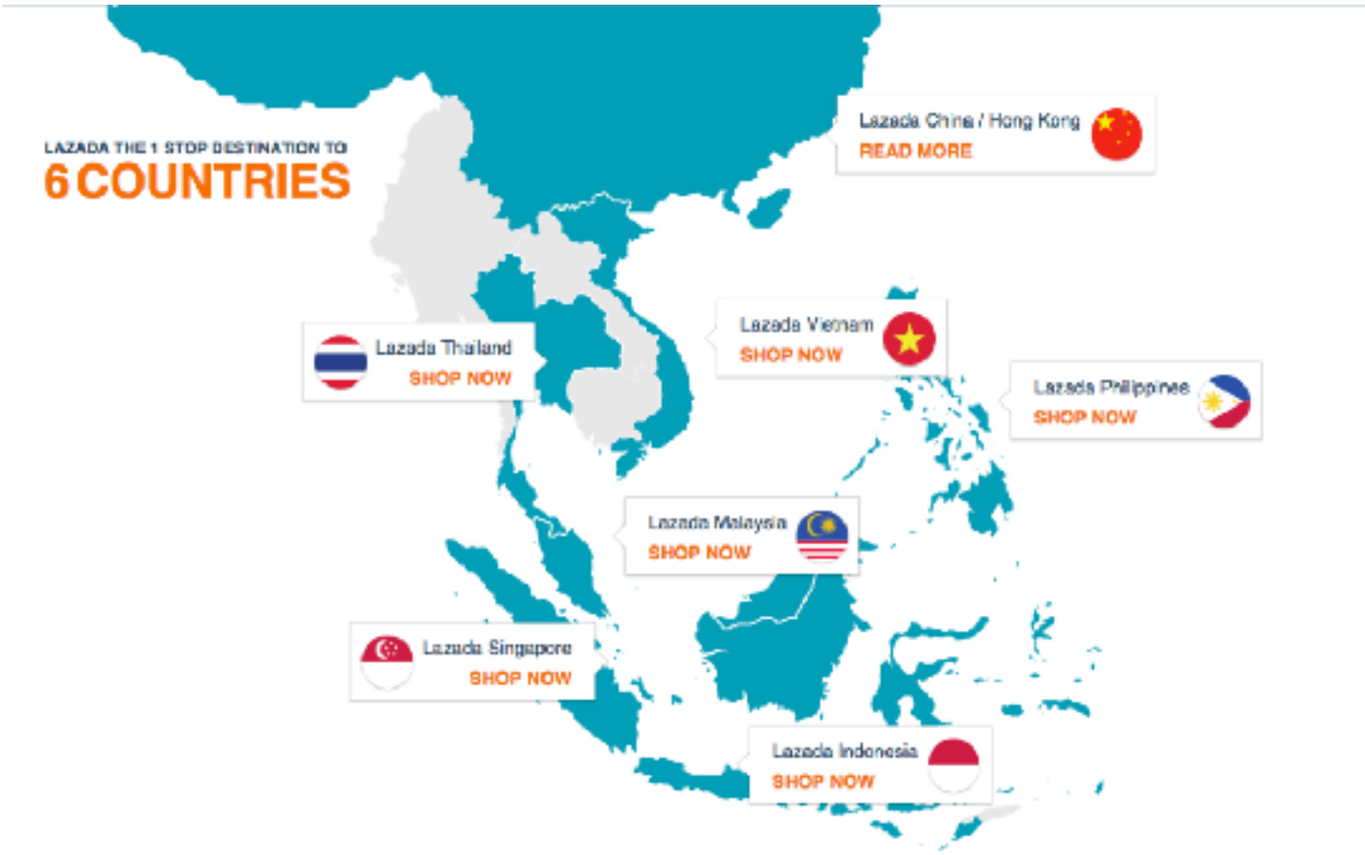
LAZADA GROUP

Golang – опыт промышленной разработки
Три года в бою

Васияров Юрий
SVP Engineering
Lazada Group

О компании Lazada

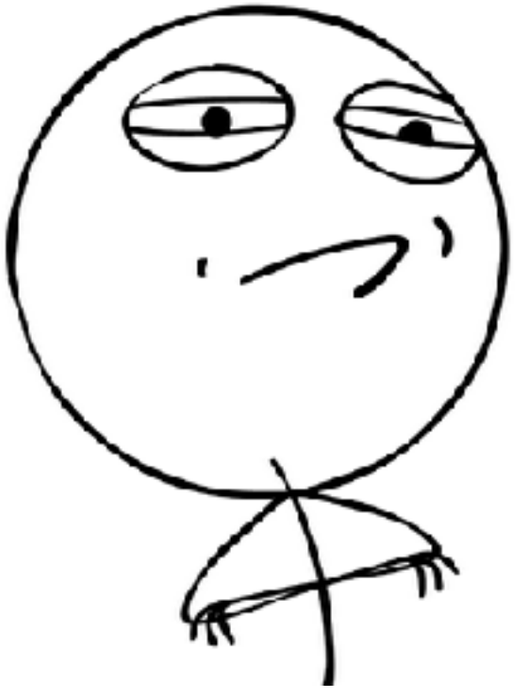
- Основана в 2012 году
- Крупнейшая e-commerce компания Юго-Восточной Азии
- Работает в 6 странах Юго-Восточной Азии с суммарным населением 650M человек
- Более 40 миллионов продуктов
- 1.2M товаров за время последней распродажи (3 дня)
- С 2016 года является частью Alibaba Group



Инфраструктура

Команда Lazada

- 4 TechHub: Вьетнам, Сингапур, Бангкок, Москва
- Более 650 инженеров
- Разработка платформы только на Golan
- 90% платформы разрабатывается в Москве
- Более 100 Golang программистов в московской команде



Почему именно Golang ?

LAZADA GROUP

История одной оптимизации

Оптимизации PHP

5.5 часов

2x

Параллелизация

2.5 часа

5x

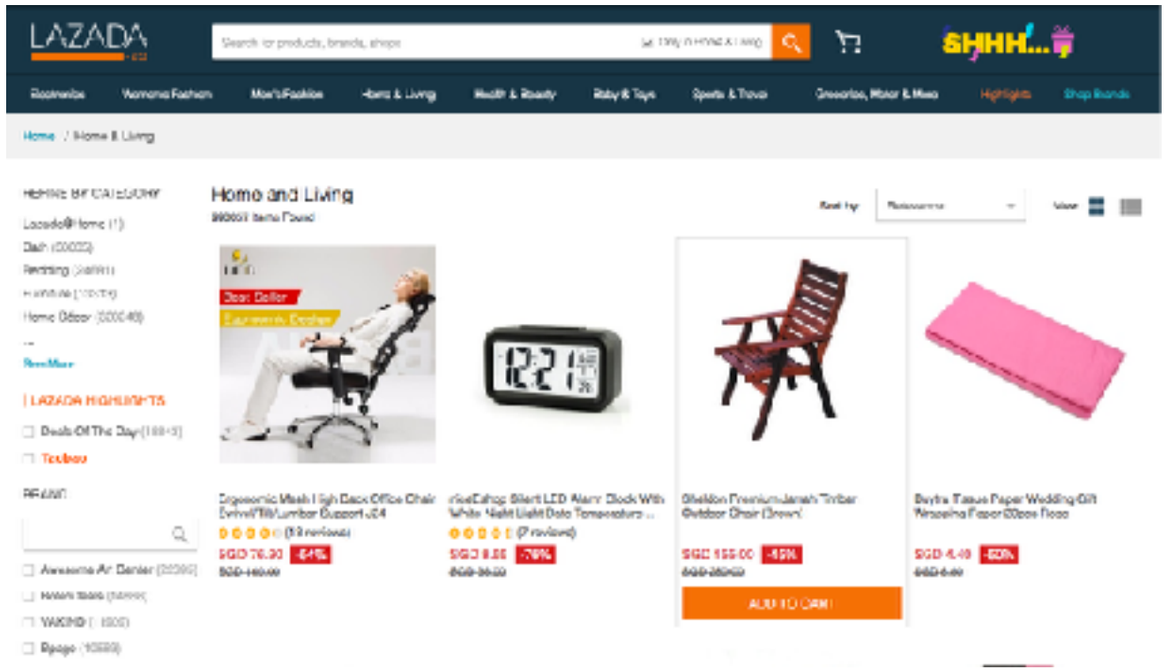
Оптимизация Go кода

30 минут

10x

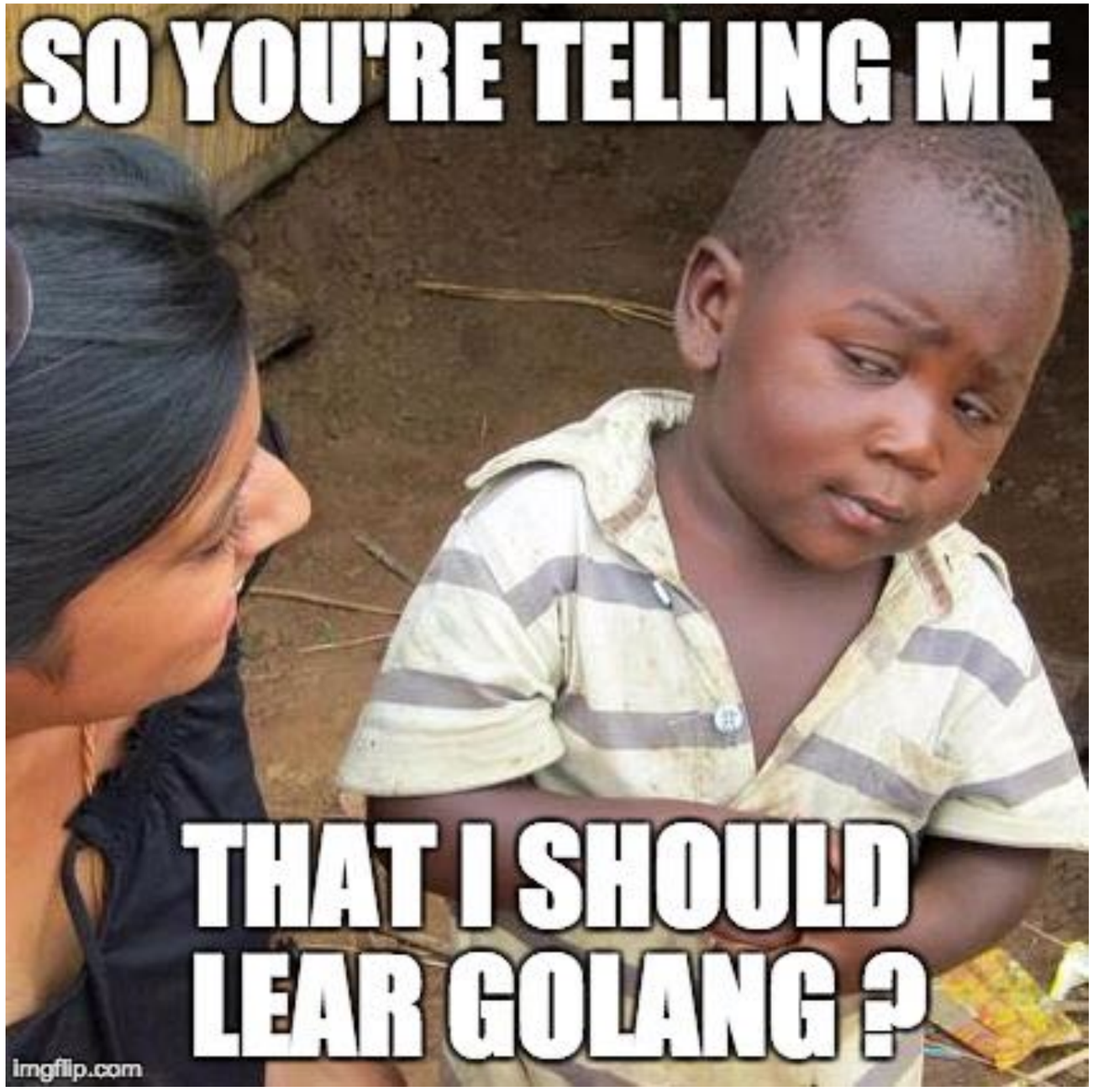
2-3 минуты

Проект Goblin



80 серверов → 20 серверов

Но где взять Golang программистов ?

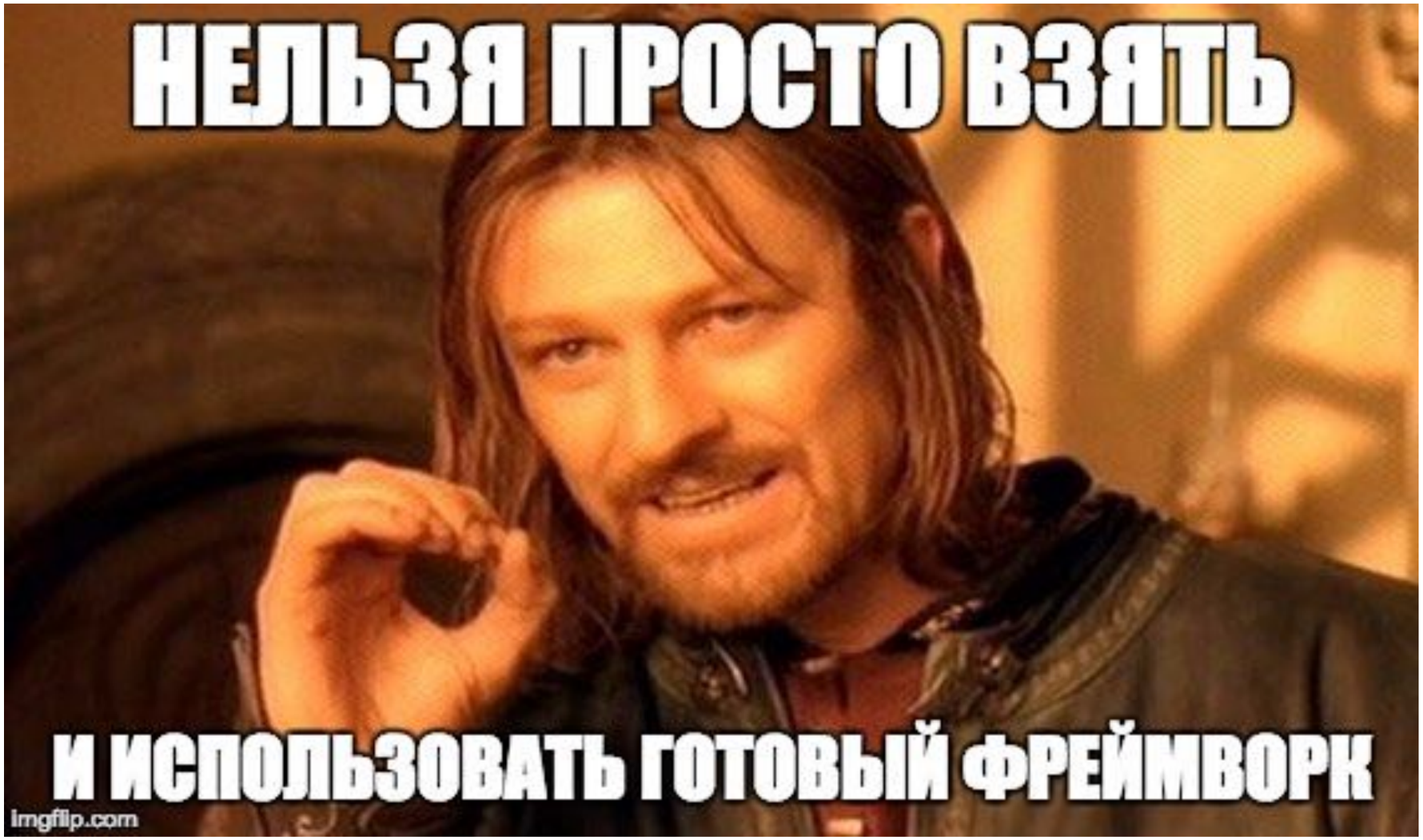


Какой фреймворк использовать ?

LAZADA GROUP

- Echo - https://github.com/labstack/echo
- kit - https://github.com/go-kit/kit
- go-micro - https://github.com/micro/go-micro

Фреймворк


НЕЛЬЗЯ ПРОСТО ВЗЯТЬ
И ИСПОЛЬЗОВАТЬ ГОТОВЫЙ ФРЕЙМВОРК
imgflip.com

Наш фреймворк - Brick

- Роутинг: gorilla/mux
- Транспорт: REST/RPC, grpc
- Dependency injection: github.com/tsaikd/inject
- Логгирование: небольшая обертка вокруг unixgram
- Мониторинг: обертка вокруг Prometheus клиента
- Работа с базой данных: gorp
- Юнит тесты: github.com/stretchr/testify или https://github.com/go-check/check

Документирование - Swagger

- Swagger описание генерируется автоматически при помощи рефлексии
- Также используется для расчета покрытия функциональными тестами
- github.com/lazada/swgen
- github.com/lazada/swgui

Профилирование

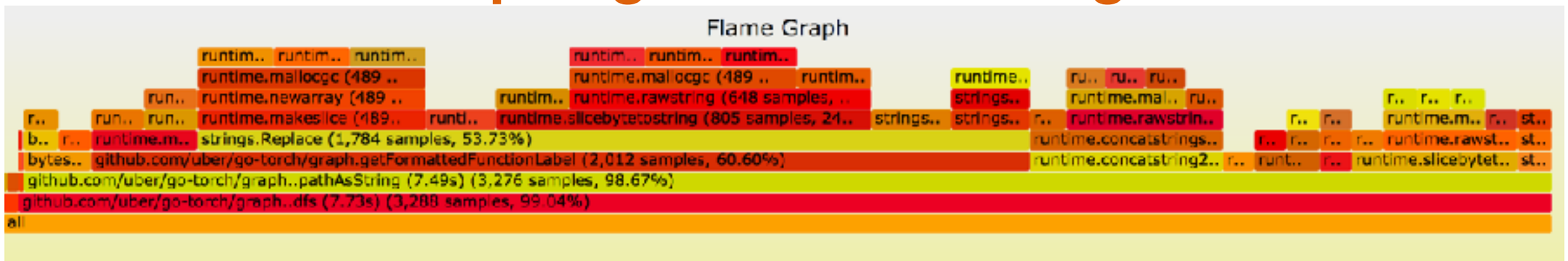
github.com/lazada/goprof

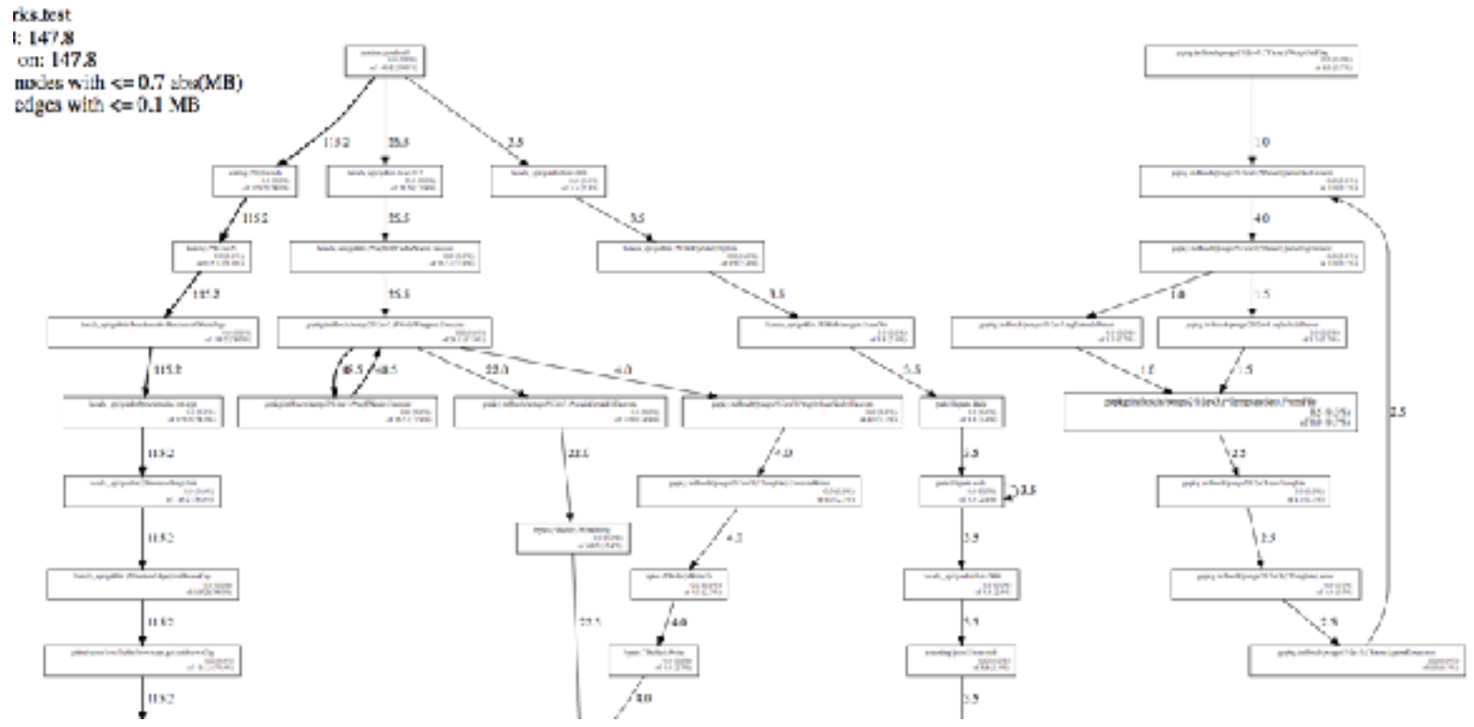
- Простой интерфейс для запуска/остановки профайлера
- Автоотключение через 5 минут
- Удобная возможность скачать профайл вместе с бинарным файлом
- Активно используется в production

Профилирование - Go Torch

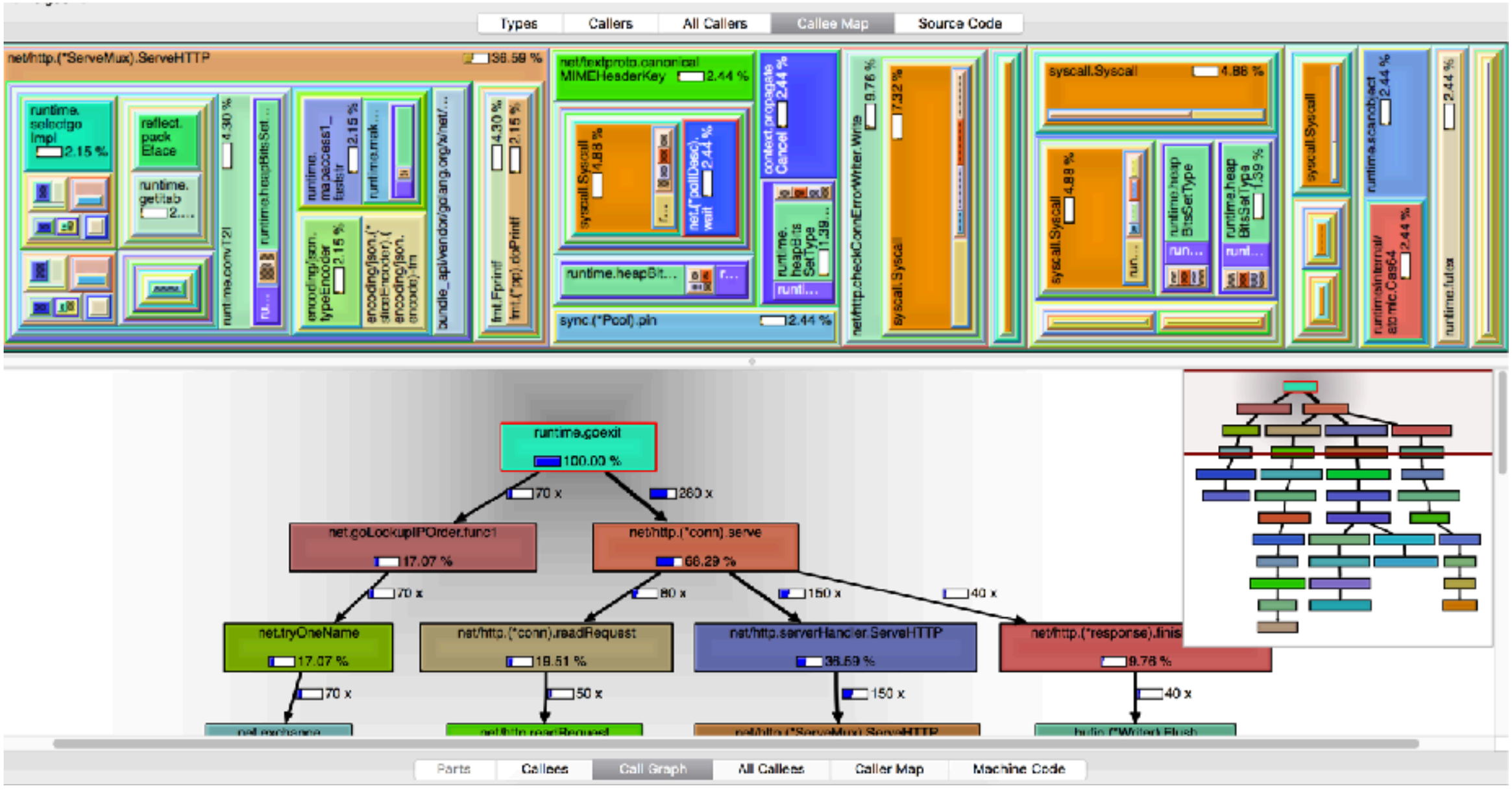
LAZADA GROUP

https://github.com/uber/go-torch



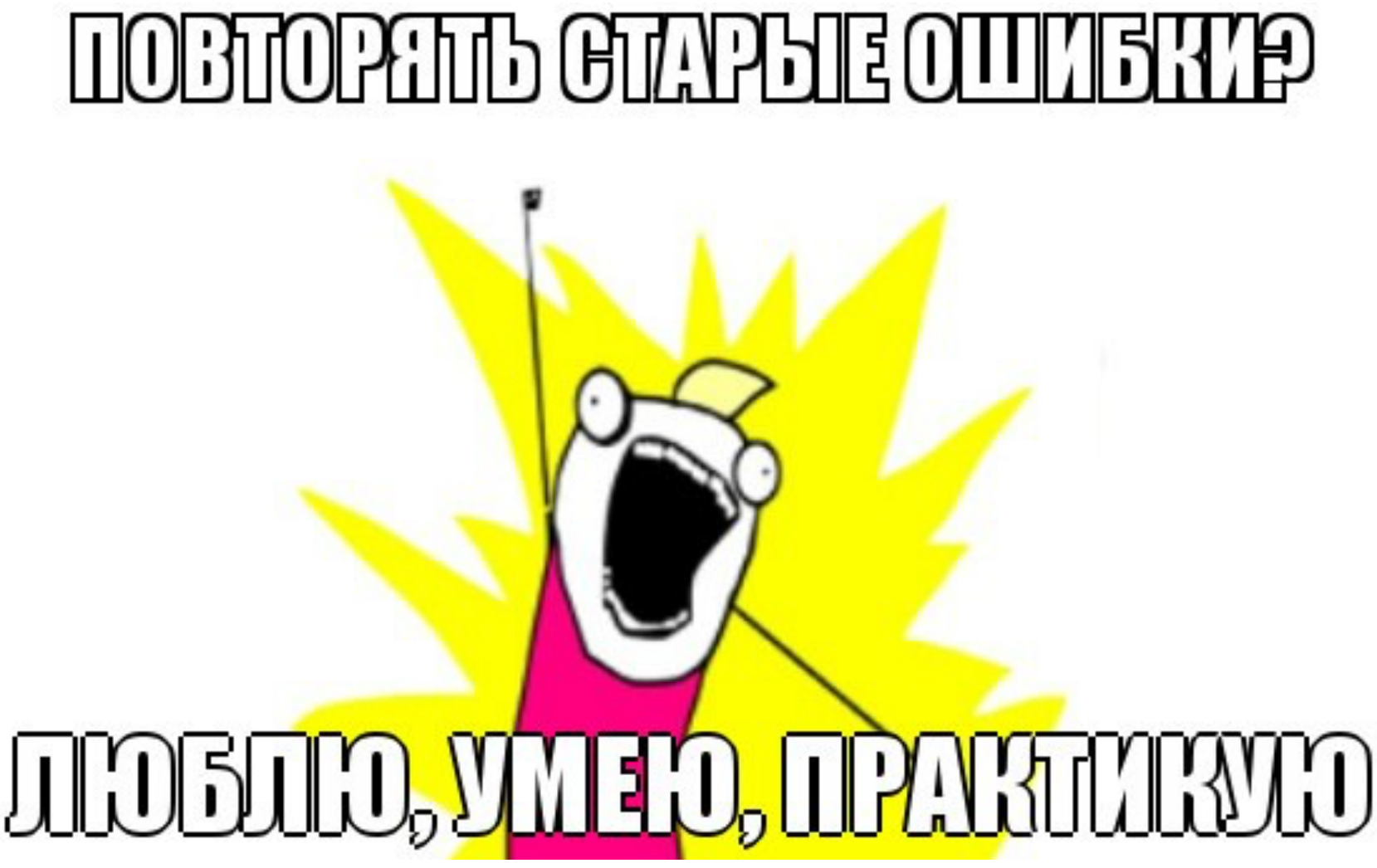


kcachegrind

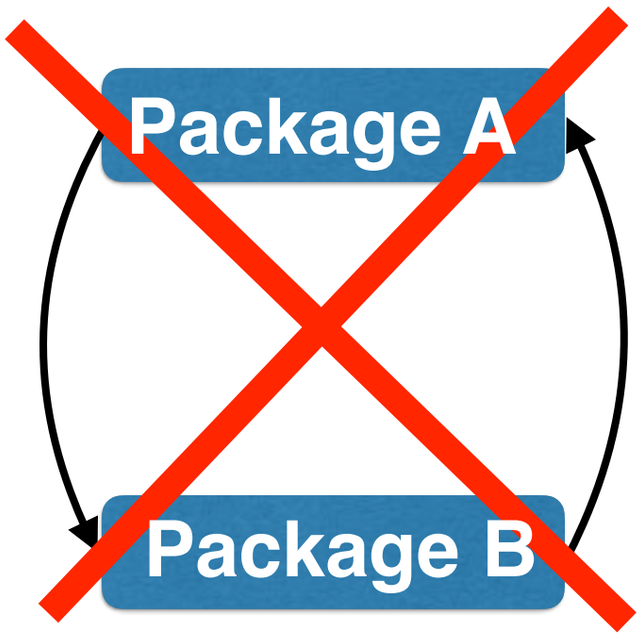


go tool pprof -callgrind -output callgrind.out ~/xxx_api ~/cpu-profile

Типичные ошибки



Циклические зависимости между пакетами



Циклические зависимости

- Не усложняйте
- Пакеты должны быть самодостаточными
- Взаимодействие между пакетами - только через интерфейсы

https://peter.bourgon.org/go-in-production/

Сокрытие переменных

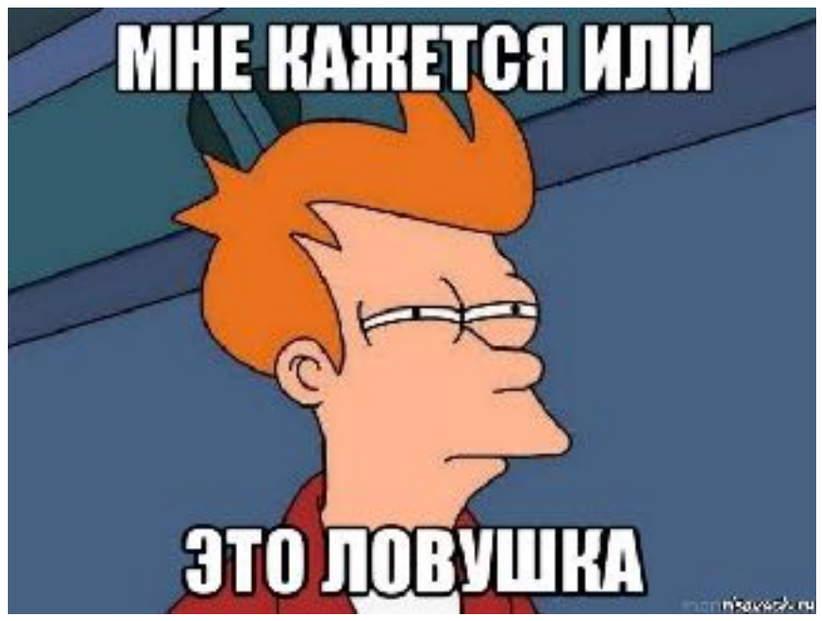
```
 7 func one() (int, error) {
 8         return 1, fmt.Errorf("shadowed error")
 9 }
10 func two() (int, error) {
11         return 2, fmt.Errorf("second error")
12 }
13
14 func main() {
15         one, err := one()
16         two, err := two()
17         fmt.Printf("Hello, bug: %v, %d, %d\n", err, one, two)
18 }
```

```
Hello, bug: second error, 1, 2
```

Кэширование в памяти

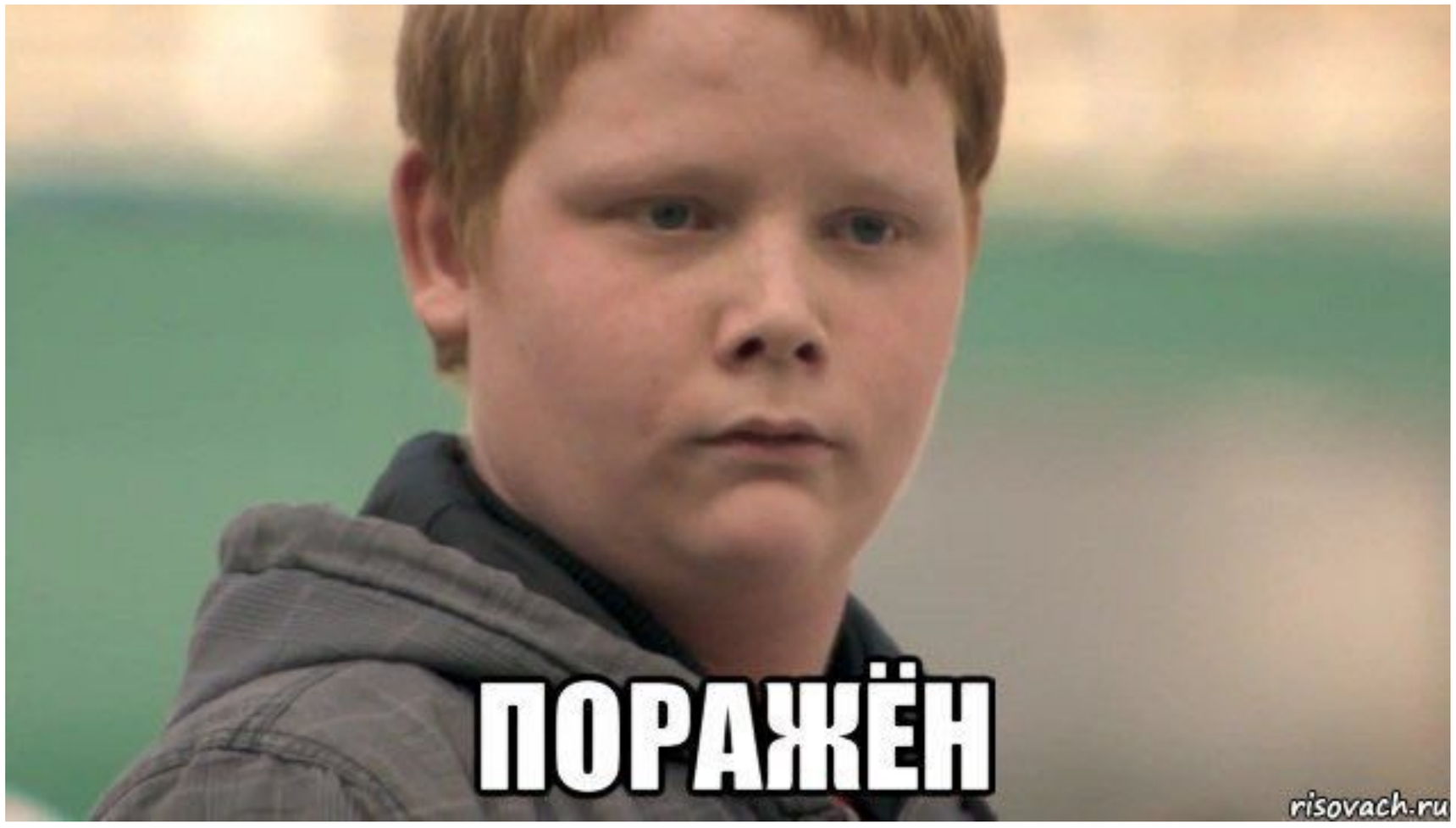
- Отдельный кэш для каждого экземпляра приложения. Меньше попаданий в кэш
- Должен быть ограничен по размеру
- Вытеснение также должно работать за O(1)
- Внутренние блокировки
- Кэширование ссылочных структур данных приводит к race condition

Может быть полезен:
github.com/lazada/go-copy-interfaces



map[]

map[] - это честный hash table
Является неупорядоченным



Передача аргументов - указатели

```
5  import "fmt"
6
7  type Person struct {
8      Name string
9  }
10 func main() {
11         vasiya := Person{Name: "Vasya"}
12         petya := &Person{Name: "Petya"}
13         fmt.Printf("Hello, %v, %v\n", vasiya.Name, petya.Name)
14 }
```

Передача аргументов - указатели

```
import "fmt"

type Person struct {
        Name string
}

func (p Person) changeName(name string) {
        p.Name = name
}
func (p *Person) reallyChangeName(name string) {
        p.Name = name
}

func main() {
        vasiya := Person{Name: "Vasya"}
        vasiya.changeName("Kolya")

        petya := &Person{Name: "Petya"}
        petya.reallyChangeName("Misha")

        fmt.Printf("Hello, %v, %v\n", vasiya.Name, petya.Name)
}
```

```
LAZ-RU-L-M-0098:tmp yuriy.vasiyarov$ go run main.go
Hello, Vasya, Misha
```

Race conditions

- Go race detector прекрасен
- Но медленный (2-20x)
- Ограничение на 8192 goroutines.
  Нигде не документировано

Неуправляемые горутины

```
http.HandleFunc("/bar", func(w http.ResponseWriter, r *http.Request) {
    doSomeImportantStuff(w, r);

    //мы не хотим ждать пока статистика обновится
    //мы хотим ответить клиенту быстрее
    go updateStatistic();
})
```

Все отлично работает в 99% случаев

database/sql

sql.DB - это пул соединений

- Когда создается подключение ?
- Когда оно закрывается ?
- А сколько вообще этих подключений ?
- А есть ли ограничение на максимальное количество соединений ?
- А что случится если в пуле не будет свободных соединений ?
- А что если SQL сервер закроет соединение ?

Утечка соединений

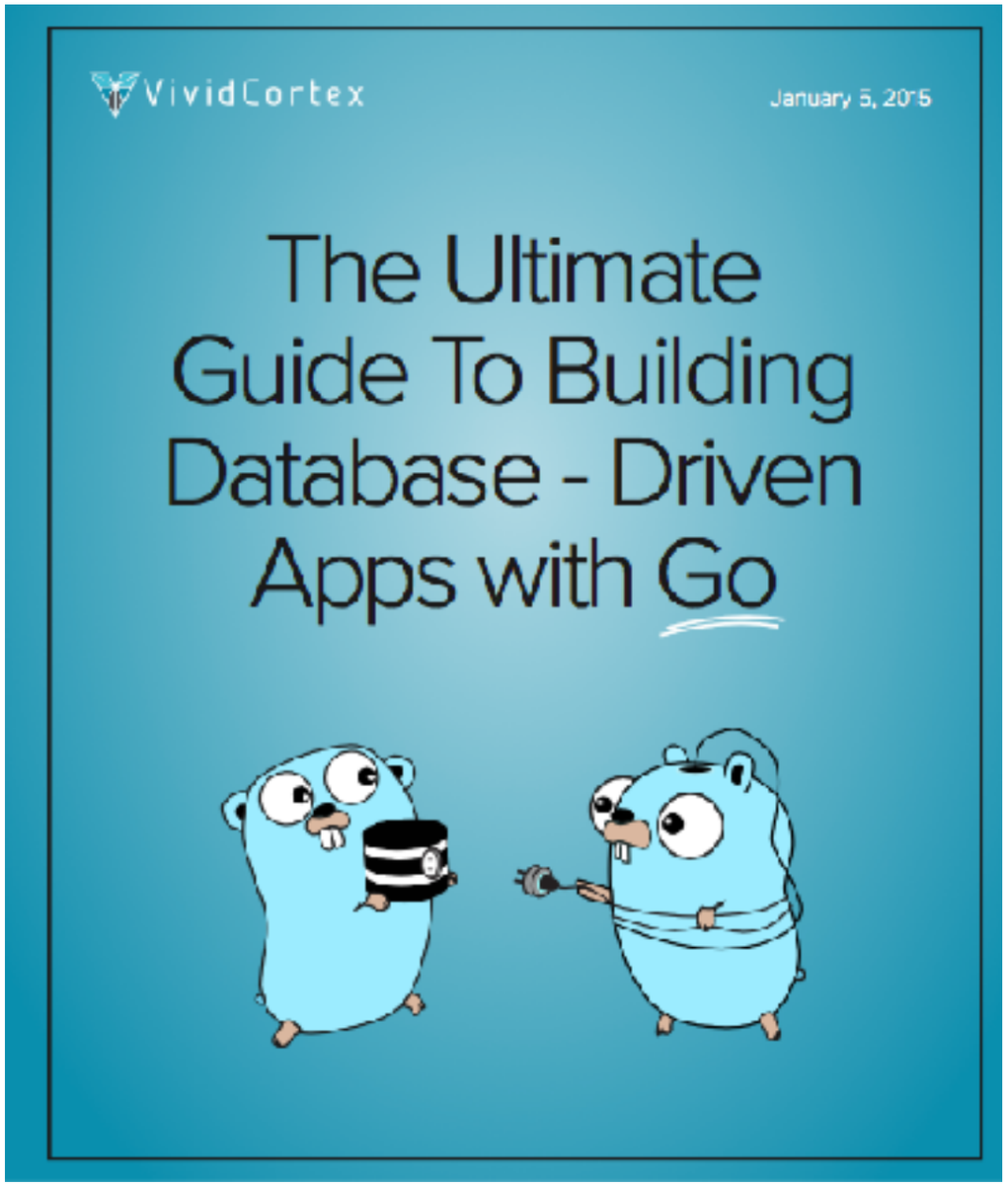
```
rows, _ := db.Query("SELECT something FROM from something WHERE ID in (?)")
defer rows.Close()
for rows.Next() {
        var id int
        if err := rows.Scan(&id); err != nil {
                log.Fatal(err)
        }
        rows2, _ := db.Query("SELECT another_thing  FROM from something WHERE ID in (?)", id)
        defer rows2.Close()

        doSomeHeavyProcessing()
}
```

PS
Никогда так не пишите :-)

К чему приводят утечки соединений

Типичные ошибки при работе с database/sql

- Использование Defer внутри цикла
- Использование нескольких **sql.DB** объектов
- Prepared statement используются только однажды
- Использование **db.Query()** не для чтения
- Предположение о том что последующие запросы будут использовать то же подключение
- Использование объекта **sql.DB** во время работы с транзакцией



Статические анализаторы кода

LAZADA GROUP

github.com/alecthomas/gometalinter

--enable=vet --enable=vetshadow
--enable=gosimple
--enable=staticcheck
--enable=ineffassign
--enable=lll
--enable=goconst
--cyclo-over=50
--dupl-threshold=290
--enable=unconvert
--enable=unused
--enable=varcheck
--enable=dupl

Code style

- **go fmt/goimports** - не обсуждается
- **go vet** - не обсуждается
- Go Code Review Comments - не является обязательным

Как выучить Golang за 24 часа

- Golang tour - http://tour.golang.org/#1
- OOP
    - Methods: a taste of OOP - http://go-book.appspot.com/methods.html
    - More meth(odes) please! http://go-book.appspot.com/more-methods.html
- 50 Shades of Go - http://devs.cloudimmunity.com/gotchas-and-common-mistakes-in-go-golang/
- Effective Go - http://golang.org/doc/effective_go.html
- Golang Blog
    - http://blog.golang.org/slices
    - http://blog.golang.org/go-slices-usage-and-internals
    - http://blog.golang.org/go-maps-in-action
    - http://blog.golang.org/cover
- Go Data Structures
    - https://research.swtch.com/godata
    - https://research.swtch.com/interfaces
- Common Pitfalls When Using database/sql in Go - https://www.vividcortex.com/blog/2015/09/22/common-pitfalls-go/

Вопросы ?

Васияров Юрий,
SVP Engineering,
Lazada Group

yuriy.vasiyarov@gmail.com
https://github.com/yvasiyarov