THE STATE OF GO

What's New since Go 1.20

白明 Tony Bai

《Go语言精进之路》作者
tonybai.com博主

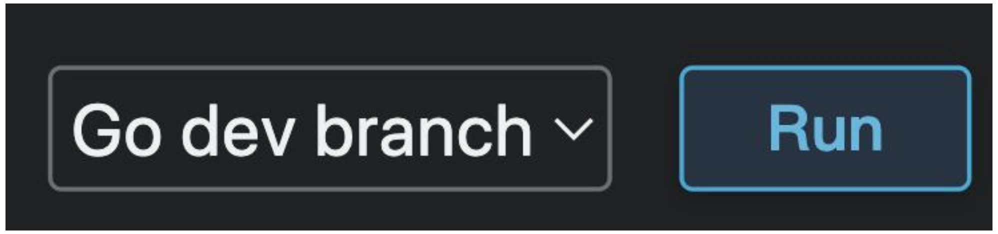
那段时间错过的Go版本

- 2022.3 Go 1.18 泛型落地
- 2022.8 Go 1.19 修订Go内存模型文档、支持Soft memory limit
- 2023.3 Go 1.20 引入Profile-Guided Optimizations

- Since Go 1.20… …

提示

本Slide中的代码可以在https://go.dev/play/上体验



目 录

第一部分

Changes to the language

新增泛型版max和min预定义函数

函数原型:

```
// builtin
import “cmp”

func max[T cmp.Ordered](x T, y ...T) T
func mix[T cmp.Ordered](x T, y ...T) T
```

示例:

```
var x, y int
m := min(x)                    // m == x
m := max(x, y, 10)             // m is the larger of x and y but at least 10
t := max("", "foo", "bar")  // t == "foo"
q := max(1, “foo”, “bar”)   // invalid argument: mismatched types untyped int and untyped
string (type of "foo")
```

新增clear预定义函数

Before go 1.21

```
m := map[float64]string{
    math.NaN(): "a",
}
fmt.Println(m) //map[NaN:a]
delete(m, math.NaN())
fmt.Println(m) //map[NaN:a]
```

With clear in go 1.21

```
m := map[float64]string{
    math.NaN(): "a",
}
fmt.Println(m) //map[NaN:a]
clear(m)
fmt.Println(m) //map[]
```

https://github.com/golang/go/issues/56351

新增clear预定义函数

clear的语义

调用	实参类型	结果
clear(m)	map[K]T	删除m中所有键值对，得到一个空map(len(m)==0)
clear(sl)	[]T	将[]T中所有元素重置为T类型的零值

clear(m)没有释放key和value占用的内存!

改变panic(nil)语义

```
defer func() {
    if err := recover(); err != nil {
        fmt.Printf("panicked: %v\n", err)
        return
    }
    fmt.Println("it's ok")
}()
panic("nil")
```

Before go 1.21

```
it’s ok
```

go 1.21

```
panicked: panic called with
nil argument
```

Go 1.21编译器会将panic(nil)替换为panic(new(runtime.PanicNilError))

https://github.com/golang/go/issues/25448

实验特性：per-iteration loop variable

```
var f func()
for i := 0; i < 10; i++ {
        if i == 0 {
                f = func() { print(i) }
        }
        f()
}
```

per-loop variable

```
0123456789
```

```
var f func()
i := 0
for ; i < 10; i++ {
        if i == 0 {
                f = func() { print(i) }
        }
        f()
}
```

https://github.com/golang/go/issues/60078

实验特性：per-iteration loop variable

```
var f func()
for i := 0; i < 10; i++ {
        if i == 0 {
                f = func() { print(i) }
        }
        f()
}
```

Per-iteration variable
GOEXPERIMENT=loopvar

```
0000000000
```

```
var f func()
for i := 0; i < 10; i++ {
    i := i
        if i == 0 {
                f = func() { print(i) }
        }
        f()
}
```

https://github.com/golang/go/issues/60078

第二部分

Ports

WebAssembly

增加对WASI的支持：GOOS=wasip1 GOARCH=wasm

标准化的WASI：在web之外运行WebAssembly的系统接口

```
$ GOARCH=wasm GOOS=wasip1 gotip build -o main.wasm main.go

$curl https://wazero.io/install.sh
$wazero run main.wasm
hello
```

https://github.com/golang/go/issues/58141

第三部分

Changes to the standard library

增加log/slog

slog：支持日志级别的高性能结构化日志包

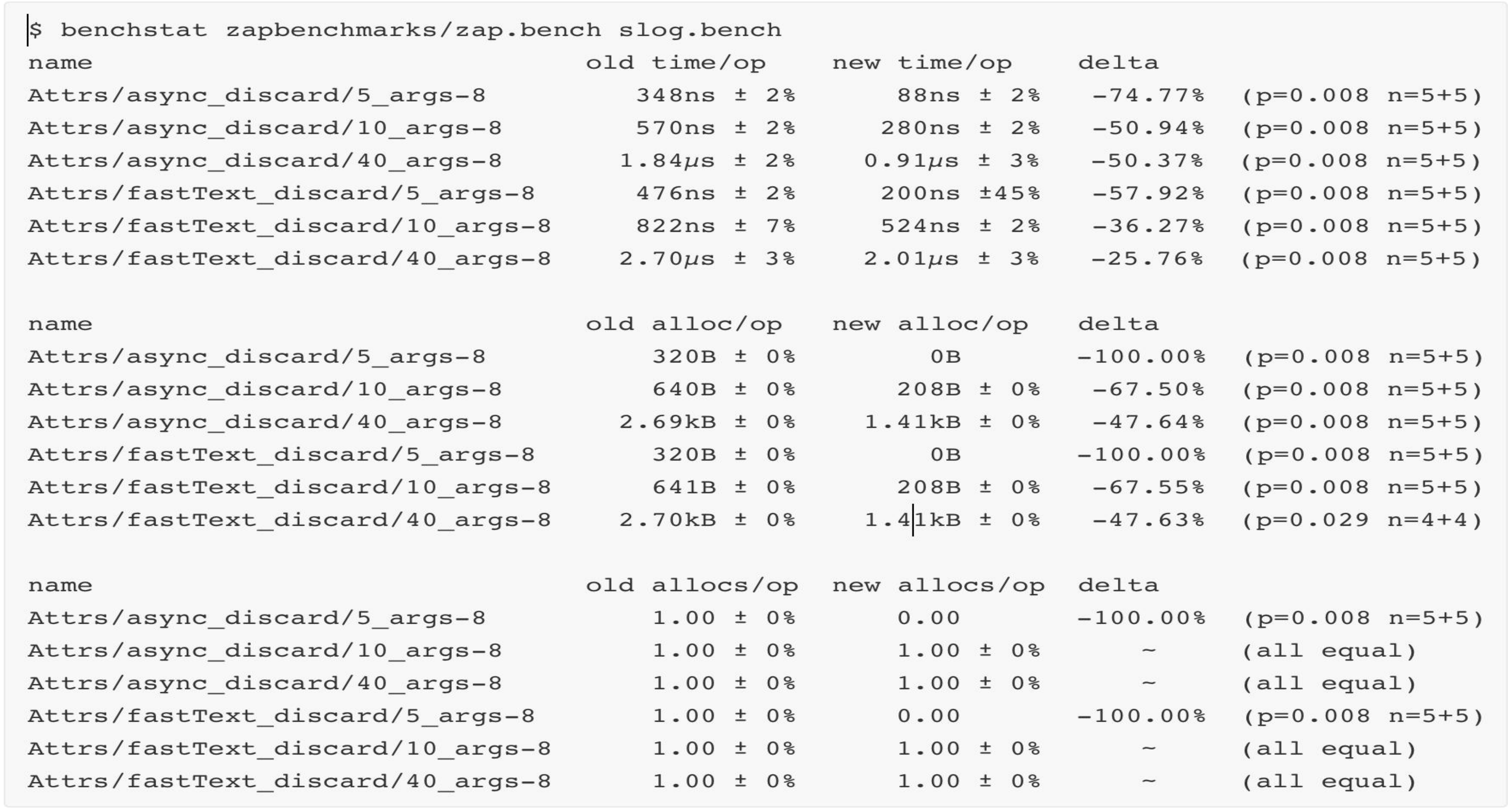
```
slog.Info("hello", "gopherchina", 2023)

slog.SetDefault(slog.New(slog.NewTextHandler(os.Stderr, nil)))
slog.Info("hello", "gopherchina", 2023)

slog.SetDefault(slog.New(slog.NewJSONHandler(os.Stderr, nil)))
slog.Info("hello", "gopherchina", 2023)
```

```
2023/06/09 23:00:00 INFO hello gopherchina=2023

time=2023-06-09T23:00:00.000Z level=INFO msg=hello gopherchina=2023

 {"time":"2023-06-09T23:00:00Z","level":"INFO","msg":"hello","gopherchina":2023}
```

https://github.com/golang/go/issues/56345

增加log/slog

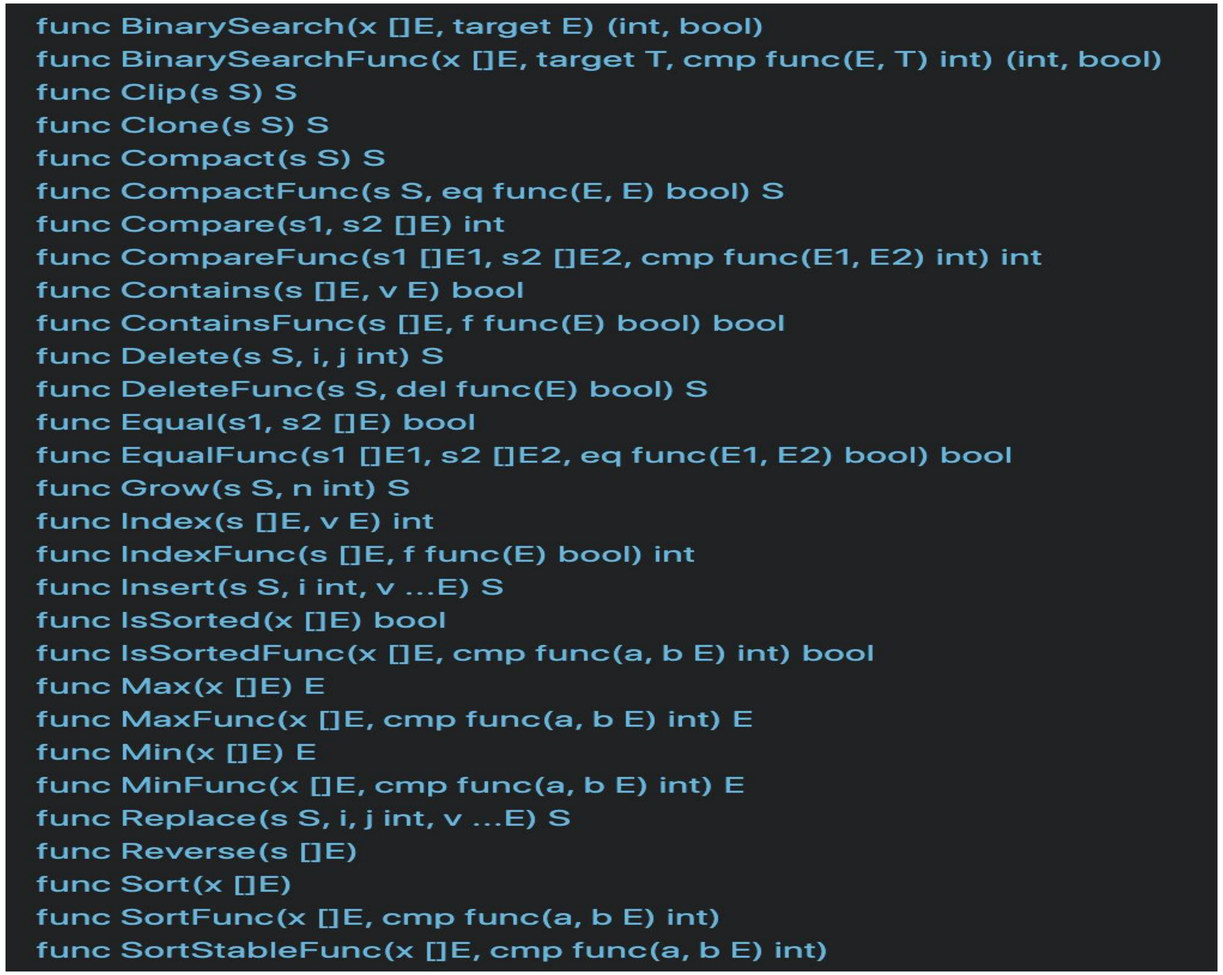
slog benchmark



```
$ benchstat zapbenchmarks/zap.bench slog.bench
name                              old time/op    new time/op    delta
Attrs/async_discard/5_args-8         348ns ± 2%      88ns ± 2%   -74.77%  (p=0.008 n=5+5)
Attrs/async_discard/10_args-8        570ns ± 2%     280ns ± 2%   -50.94%  (p=0.008 n=5+5)
Attrs/async_discard/40_args-8       1.84µs ± 2%    0.91µs ± 3%   -50.37%  (p=0.008 n=5+5)
Attrs/fastText_discard/5_args-8      476ns ± 2%     200ns ±45%   -57.92%  (p=0.008 n=5+5)
Attrs/fastText_discard/10_args-8     822ns ± 7%     524ns ± 2%   -36.27%  (p=0.008 n=5+5)
Attrs/fastText_discard/40_args-8    2.70µs ± 3%    2.01µs ± 3%   -25.76%  (p=0.008 n=5+5)

name                              old alloc/op   new alloc/op   delta
Attrs/async_discard/5_args-8          320B ± 0%        0B       -100.00%  (p=0.008 n=5+5)
Attrs/async_discard/10_args-8         640B ± 0%      208B ± 0%   -67.50%  (p=0.008 n=5+5)
Attrs/async_discard/40_args-8       2.69kB ± 0%    1.41kB ± 0%   -47.64%  (p=0.008 n=5+5)
Attrs/fastText_discard/5_args-8       320B ± 0%        0B       -100.00%  (p=0.008 n=5+5)
Attrs/fastText_discard/10_args-8      641B ± 0%      208B ± 0%   -67.55%  (p=0.008 n=5+5)
Attrs/fastText_discard/40_args-8    2.70kB ± 0%    1.41kB ± 0%   -47.63%  (p=0.029 n=4+4)

name                              old allocs/op  new allocs/op  delta
Attrs/async_discard/5_args-8          1.00 ± 0%      0.00       -100.00%  (p=0.008 n=5+5)
Attrs/async_discard/10_args-8         1.00 ± 0%      1.00 ± 0%        ~     (all equal)
Attrs/async_discard/40_args-8         1.00 ± 0%      1.00 ± 0%        ~     (all equal)
Attrs/fastText_discard/5_args-8       1.00 ± 0%      0.00       -100.00%  (p=0.008 n=5+5)
Attrs/fastText_discard/10_args-8      1.00 ± 0%      1.00 ± 0%        ~     (all equal)
Attrs/fastText_discard/40_args-8      1.00 ± 0%      1.00 ± 0%        ~     (all equal)
```

增加slices包

slices “utils”

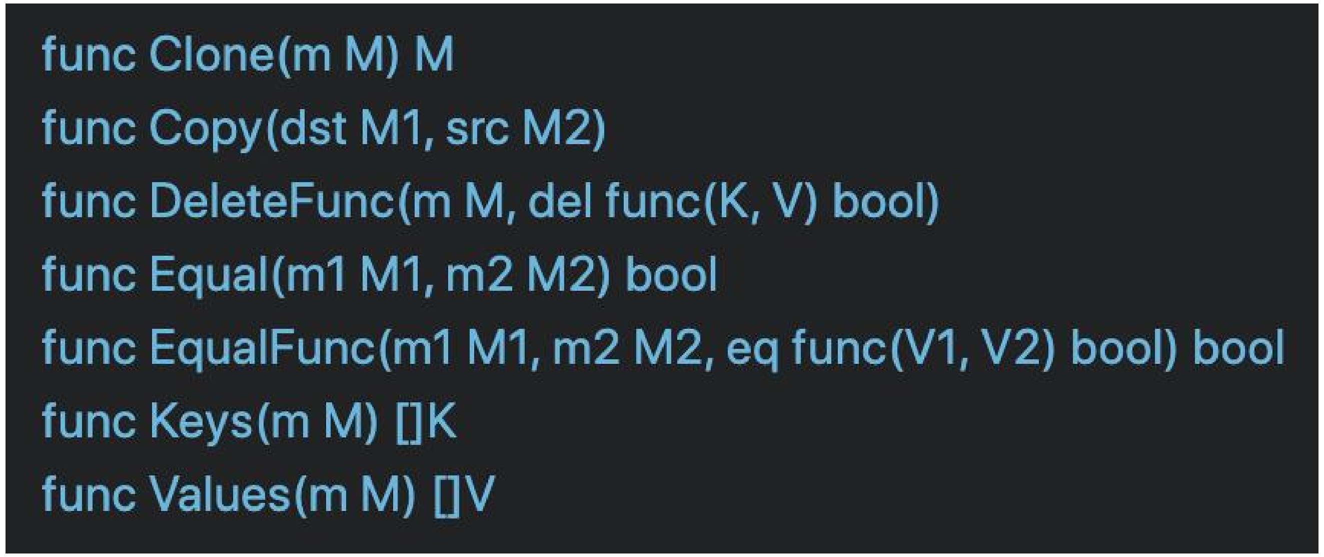


```
func BinarySearch(x []E, target E) (int, bool)
func BinarySearchFunc(x []E, target T, cmp func(E, T) int) (int, bool)
func Clip(s S) S
func Clone(s S) S
func Compact(s S) S
func CompactFunc(s S, eq func(E, E) bool) S
func Compare(s1, s2 []E) int
func CompareFunc(s1 []E1, s2 []E2, cmp func(E1, E2) int) int
func Contains(s []E, v E) bool
func ContainsFunc(s []E, f func(E) bool) bool
func Delete(s S, i, j int) S
func DeleteFunc(s S, del func(E) bool) S
func Equal(s1, s2 []E) bool
func EqualFunc(s1 []E1, s2 []E2, eq func(E1, E2) bool) bool
func Grow(s S, n int) S
func Index(s []E, v E) int
func IndexFunc(s []E, f func(E) bool) int
func Insert(s S, i int, v ...E) S
func IsSorted(x []E) bool
func IsSortedFunc(x []E, cmp func(a, b E) int) bool
func Max(x []E) E
func MaxFunc(x []E, cmp func(a, b E) int) E
func Min(x []E) E
func MinFunc(x []E, cmp func(a, b E) int) E
func Replace(s S, i, j int, v ...E) S
func Reverse(s []E)
func Sort(x []E)
func SortFunc(x []E, cmp func(a, b E) int)
func SortStableFunc(x []E, cmp func(a, b E) int)
```

https://github.com/golang/go/issues/57433

增加maps包

map “utils”



```
func Clone(m M) M
func Copy(dst M1, src M2)
func DeleteFunc(m M, del func(K, V) bool)
func Equal(m1 M1, m2 M2) bool
func EqualFunc(m1 M1, m2 M2, eq func(V1, V2) bool) bool
func Keys(m M) []K
func Values(m M) []V
```

https://github.com/golang/go/issues/57436

增加cmp包

```
type Ordered interface {
        ~int | ~int8 | ~int16 | ~int32 | ~int64 |
                ~uint | ~uint8 | ~uint16 | ~uint32 | ~uint64 | ~uintptr |
                ~float32 | ~float64 |
                ~string
}

func Less[T Ordered](x, y T) bool
func Compare[T Ordered](x, y T) int
```

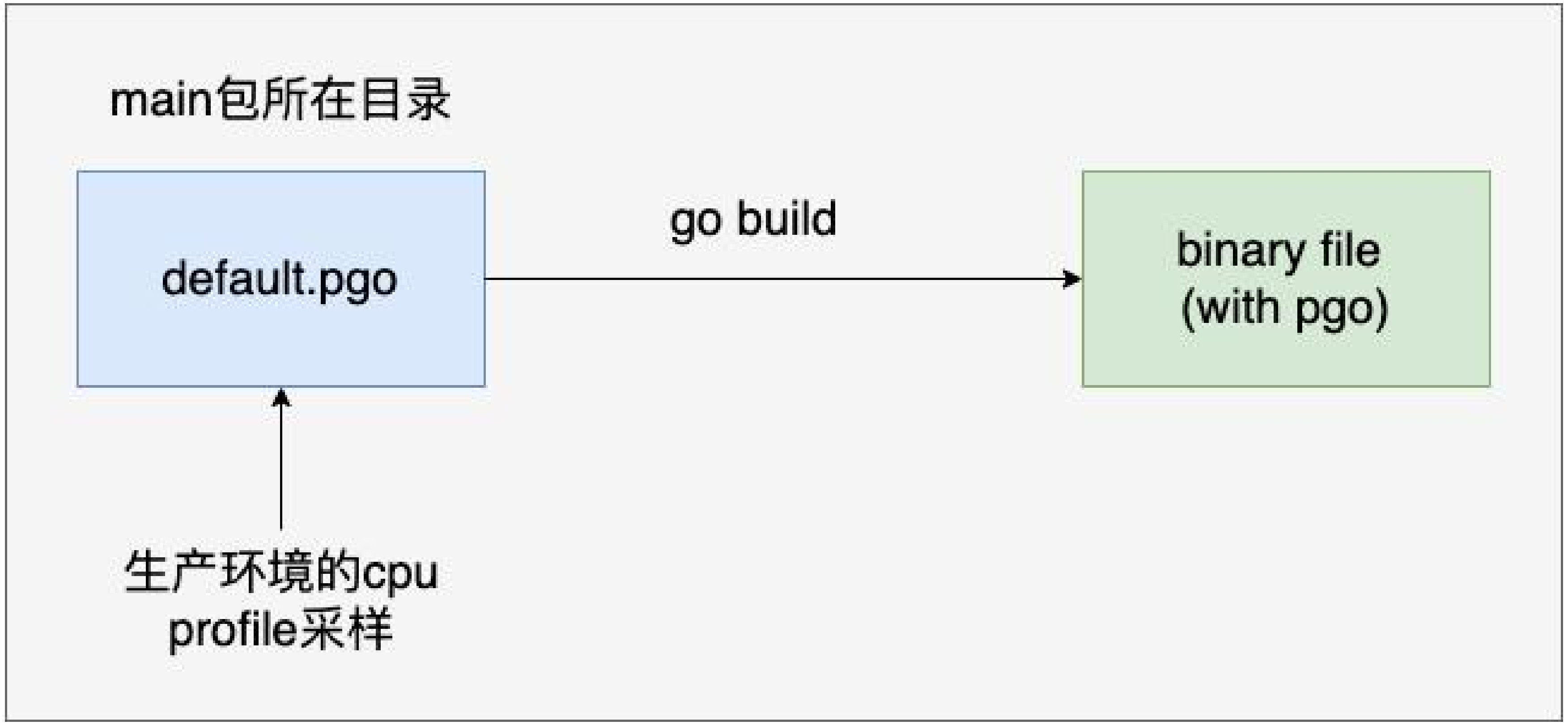
https://github.com/golang/go/issues/59488

第四部分

Compiler

Profile-guided optimization默认开启

Go 1.21：默认 - pgo=auto



第五部分

Tools

Go version

First release version change to Go 1.N.0

Before go 1.21

- Go lang version: Go 1.N
- Go release family: Go 1.N
- First release of the release family: Go 1.N

Start from go 1.21

- Go lang version: Go 1.N
- Go release family: Go 1.N
- First release of the release family: **Go 1.N.0**
  - e.g. Go 1.21.0

明确go.mod/go.work中go line的含义

```
// go.mod

module demo

go 1.18
```

对go line的理解	正确与否	备注
设置可用于构建module代码的Go的最小版本	no	任何版本的go compiler都可以去构建
它设定了要使用的Go compiler的确切版本。	no	可以使用本地安装的任何版本go compiler去构建
用于确定编译器在编译一个特定的源文件时使用的Go语言版本	Yes	当编译该module时，编译器要保证提供go line指示的版本的所有Go语义

问题：当本地Go编译器无法提供go line指示版本的所有语义时。
比如：Go 1.17编译器去编译使用泛型的go 1.18 demo module时？

GOTOOLCHAIN和toolchain

- GOTOOLCHAIN=local
  - 使用本地安装的Go工具链编译（与Go 1.21之前保持一致）
    - 如果本地Go工具链版本 < go version(in go.mod)，拒绝编译。

- GOTOOLCHAIN=auto(默认)
  - 如果本地Go工具链版本 > go version(in go.mod)，使用本地工具链编译
  - 如果本地Go工具链版本 < go version(in go.mod)，下载与go version(in go.mod)相同的Go工具链版本，然后使用新下载的工具链进行编译
  - 如果go.mod中新增toolchain并指定了版本，那么下载toolchain指示版本的go工具链，然后使用新下载的工具链进行编译

https://go.googlesource.com/proposal/+/master/design/57001-gotoolchain.md

Thanks