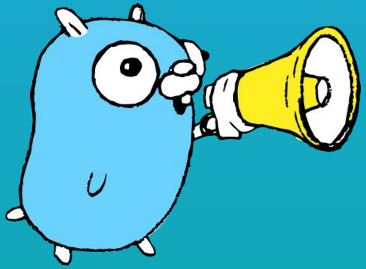
CodeConf HK, Nov 24 2018

Golang Pro Tips

Eddie Chan

@maded2



GolangHK

01

Quick Introduction

TENETS OF GO DESIGN

Simplicity.

Each language feature should be easy to understand.

TENETS OF GO DESIGN

Orthogonality.

Go's features should interact in predictable and consistent ways.

TENETS OF GO DESIGN

Internet Age.

- Go is a modern, general purpose language
- Open Source* (BSD-style license)
- Compiles to native machine code
- Compact and Lightweight syntax
- Rich standard Toolchain and Libraries
- Designed for the Cloud
- Designed for Teams

**github.com/golang/go*

02

Language Features

Types & Functions

```
type Radius float64

func (r Radius) Area() float64 {
    return 3.14 * r * r
}

func (r Radius) Circumference() float64 {
    return 2 * 3.14 * r
}
```

Basic types: bool, string, int, int8, int16, int32, int64, uint, uint8, uint16, uint32, uint64, uintptr, float32, float64, complex64, complex128, byte, rune

Types & Functions

```
type Car struct {
    Wheels     int
    Doors      int
    Colour     string
    Running    bool
}

func (c *Car) TurnLeft() {}
func (c *Car) TurnRight() {}
func (c *Car) Stop() {}
func (c *Car) MoveForward() {}
func (c *Car) Reverse() {}
```

Default Initialisation

```
c := Car{
    Wheels: 4,
    Doors: 2,
    Colour: “red”,
}
var r Radius                   0.0
var s string                   “”
var i int                      0
Var aList []int                nil
Var aMap map[string]int        nil
```

Tip #1 - Default initialisation value is your friend

```
    * map & slice is like a pointer
```

Interfaces

```
Package action                // package/module

interface Vehicle {           // duck typing
    func TurnLeft()
    func TurnRight()
    func Stop()
    func MoveForward()
    func Reverse()
}
```

Tip #2 – Compile Time checking of interface contract

```
    var _ Vehicle = (*Car)(nil)
```

Tip #3 – All interfaces should be at the root package

03

Runtime Features

Go-routines

```
go aFunc()
```

- Golang runtime has a built-in scheduler, uses minimal OS threads
- Cost 4kb for each go-routine
- Runtime can manages 100k+ go-routines

Tip #4 - Allocate enough OS threads for your app

```
runtime.GOMAXPROCS(runtime.NumCPU() * 2)
```

Memory management

- Golang is a garbage collected language
- Golang runtime allocates memory either on stack or heap
- Max <500µs STW GC pause
- GOGC env variable / SetGCPercent() controls the Garbage Collector

Tip #5 – Turn off GC if your app needs the extra performance and don’t care about memory usage

```
GOGC=off / SetGCPercent(-1)
```

Runtime Metrics

- Golang has a built-in metrics collection library
- Metrics can be query using http/json from a running Golang app:
  http://127.0.0.1:8080/debug/vars

- System metrics is also published using the same mechanism
- Custom application specific metrics can be added

Tip #6 - Use the built-in metrics library

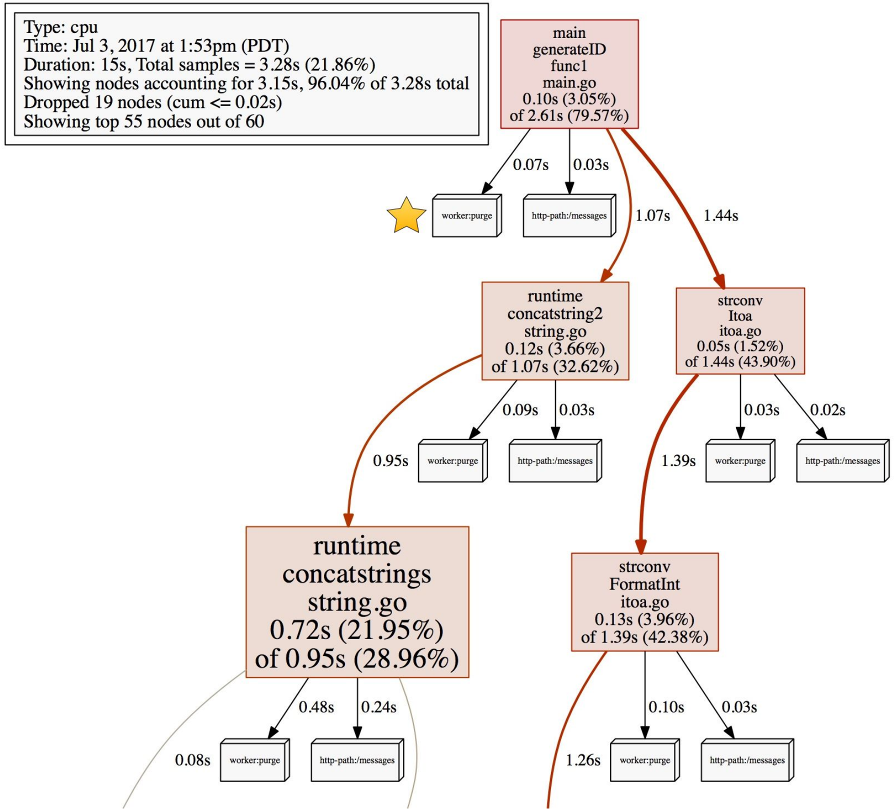
```
import _ "expvar"
```

Runtime Profiling

- Golang runtime has built-in profiling support
- Both live and offline profiling is available
  - Memory profiling (live & offline)
  - CPU Profiling (live & offline)
  - Go-routine blocking (live)
  - Execution Stack (live)
  - Mutex Profiling (live)
- Profiling visualisation tool is part of the toolchain
- Live Profiling can be query using http/json from a running Golang app:

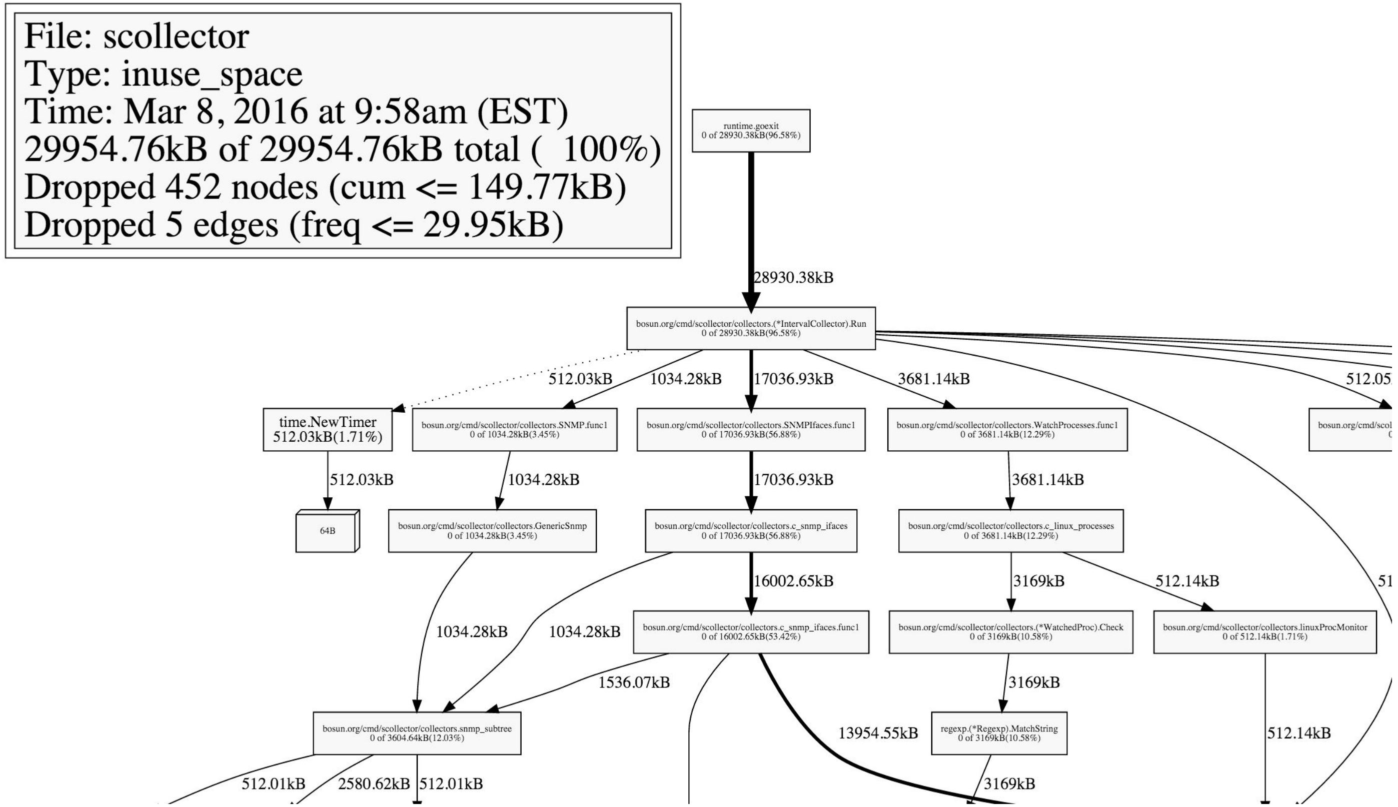
  http://127.0.0.1:8080/debug/pprof

Runtime Profiling


Type: cpu
Time: Jul 3, 2017 at 1:53pm (PDT)
Duration: 15s, Total samples = 3.28s (21.86%)
Showing nodes accounting for 3.15s, 96.04% of 3.28s total
Dropped 19 nodes (cum <= 0.02s)
Showing top 55 nodes out of 60

Example CPU Profiling visualisation (offline)

Runtime Profiling

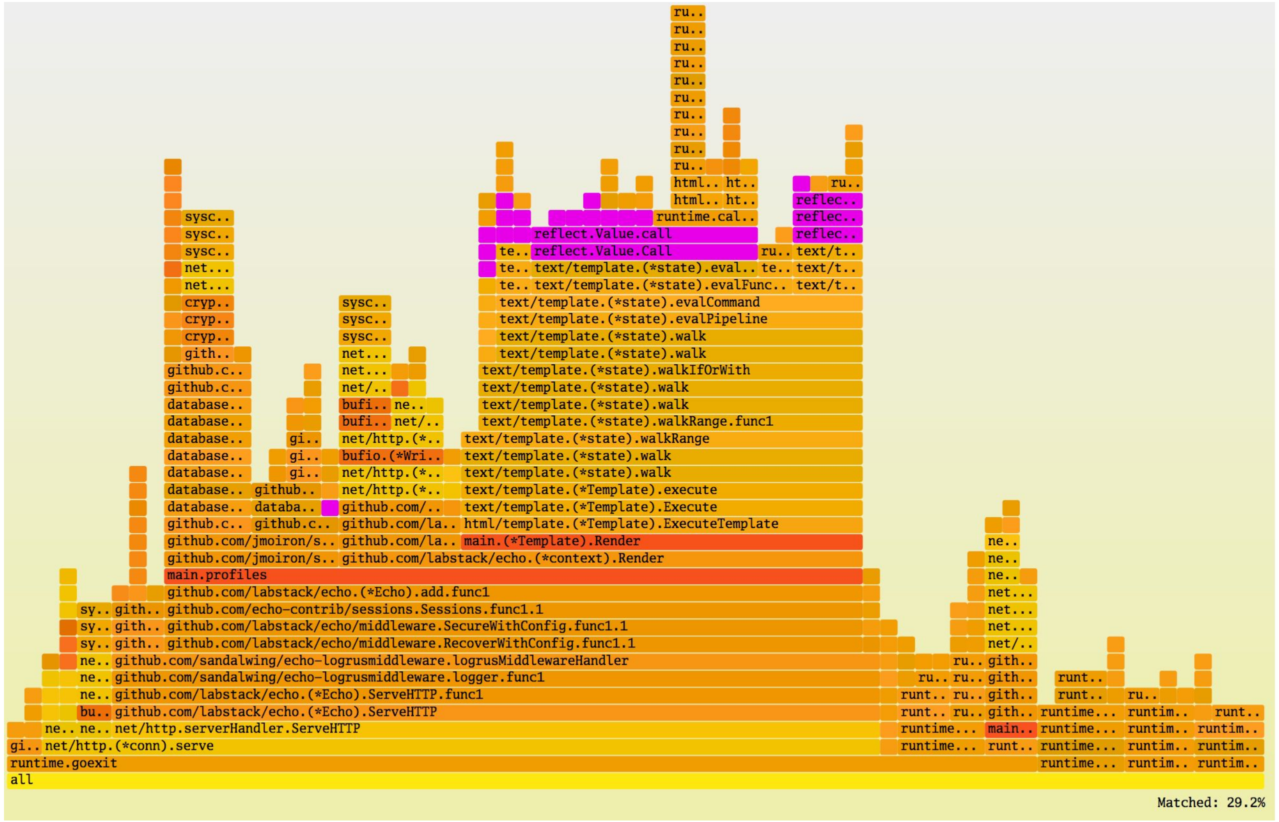


```
File: scollector
Type: inuse_space
Time: Mar 8, 2016 at 9:58am (EST)
29954.76kB of 29954.76kB total (  100%)
Dropped 452 nodes (cum <= 149.77kB)
Dropped 5 edges (freq <= 29.95kB)
```

runtime.goexit
0 of 28930.38kB(96.58%)
28930.38kB
bosun.org/cmd/scollector/collectors.(*IntervalCollector).Run
0 of 28930.38kB(96.58%)
512.03kB
1034.28kB
17036.93kB
3681.14kB
512.05
time.NewTimer
512.03kB(1.71%)
bosun.org/cmd/scollector/collectors.SNMP.func1
0 of 1034.28kB(3.45%)
bosun.org/cmd/scollector/collectors.SNMPIfaces.func1
0 of 17036.93kB(56.88%)
bosun.org/cmd/scollector/collectors.WatchProcesses.func1
0 of 3681.14kB(12.29%)
bosun.org/cmd/scol
512.03kB
1034.28kB
17036.93kB
3681.14kB
64B
bosun.org/cmd/scollector/collectors.GenericSnmp
0 of 1034.28kB(3.45%)
bosun.org/cmd/scollector/collectors.c_snmp_ifaces
0 of 17036.93kB(56.88%)
bosun.org/cmd/scollector/collectors.c_linux_processes
0 of 3681.14kB(12.29%)
16002.65kB
3169kB
512.14kB
51
1034.28kB
1034.28kB
bosun.org/cmd/scollector/collectors.c_snmp_ifaces.func1
0 of 16002.65kB(53.42%)
bosun.org/cmd/scollector/collectors.(*WatchedProc).Check
0 of 3169kB(10.58%)
bosun.org/cmd/scollector/collectors.linuxProcMonitor
0 of 512.14kB(1.71%)
1536.07kB
3169kB
13954.55kB
512.14kB
bosun.org/cmd/scollector/collectors.snmp_subtree
0 of 3604.64kB(12.03%)
regexp.(*Regexp).MatchString
0 of 3169kB(10.58%)
512.01kB
2580.62kB
512.01kB
3169kB

Example Memory Profiling visualisation (offline)

Runtime Profiling



Example Execution Profiling visualisation (live)

Runtime Profiling

Tip #7 - Use the built-in profiling

- Add trigger into your app to save profiling dumps
  - SIGTERM is a good trigger mechanism

- To save CPU profiling dump use
  `pprof.StartCPUProfile() / pprof.StopCPUProfile()`

- To save Memory profiling dump use
  `pprof.WriteHeapProfile()`

04

Microservices API Tips

Built-in Web Server

```
http.ListenAndServe(":8080", handler)
```

- Web Server is part of the standard library
- Uses Go-routine to handle all incoming requests, no callback
- Strong support for encryptions and ciphers
- Context package to aid chaining of webservice calls

Tip #8 - Collocate your webservice into a single process

JSON encoding support

- Strong support for message encoding/decoding
  - Xml, json, base64, csv, Protobuf

```
type Car struct {
    Wheels    int       `json:"wheels";db:"car_wheel"`
    Doors     int       `json:"doors";db:"car_doors"`
    Colour    string    `json:"colour";db:"car_colour"`
    Running   bool      `json:"running";db:"car_running"`
}
```

Tip #9 – Golang use struct tags pattern to give hints to codec; providing hints to multiple codec so that we reuse the same struct for different operations

05

Architecture Implications

Architecture

Tip #10 - Rethink your software architecture

- Go-routines allow developer to rethink how to model your application - isolate your data from different thread of execution
  - 1 go-routine per user
  - 1 go-routine per account
  - 1 go-routine per product

06

Q & A