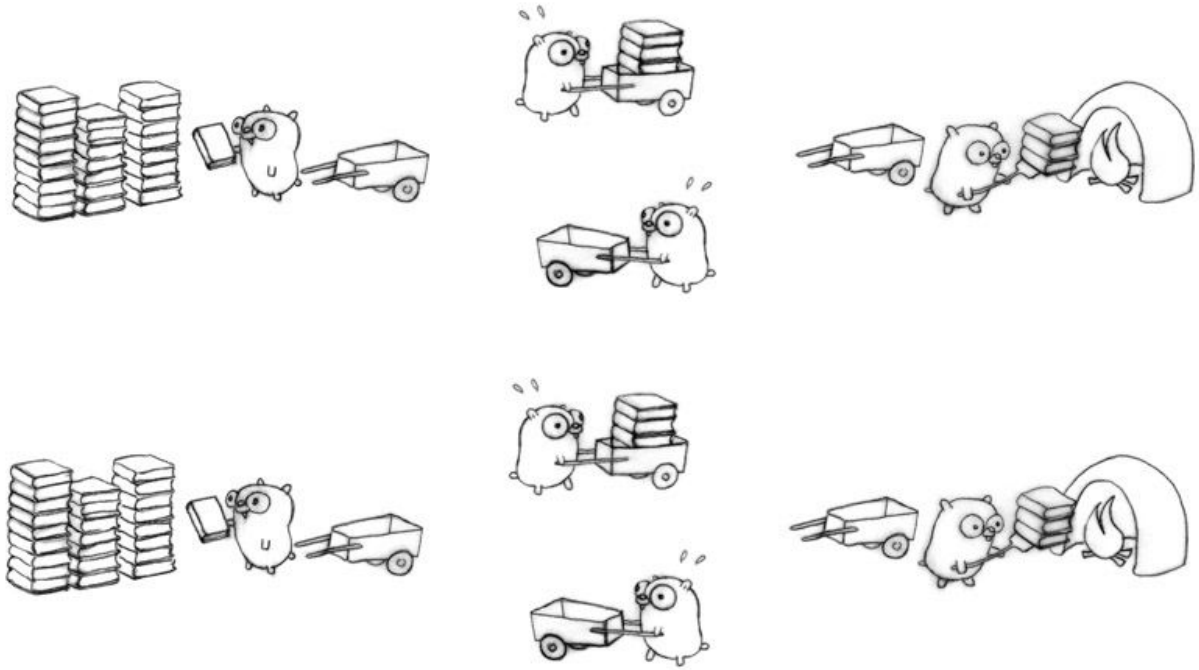
https://talks.golang.org/2012/waza.slide#12

Concurrency



https://talks.golang.org/2012/waza.slide#15

Concurrency



https://talks.golang.org/2012/waza.slide#23

Ping-pong

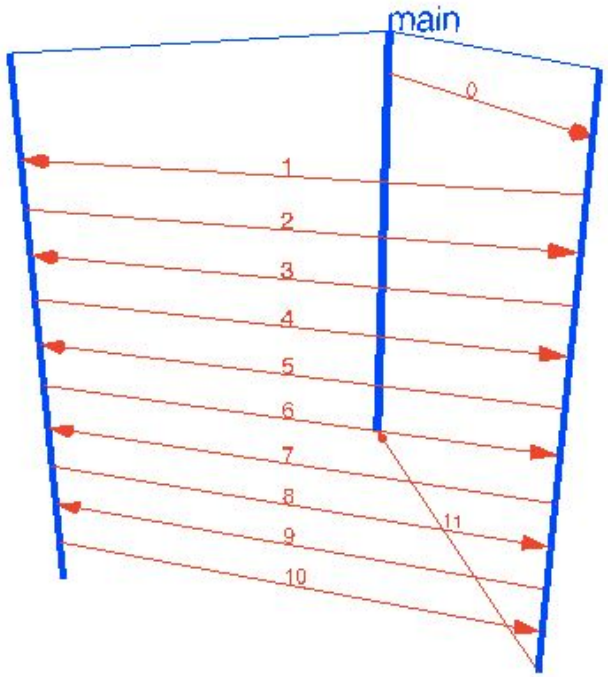
```go
package main

import (
    "fmt"
    "time"
)

func main() {
    table := make(chan Ball)
    go player("ping", table)
    go player("pong", table)

    table <- Ball{} // Game on; Toss the ball
    time.Sleep(1 * time.Second)
    <-table // game over; grab the ball
}
```

```go
type Ball struct {
    hits int
}

func player(name string, table chan Ball) {
    for {
        ball := <-table
        ball.hits++
        fmt.Println(name, ball.hits)
        time.Sleep(100 * time.Millisecond)
        table <- ball
    }
}
```

Ping-pong

```
pong 1
ping 2
pong 3
ping 4
pong 5
ping 6
pong 7
ping 8
pong 9
ping 10
```

Concurrency

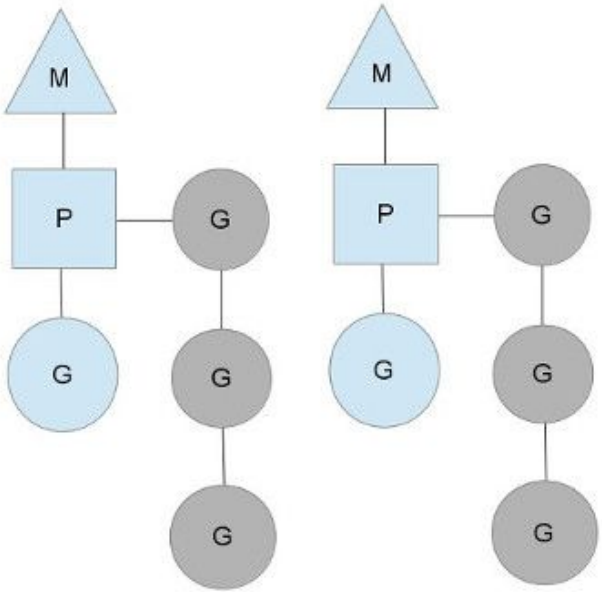


http://divan.github.io/posts/go_concurrency_visualize/

avast

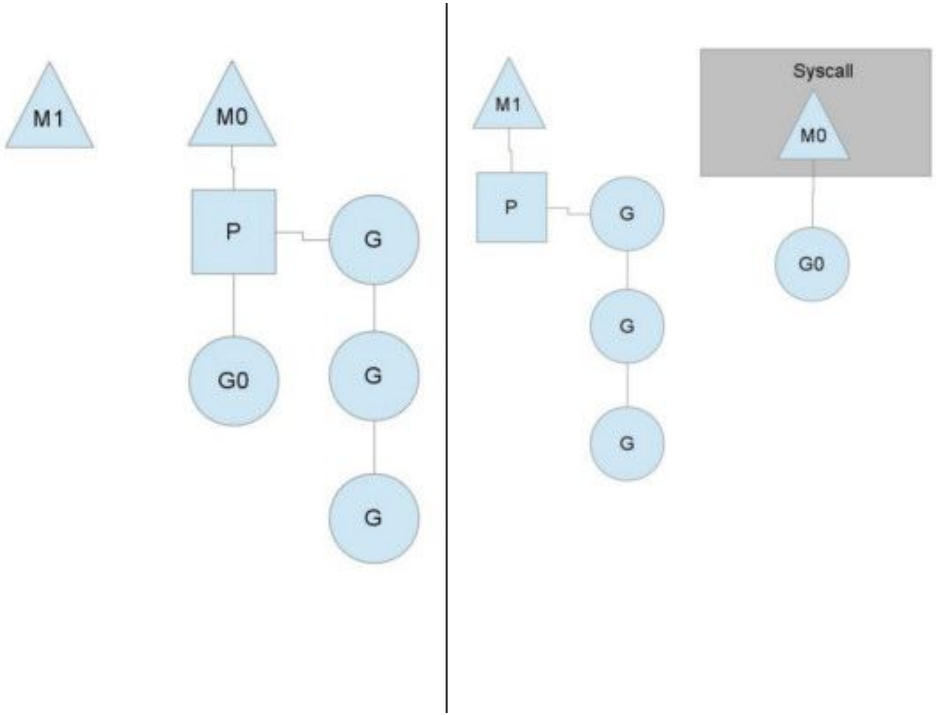
Goroutine

How the scheduler works



https://morsmachine.dk/go-scheduler

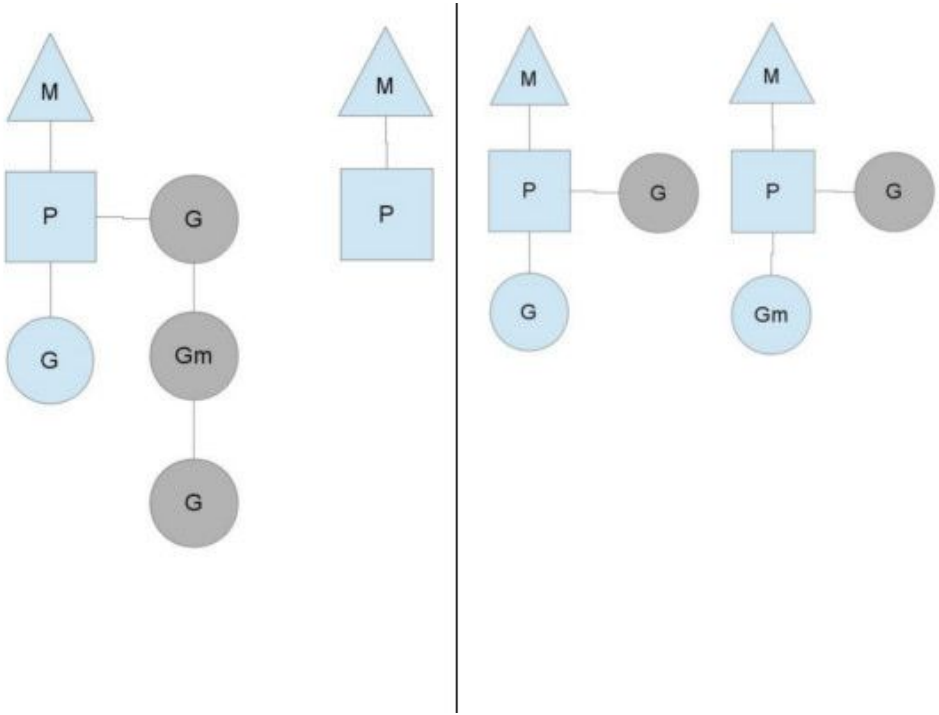
How the scheduler works - syscall



https://morsmachine.dk/go-scheduler

avast

How the scheduler works - stealing



https://morsmachine.dk/go-scheduler

avast

Goroutines yield to scheduler on:

- sleeping
- IO (network, disk operations,...)
- channel operation
- blocking primitives in the sync package
- call syscalls
- runtime.Gosched()

CPU intensive tasks never yield to scheduler!

demo

Concurrency summary

- goroutine != thread
- `GOMAXPROCS`
- work-stealing scheduler
- blocking vs cpu intensive operations

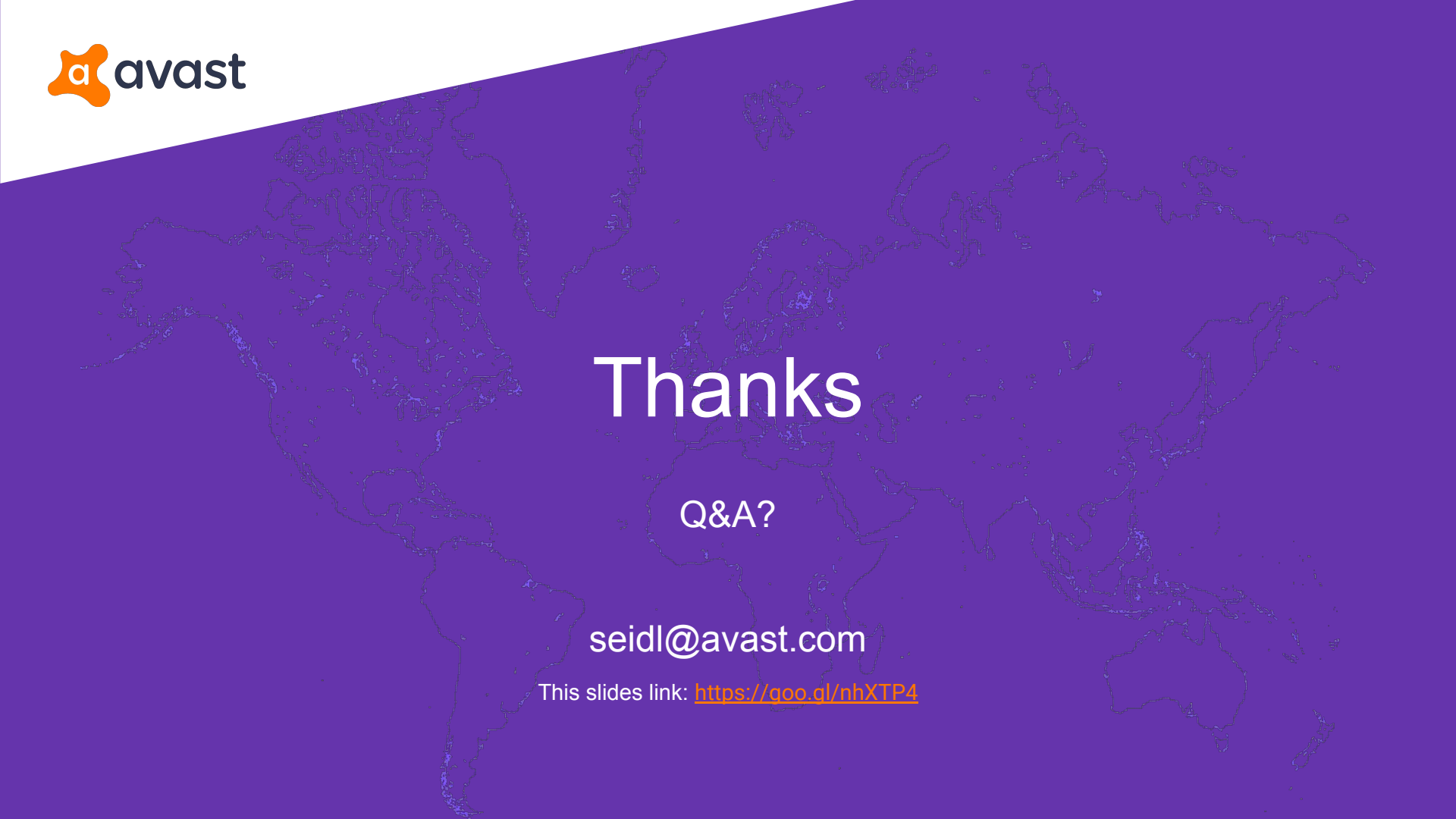
Conclusion

- easy to learn / read / use
- as fast as native code
- good concurrency patterns
- statically compiled
- good for ~~system programming~~ programming of systems, cli, backends,...
  - kubernetes tooling
- open source friendly language
- worst for domain oriented programming

Less is exponentially more

Rob Pike

https://commandcenter.blogspot.cz/2012/06/less-is-exponentially-more.html



Thanks

Q&A?

seidl@avast.com

This slides link: https://goo.gl/nhXTP4