

Яндекс.Практикум

Некоторые техники оптимизации Go-кода

Яндекс Практикум

Что нас сегодня ждет

- разберем некоторые универсальные техники увеличения производительности Go-программ
- чуть-чуть заглянем под капот Go рантайма

Преждевременная оптимизация – корень всех зол?

“...преждевременная оптимизация – корень всех зол”

– Дональд Кнут

Преждевременная оптимизация – корень всех зол?

“Программисты тратят огромное количество времени на размышления или беспокойство о скорости некритических частей своих программ, и эти попытки достижения эффективности фактически имеют сильное негативное влияние, когда речь идет об отладке и обслуживании. Стоит забыть о мелких увеличениях эффективности, скажем, в 97% случаев: **преждевременная оптимизация - корень всех бед**. Однако не следует упускать возможности повышения эффективности в этих критических 3%.”

– Дональд Кнут

Оптимизации на реальных проектах

Условия для оптимизации существующего кода:

- вы обнаружили bottleneck
- оптимизируемый участок покрыт тестами
- (желательно) вы написали бенчмарки, чтобы доказать увеличение производительности благодаря вашему решению

Оптимизации ради оптимизаций – плохая идея

Как обнаружить bottleneck ?

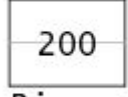
Необходимые условия:

- понять в каком месте программа тратит много времени: *pprof*, анализ алгоритмов...
- понять что именно занимает много времени:
 - анализ кода
 - чтение сгенерированного листинга Go-ассемблера

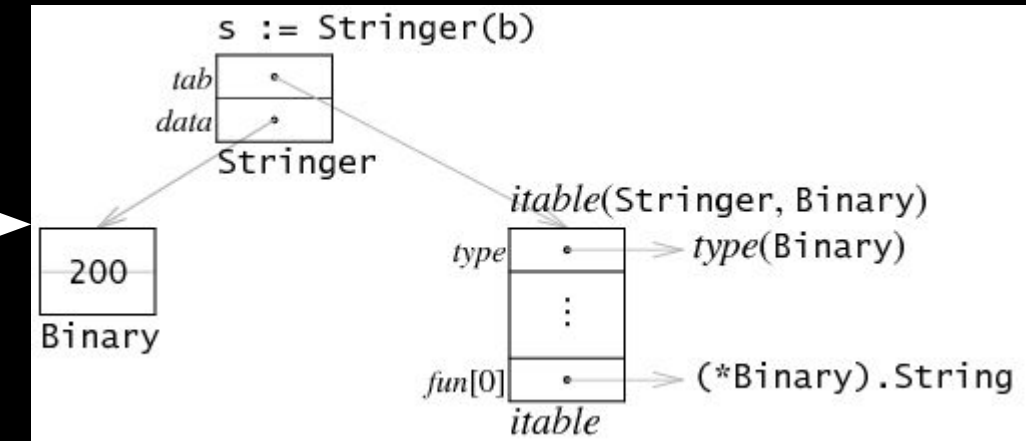
Интерфейсы в Go

```
type Binary uint64  
  
func (i Binary) String() string {  
    return strconv.Uitob64(i.Get(), 2)  
}  
  
func (i Binary) Get() uint64 {  
    return uint64(i)  
}
```

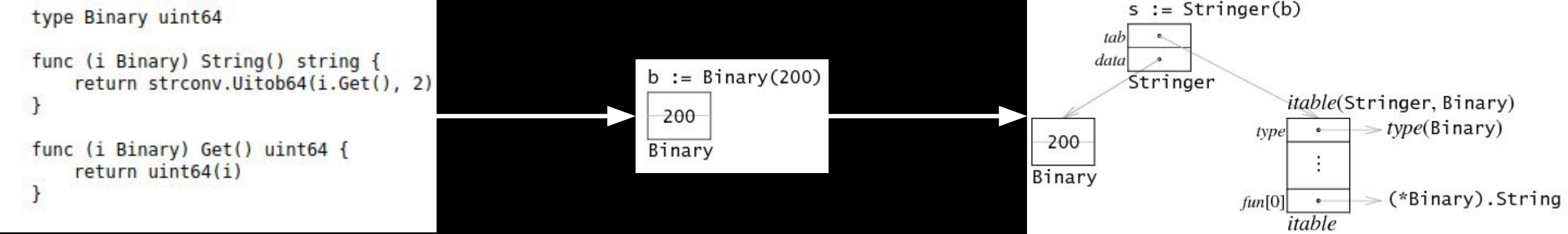
b := Binary(200)



A small rectangular box representing memory. The top half contains the value '200'. The bottom half contains the label 'Binary'.



Интерфейсы в Go

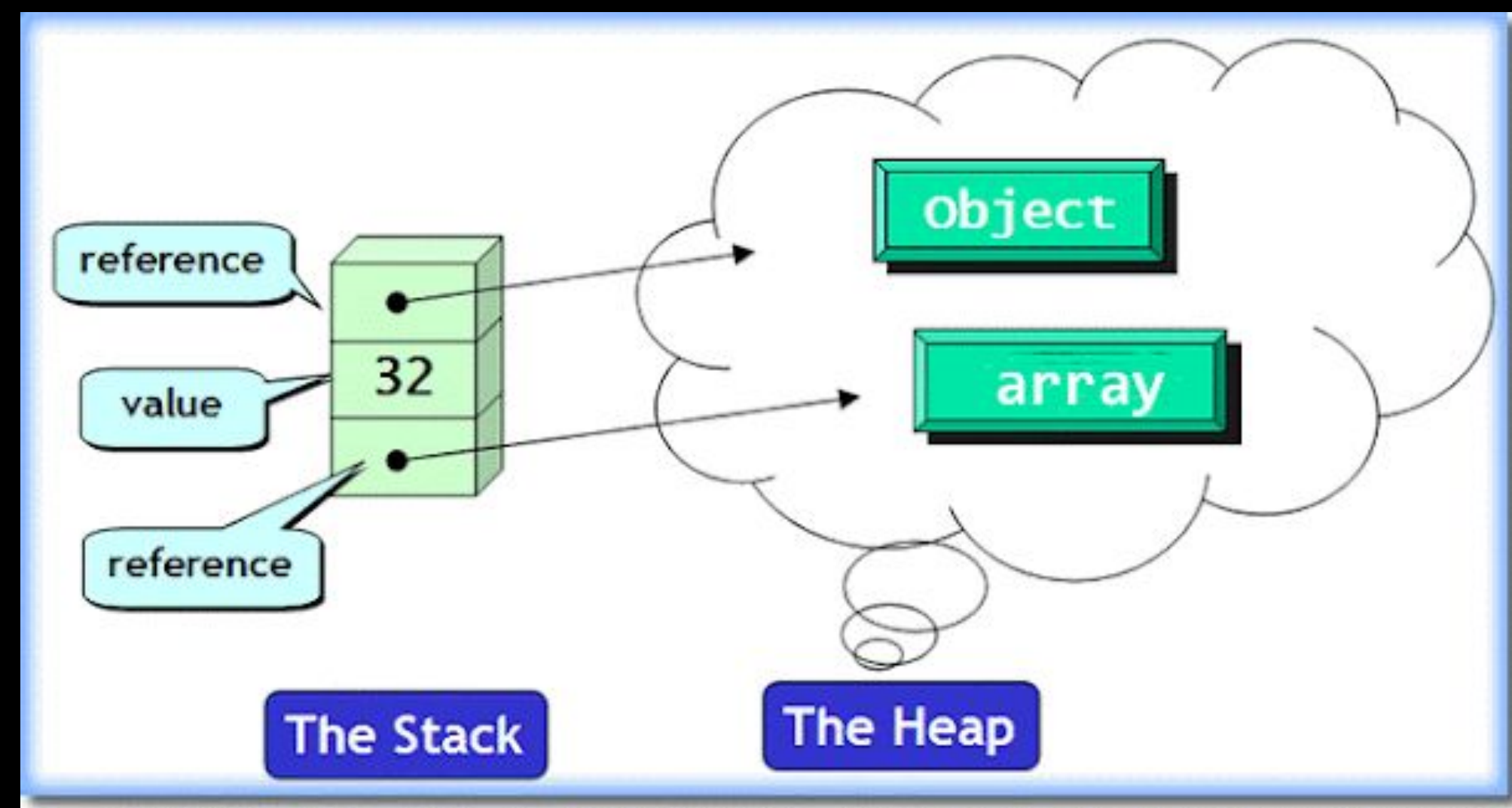


key: bit: bits , c: characters, d: decimal digits , w: word size of architecture, n: variable size, wm: word mark							
Year	Computer architecture	Word size <i>w</i>	Integer sizes	Floating-point sizes	Instruction sizes	Unit of address resolution	Char size
2013	ARMv8-A and ARMv9-A	64 bit	8 bit, $\frac{1}{4}w$, $\frac{1}{2}w$, w	$\frac{1}{2}w$, w	$\frac{1}{2}w$	8 bit	8 bit
2003	x86-64	64 bit	8 bit, $\frac{1}{4}w$, $\frac{1}{2}w$, w	$\frac{1}{2}w$, w , 80 bit	8 bit, ... 120 bit	8 bit	8 bit

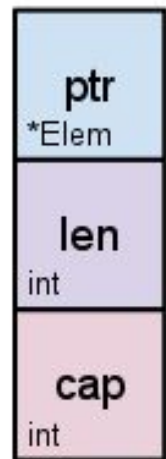
```
goos: linux
goarch: amd64
pkg: optimizations/cmd/strconv_vs_fmt
cpu: Intel(R) Core(TM) i7-10850H CPU @ 2.70GHz
BenchmarkFormatIntStrconv-12      39540540      29.51 ns/op      7 B/op      0 allocs/op
BenchmarkFormatIntSprintf-12     12431749      91.58 ns/op     16 B/op      1 allocs/op
PASS
ok      optimizations/cmd/strconv_vs_fmt  2.439s
```


Stack vs Heap

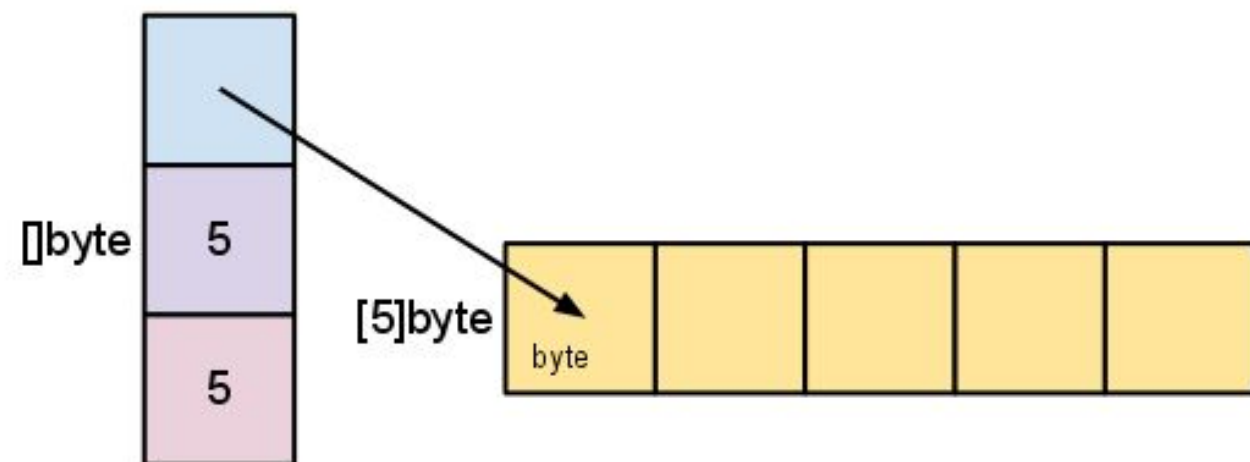
1. Память в стеке очищается
“автоматически”, в куче – при помощи
GC
2. Память в стеке линейна, в куче –
разрознена
3. Доступ к памяти стека быстрее, чем к
памяти кучи
4. Память на стеке не может
использоваться сразу несколькими
горутинами



Слайсы в Go



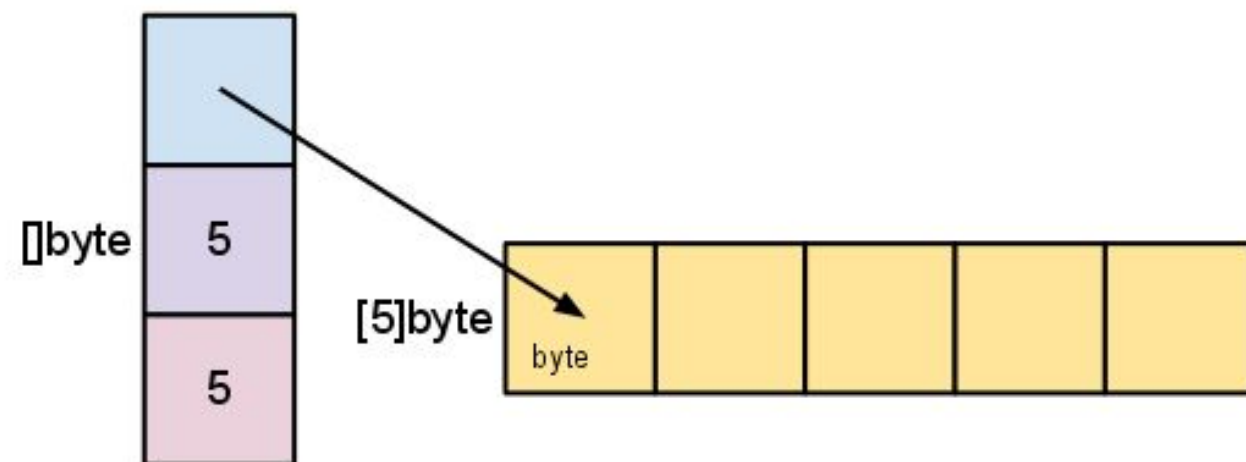
Our variable `s`, created earlier by `make([]byte, 5)`, is structured like this:



Слайсы в Go



Our variable `s`, created earlier by `make([]byte, 5)`, is structured like this:



Что делать, если необходимая емкость неизвестна:

1. собрать данные по использованию слайса в “боевых условиях”
2. Далее возможны два варианта:
 - а. если быстродействие важнее RAM – взять с запасом (например, $\times 1.2$ от максимального значения)
 - б. если важно оптимизировать по памяти: взять какую-то перцентиль (90-ю или 99-ю) и использовать ее

Спасибо за внимание!

Вопросы?