C6 BANK

CAMINHANDO COM GO

Rafael Gottardi, Beatriz Vieira e Sergio Tutumi

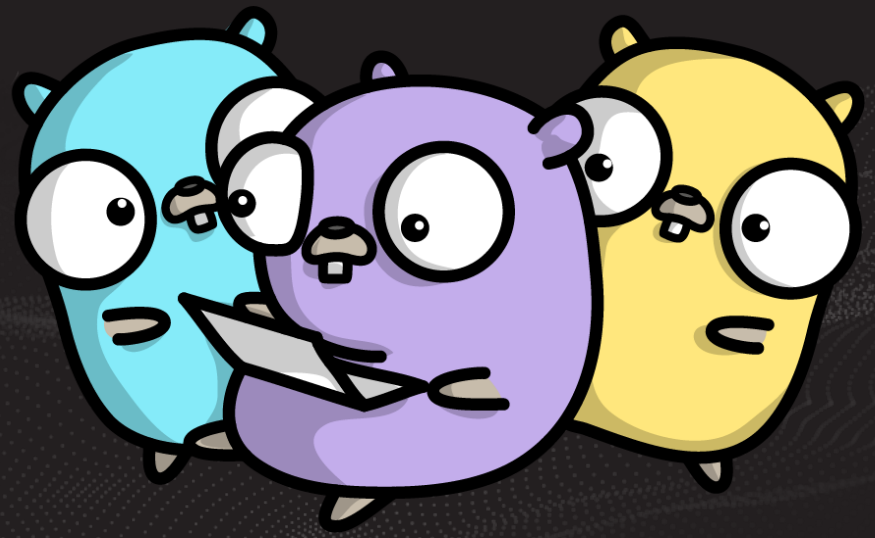
Quem somos?

Agenda

- Go?
- Case C6 Bank

C6BANK

Go?



O que é?

- Criada em 2009
- Compilada
- Estaticamente tipada
- Garbage collector
- Concorrência
- Simplicidade
- Binário estático e autocontido

"Clear is better than clever"

"Design the architecture, name the components, document the details"

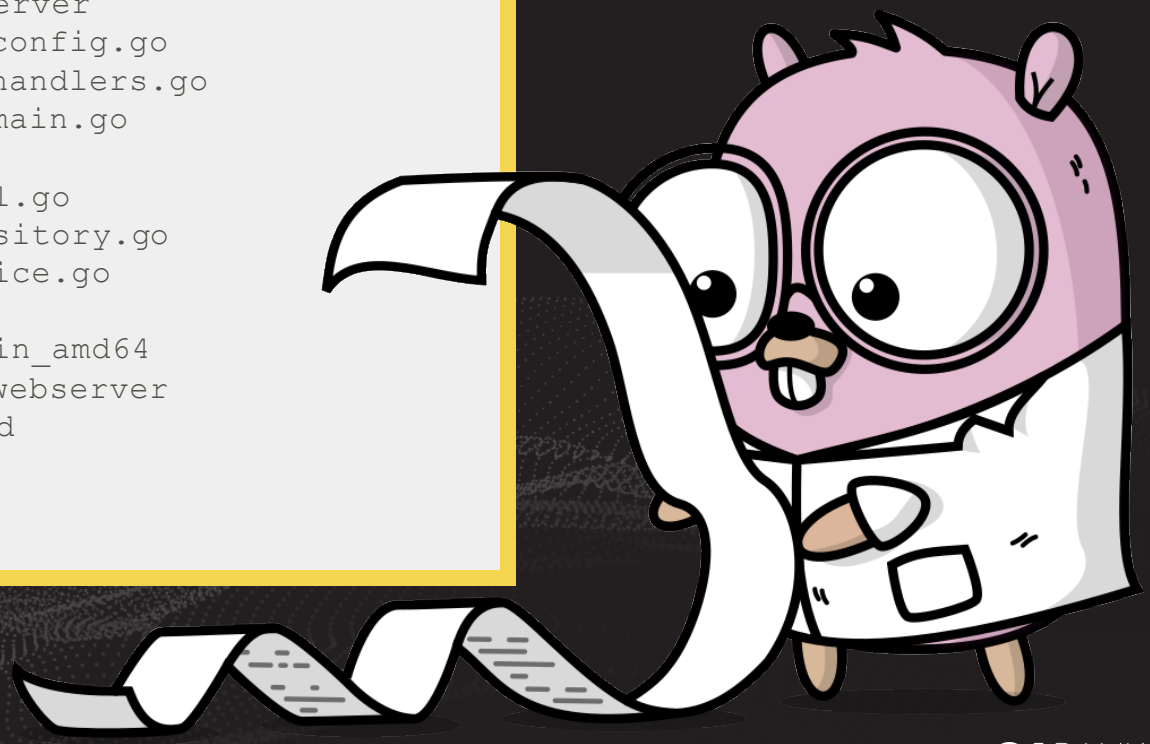
"A little copying is better than a little dependency"

Go Proverbs

Packages

- Estrutura enxuta
- *Library driven development*
- Pacotes especiais
  - vendor
  - internal

```
.
├── cmd
│   └── webserver
│       ├── config.go
│       ├── handlers.go
│       └── main.go
├── api
│   ├── model.go
│   ├── repository.go
│   └── service.go
├── build
│   └── darwin_amd64
│       └── webserver
├── README.md
├── Makefile
├── go.mod
└── go.sum
```

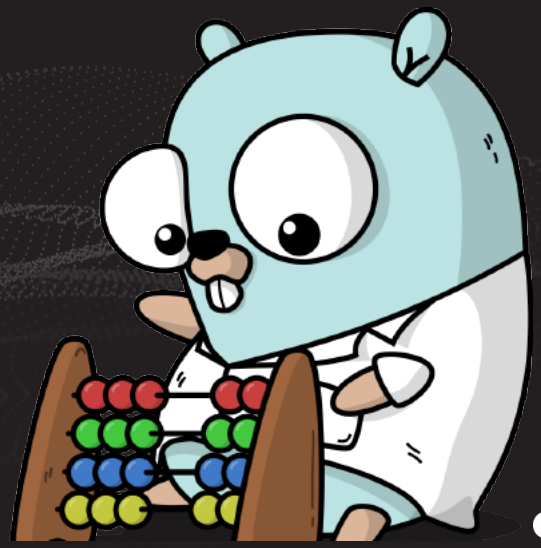


Interfaces

- Define comportamento
- Implementação implícita
- “interface{} says nothing”
- “The bigger the interface, the weaker the abstraction”

```
type Reader interface {
        Read(p []byte) (n int, err
error)
}

type ReadWriter interface {
        Reader
        Writer
}
```



C6BANK

Erros

- é uma interface
- “Don't panic”
- “Errors are values”
- “Don't just check errors, handle the gracefully”
- Draft: Go 2.0

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

```
func someFunc() (Result, error) {
    result, err := repository.Find(id)
    if err != nil {
        log.Errof(err)
        return Result{}, err
    }
    return result, nil
}
```

Concorrência

- Linguagem já foi pensada com suporte a concorrência
- go, channel e select

```
func main() {
    messages := make(chan string)
    go func() { messages <- "ping" }()
    msg := <-messages
    fmt.Println(msg)
}
```

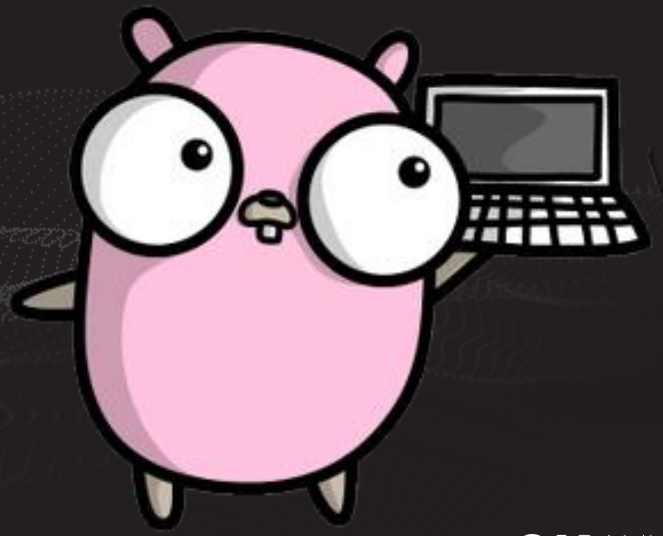
```
select {
case msg1 := <-c1:
    fmt.Println("received", msg1)
case msg2 := <-c2:
    fmt.Println("received", msg2)
}
```

- “Don't communicate by sharing memory, share memory by communicating”
- “Channels orchestrate; mutexes serialize”
- “Concurrency is not parallelism”

C6BANK

Toolchain - Coding

- Formatação - gofmt
  - “Gofmt's style is no one's favorite, yet gofmt is everyone's favorite.”
- Linter - golint
  - Segue o padrão do Effective Go e CodeReviewComments
- Vet - go vet
  - Conjunto de heurísticas para apontar erros comuns



Toolchain – Testing

- Testing package built-in
- Testes ficam no mesmo package do que está sendo testado
- Permite testes white-box
- Arquivos _*test* excluídos do build

```
.
├── cmd
│   └── webserver
│       ├── main.go
│       └── main_test.go
├──  api
│    ├── service.go
│    └── service_test.go
```

Toolchain – Testing

- Table Driven Tests: escreva a lógica do teste uma vez

```
func TestSplit(t *testing.T) {
  tests := []struct {
    name string
    input string
    sep string
    want []string
  }{
    {name: "no sep", input: "abc", sep: "/", want: []string{"abc"}},
    {name: "trailing sep", input: "a/b/c/", sep: "/", want: []string{"a", "b", "c"}},
  }

  for _, tc := range tests {
    got := Split(tc.input, tc.sep)
    If !reflect.DeepEqual(tc.want, got) {
      t.Fatalf("%s: expected: %v, got: %v", tc.name, tc.want, got)
    }
  }
}
```

```
go test
--- FAIL: TestSplit (0.00s)
    split_test.go:25: trailing sep: expected: [a b c], got: [a b c ]
```

- Dica: gotests

Toolchain – Testing

Benchmark

```
func BenchmarkFib10(b *testing.B) {
  for n := 0; n < b.N; n++ {
    Fib(10)
  }
}
```

```
go test -bench=.
PASS
BenchmarkFib10        5000000                509 ns/op
Ok         github.com/davecheney/fib        3.084s
```

Coverage

```
go test -coverprofile=coverage.out
PASS
coverage: 42.9% of statements
ok      size    0.030s
```

Race

```
WARNING: DATA RACE
Read by goroutine 185:
  net.(*pollServer).AddFD()
      src/net/fd_unix.go:89 +0x398
  net.(*pollServer).WaitWrite()
      src/net/fd_unix.go:247 +0x45
  net.(*netFD).Write()
      src/net/fd_unix.go:540 +0x4d4
  net.(*conn).Write()
      src/net/net.go:129 +0x101
  net.func·060()
      src/net/timeout_test.go:603 +0xaf

Previous write by goroutine 184:
  net.setWriteDeadline()
      src/net/sockopt_posix.go:135 +0xdf
  net.setDeadline()
      src/net/sockopt_posix.go:144 +0x9c
  net.(*conn).SetDeadline()
      src/net/net.go:161 +0xe3
  net.func·061()
      src/net/timeout_test.go:616 +0x3ed

Goroutine 185 (running) created at:
  net.func·061()
      src/net/timeout_test.go:609 +0x288
```

C6BANK

Toolchain – pprof

https://golang.org/pkg/net/http/pprof/

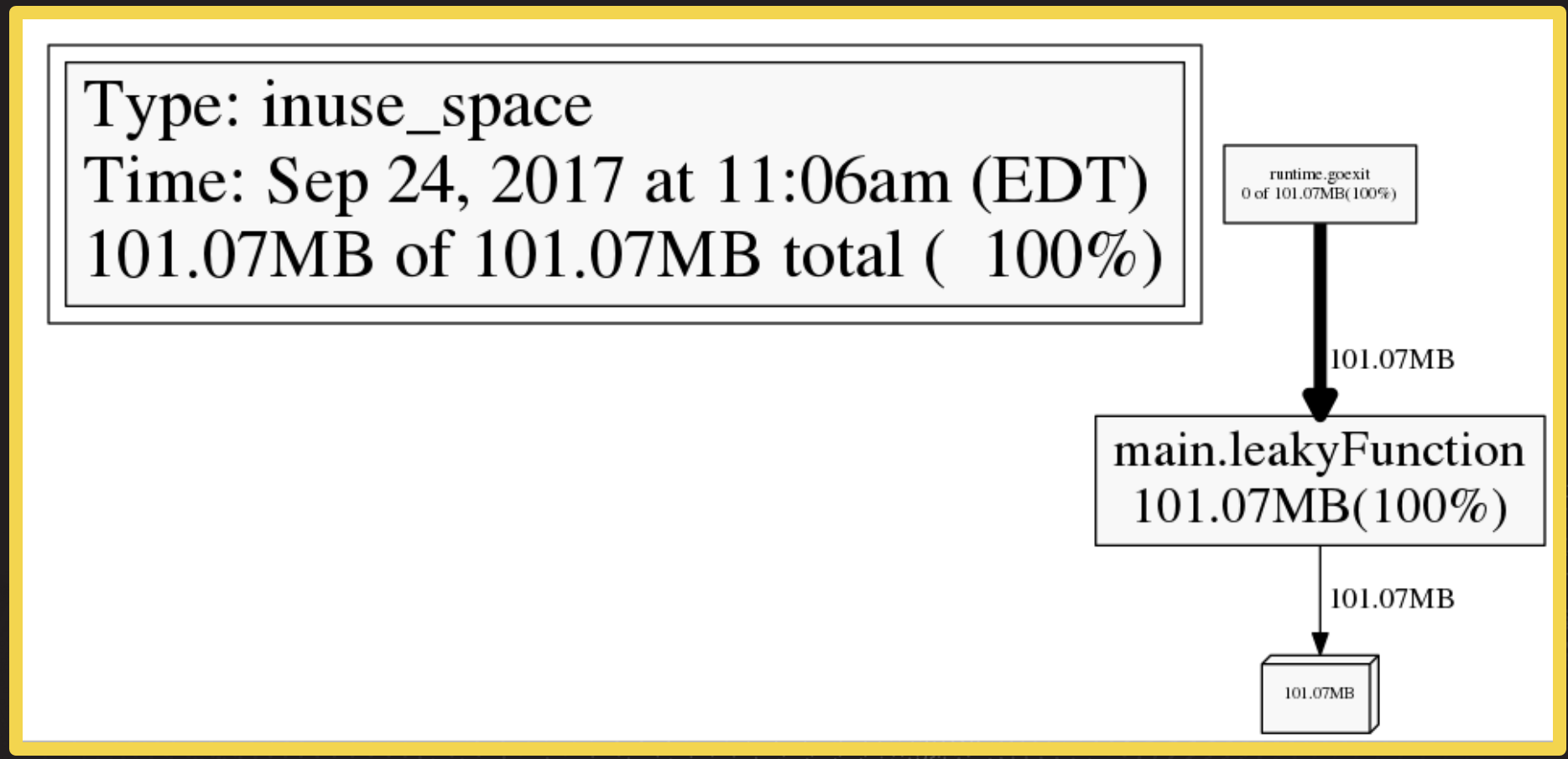
```
func main() {
    // we need a webserver to get the pprof webserver
    go func() {
        log.Println(http.ListenAndServe("localhost:6060", nil))
    }()
    fmt.Println("hello world")
    var wg sync.WaitGroup
    wg.Add(1)
    go leakyFunction(wg)
    wg.Wait()
}

func leakyFunction(wg sync.WaitGroup) {
    defer wg.Done()
    s := make([]string, 3)
    for i:= 0; i < 10000000; i++{
        s = append(s, "magical pandas")
        if (i % 100000) == 0 {
            time.Sleep(500 * time.Millisecond)
        }
    }
}
```

https://jvns.ca/blog/2017/09/24/profiling-go-with-pprof/

Toolchain - Profiling

```
go tool pprof -png http://localhost:6060/debug/pprof/heap > out.png
```

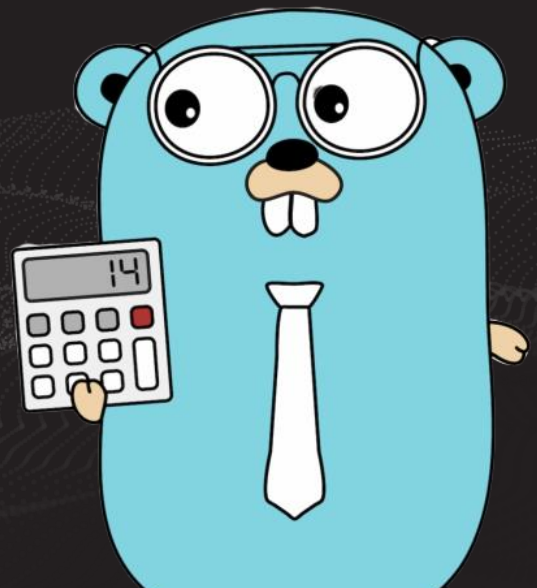


Case C6 Bank

Case C6 Bank

Nosso contexto:

- Event-driven
- Milhares de métricas e logs por segundo (tratados como eventos)
- Micro-serviços distribuídos e se comunicando
- Informações críticas de usuários
- Grandes massas de dados para processar



C6BANK

Case C6 Bank

Por que escolhemos Go?

- Binário autocontido (sem dependências)
- Grande capacidade de escalabilidade e performance
- Multiplataforma
- Conhecimento do time técnico
- Lib padrão que resolve ou ajuda quase sempre (standard libraries)
- Grande acervo de libs de apoio de código aberto com ótima qualidade (Comunidade muito atuante)
- Código mais limpo e padronizado, simplicidade
- É divertido =)

Case C6 Bank

Ferramentas que escolhemos e são escritas em Go:

- concourse
- terraform
- vault
- consul
- prometheus
- nomad
- nats.io
- snap
- moira

Case C6 Bank

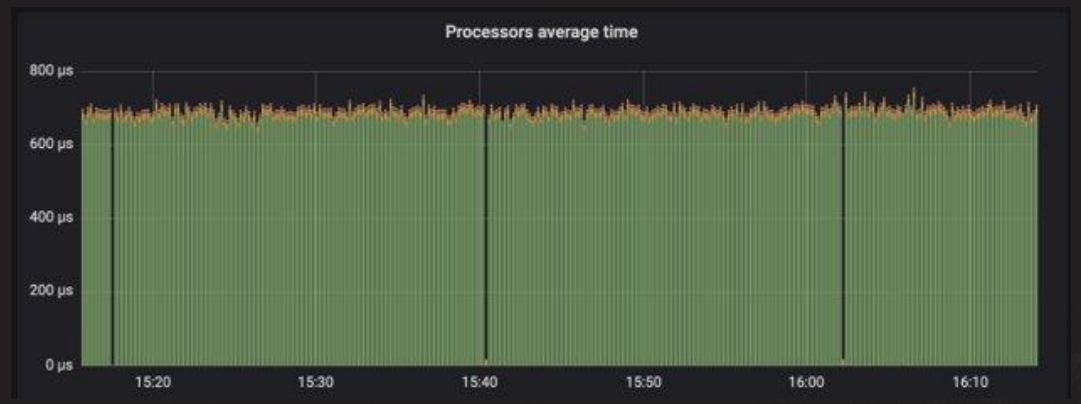
Batalhas, surras e cicatrizes! O que aprendemos até aqui:

- Channels são maravilhosos!
- Mais go routines nem sempre deixam mais eficiente
- Custam mais do que dinheiro:
  - trocas de contexto
  - alocação de estruturas
  - marshal/unmarshal
- *err == nil* é legal! \o/
- Nos ajudou muito aprender e seguir o layout padrão de projeto
- Ponteiros: use com cautela

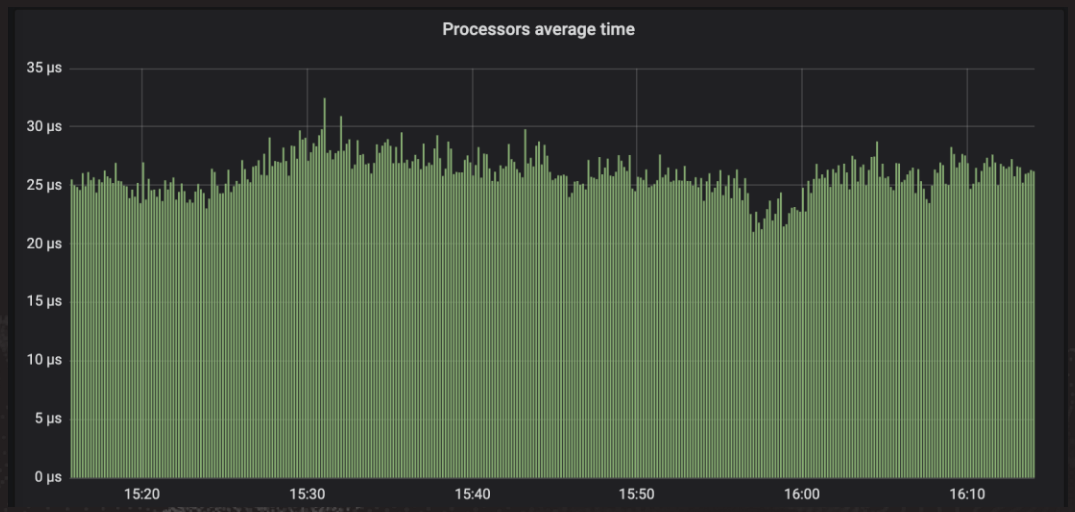
Case C6 Bank

Tirando leite de pedra: **26x mais rápido!**

Antes: ~ **0.65ms** por evento



Agora: ~ **0.025ms** por evento



C6BANK

Dúvidas?

C6 BANK