Golang 1.8 Release

Kent Huang @ Umbo CV

Umbots.

Agenda

- Changes since Go 1.7:
  - The Language
  - The Standard Library
  - The Runtime
  - The Tooling

Change to the language

Conversion rules

```go
type Person struct {
    Name     string
    AgeYears int
    SSN      int
}
```

```go
var aux struct {
    Name     string `json:"full_name"`
    AgeYears int    `json:"age"`
    SSN      int    `json:"social_security"`
}
```

How many times have you found yourself with two types that were almost equal?

In order to convert aux to type Person you needed to do

```
return Person{
    Name:     aux.Name,
    AgeYears: aux.AgeYears,
    SSN:      aux.SSN
}
```

Since Go 1.8 you can simply do

```
return Person(aux)
```

- Both types still need to have:
  - same sequence of fields (the order matters)
  - corresponding fields with same type.

Changes to the standard library

Sorting

- Given a slice of Person

```
var p []Person
```

- Print the slice sorted by name, age, and SSN

```
sort.Sort(byName(p))
sort.Sort(byAge(p))
sort(bySSN(p))
```

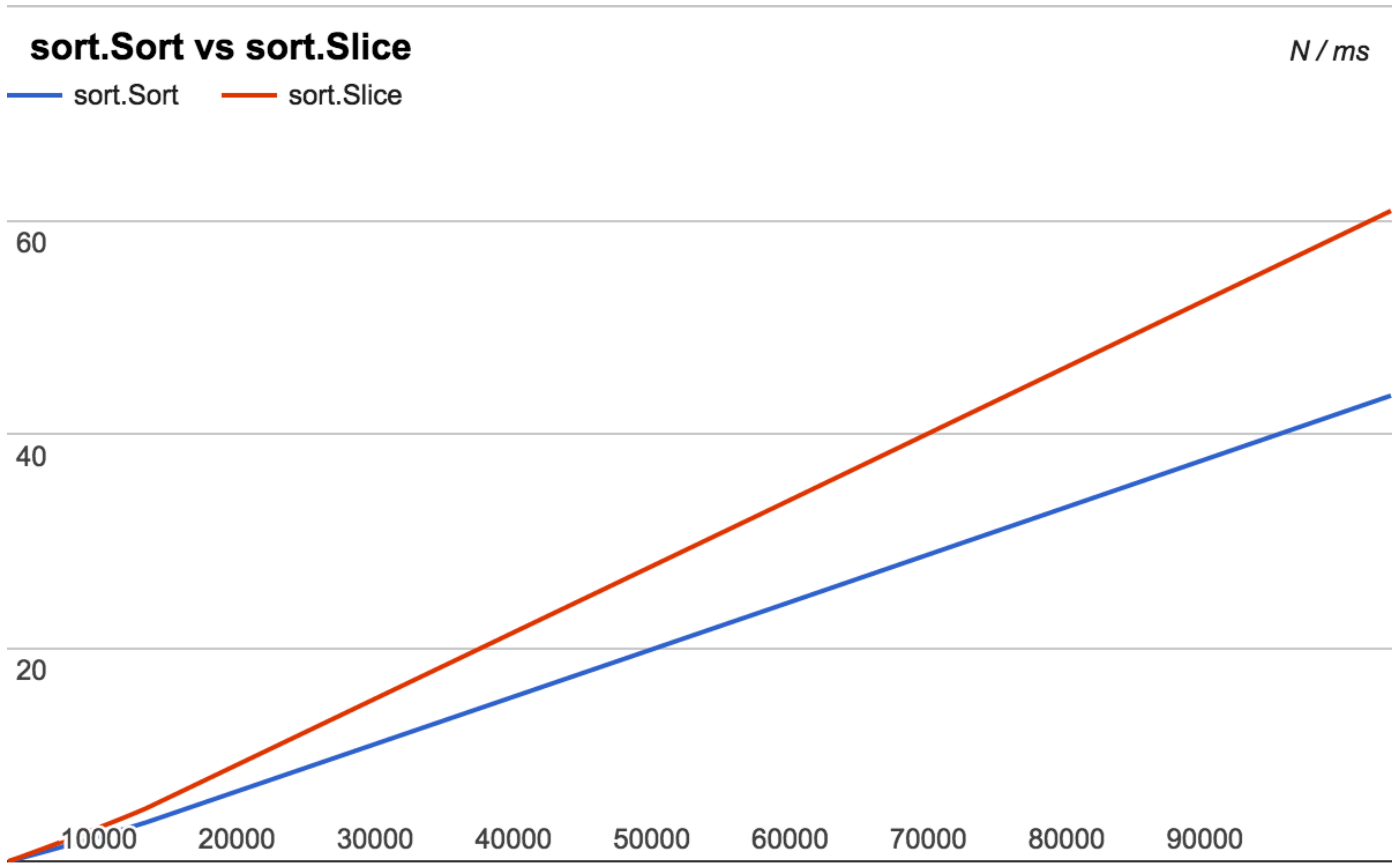
Still need

```
type byName []Person

func (b byName) Len() int                { return len(b) }
func (b byName) Less(i, j int) bool { return b[i].Name < b[j].Name }
func (b byName) Swap(i, j int)      { b[i], b[j] = b[j], b[i] }

type byAge []Person

func (b byAge) Len() int                { return len(b) }
func (b byAge) Less(i, j int) bool { return b[i].AgeYears <
b[j].AgeYears }
func (b byAge) Swap(i, j int)      { b[i], b[j] = b[j], b[i] }

type bySSN []Person

func (b bySSN) Len() int                { return len(b) }
func (b bySSN) Less(i, j int) bool { return b[i].SSN < b[j].SSN }
func (b bySSN) Swap(i, j int)      { b[i], b[j] = b[j], b[i] }
```

sort.Slice

- Since Go 1.8 can simply write this

```
sort.Slice(p, func(i, j int) bool { return p[i].Name < p[j].Name })
sort.Slice(p, func(i, j int) bool { return p[i].AgeYears <
p[j].AgeYears })
sort.Slice(p, func(i, j int) bool { return p[i].SSN < p[j].SSN })
```


sort.Sort vs sort.Slice
N / ms
sort.Sort
sort.Slice
60
40
20
10000
20000
30000
40000
50000
60000
70000
80000
90000

Benchmark

HTTP shutdown

- Added Shutdown method to http.Server.

```
// subscribe to SIGINT signals
quit := make(chan os.Signal)
signal.Notify(quit, os.Interrupt)

srv := &http.Server{Addr: ":8080", Handler: http.DefaultServeMux}
go func() {
    <-quit
    log.Println("Shutting down server...")
    if err := srv.Shutdown(context.Background()); err != nil {
        log.Fatalf("could not shutdown: %v", err)
    }
}()
```

```go
    http.HandleFunc("/", handler)
    err := srv.ListenAndServe()
    if err != http.ErrServerClosed {
        log.Fatalf("listen: %s\n", err)
    }
    log.Println("Server gracefully stopped")
```

HTTP/2

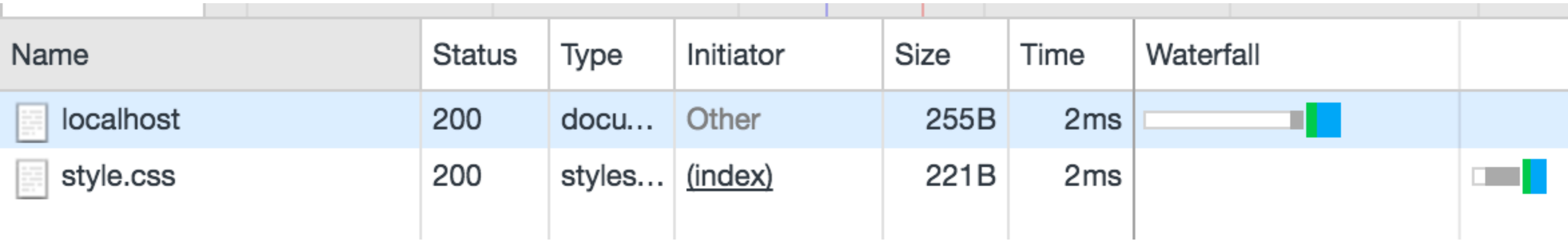
- http.Response now satisfies the http.Pusher interface.

```
type Pusher interface {
    Push(target string, opts *PushOptions) error
}
```

```go
func rootHandler(w http.ResponseWriter, r *http.Request) {
    if p, ok := w.(http.Pusher); ok {
        err := p.Push("/style.css", nil)
        if err != nil {
            log.Printf("could not push: %v", err)
        }
    }

    fmt.Fprintln(w, html)
}
```

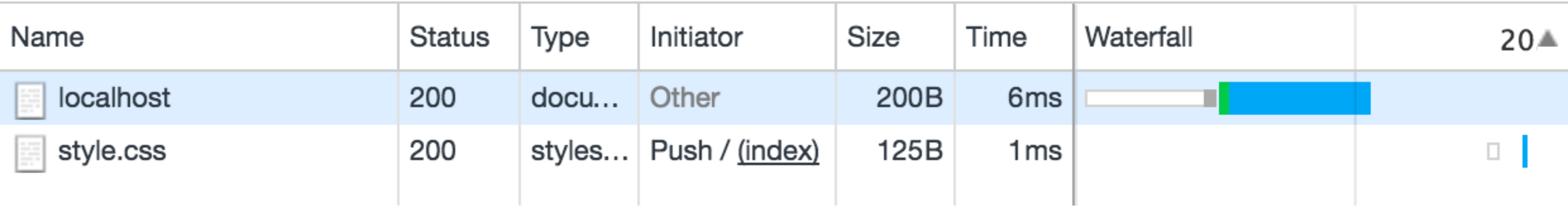
```go
func main() {
    http.HandleFunc("/", rootHandler)
    http.HandleFunc("/style.css", cssHandler)

    go func() {
        log.Fatal(http.ListenAndServeTLS("127.0.0.1:8081", cert, key,
nil))
    }()
    log.Fatal(http.ListenAndServe("127.0.0.1:8080", nil))
}
```

HTTP



Name	Status	Type	Initiator	Size	Time	Waterfall
localhost	200	docu…	Other	255B	2ms	
style.css	200	styles…	(index)	221B	2ms	

HTTP/2



Name	Status	Type	Initiator	Size	Time	Waterfall	20▲
localhost	200	docu…	Other	200B	6ms		
style.css	200	styles…	Push / (index)	125B	1ms		

Change to the runtime

Detection of concurrent map accesses

```go
    const workers = 100 // what if we have 1, 2, 25?

    var wg sync.WaitGroup
    wg.Add(workers)
    m := map[int]int{}
    for i := 1; i <= workers; i++ {
        go func(i int) {
            for j := 0; j < i; j++ {
                m[i]++
            }
            wg.Done()
        }(i)
    }
    wg.Wait()
```

Mutex Contention Profiling

- Profile your benchmarks and the contention on your mutexes.
    - `go test bench=. -mutexprofile=mutex.out`
- Alternatively, activate contention profiling with this new method.
    - `runtime.SetMutexProfileFraction`
- Note: For now sync.RWMutex is not profiled.

Ports to other platforms

Ports to other platforms

- 32-bit MIPS
  - big-endian (linux/mips)
  - little-endian (linux/mipsle) - requires Floating Point Unit
- Go on DragonFly BSD now requires DragonFly 4.4.4+.
- Go on OpenBSD now requires OpenBSD 5.9+.
- Plan 9 is now better!

Ports to other platforms

- Go 1.8 supports OS X 10.8+. Likely last time we support 10.8.
- ARM:
  - Go 1.8 is the last version to support ARMv5E and ARMv6 processors.
  - Go 1.9 will require ARMv6K. Will it work on my platform?
    - go tool dist -check-armv6k

Tools

Tools

- Default GOPATH
  - When GOPATH is not defined, the tool will use:
    - $HOME/go on Unix
    - %USERPROFILE%\go on Windows

- go bug
    - The new “go bug” command starts a bug report on GitHub, prefilled with information about the current system.

Have fun with Golang 1.8