链滴

golang pprof 调试 goroutine

作者：happyue

原文链接：https://ld246.com/article/1585904405240

来源网站：链滴

一 给项目加上pprof

1 简单的http服务直接使用 _ "net/http/pprof"

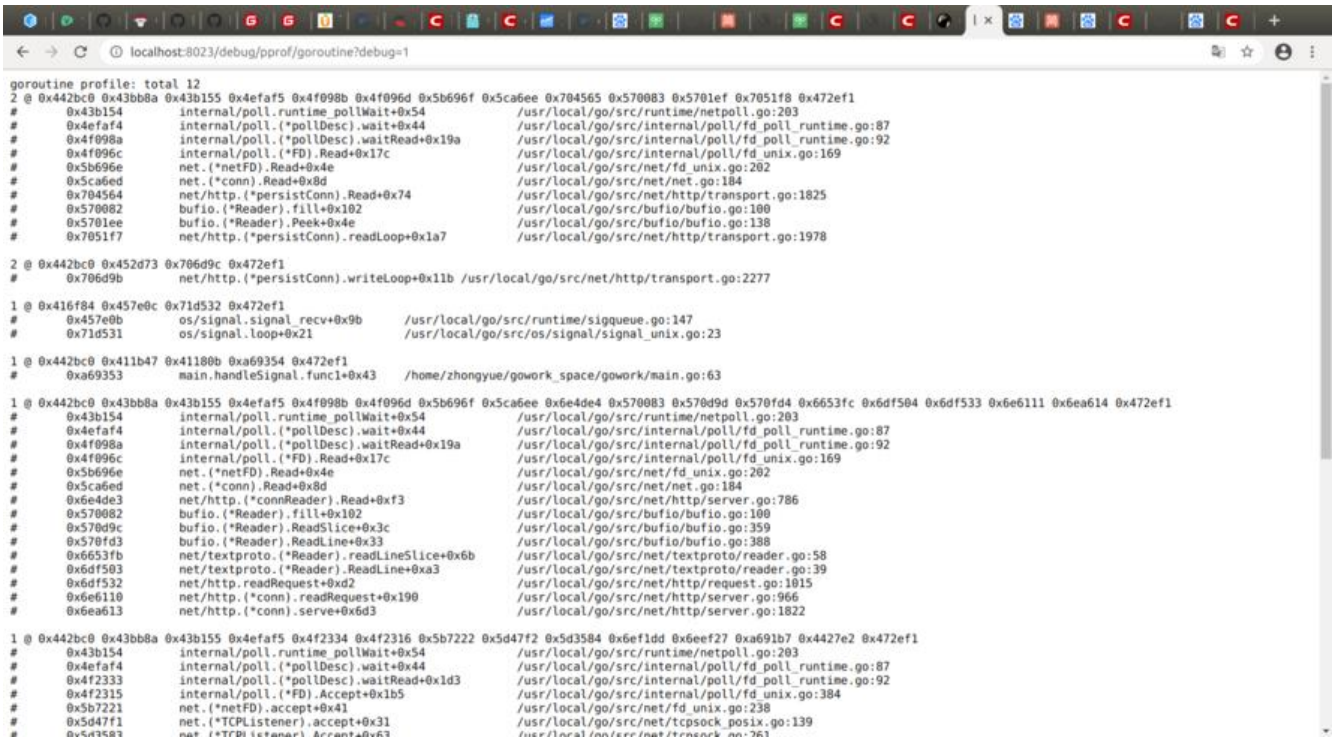
```
package main

import (
    // 略
     "net/http/pprof" // 会自动注册 handler 到 http server，方便通过 http 接口获取程序运行采
报告
    // 略
)

func main() {
    // 略

    runtime.GOMAXPROCS(1) // 限制 CPU 使用数，避免过载
    runtime.SetMutexProfileFraction(1) // 开启对锁调用的跟踪
    runtime.SetBlockProfileRate(1) // 开启对阻塞操作的跟踪

    go func() {
        // 启动一个 http server，注意 pprof 相关的 handler 已经自动注册过了
        if err := http.ListenAndServe(":8023", nil); err != nil {
            log.Fatal(err)
        }
        os.Exit(0)
    }()

    // 略
}
```

2 gowork项目使用gin框架的，pprof使用 "github.com/DeanThompson/ginpprof"

```
package main

import (
    "github.com/gin-gonic/gin"

    "github.com/DeanThompson/ginpprof"
)

func main() {
    router := gin.Default()

    router.GET("/ping", func(c *gin.Context) {
        c.String(200, "pong")
    })

    // automatically add routers for net/http/pprof
    // e.g. /debug/pprof, /debug/pprof/heap, etc.
    ginpprof.Wrapper(router)

    router.Run(":8023")
}
```

原文链接：golang pprof 调试 goroutine

2 运行项目，可以在http://localhost:8023/debug/pprof/goroutine?debug=1看到输出的信息。



可以看到，刚开始协程total：12， 然后做一些引起开协程并结束动作，看看协程的状态：

这时协程已经失控了。

我们使用命令行go自带的工具命令

go tool pprof http://localhost:8023/debug/pprof/goroutine会返回一个交互对话框，

这里使用 top， list ×××， web 命令查看具体信息

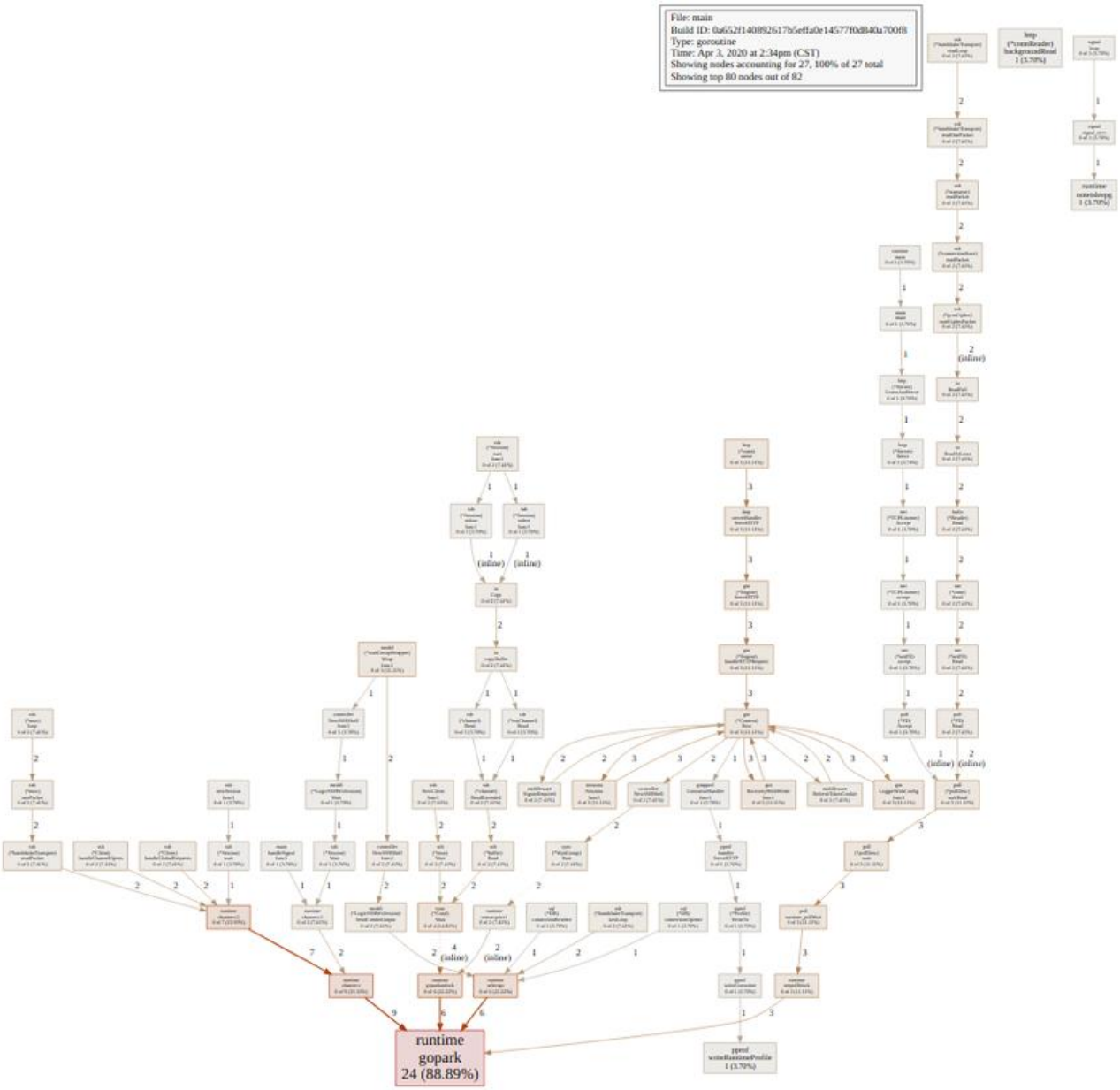
```
zhongyue@zhongyue ~/gowork_space/gowork  ‹master›
➜ go tool pprof http://localhost:8023/debug/pprof/goroutine
Fetching profile over HTTP from http://localhost:8023/debug/pprof/goroutine
Saved profile in /home/zhongyue/pprof/pprof.main.goroutine.001.pb.gz
File: main
Build ID: 0a652f140892617b5effa0e14577f0d840a700f8
Type: goroutine
Time: Apr 3, 2020 at 2:34pm (CST)
Entering interactive mode (type "help" for commands, "o" for options)
(pprof) top
Showing nodes accounting for 27, 100% of 27 total
Showing top 10 nodes out of 82
      flat  flat%   sum%        cum   cum%
        24 88.89% 88.89%         24 88.89%  runtime.gopark
         1  3.70% 92.59%          1  3.70%  net/http.(*connReader).backgroundRead
         1  3.70% 96.30%          1  3.70%  runtime.notetsleepg
         1  3.70%   100%          1  3.70%  runtime/pprof.writeRuntimeProfile
         0     0%   100%          2  7.41%  bufio.(*Reader).Read
         0     0%   100%          1  3.70%  database/sql.(*DB).connectionOpener
         0     0%   100%          1  3.70%  database/sql.(*DB).connectionResetter
         0     0%   100%          1  3.70%  github.com/DeanThompson/ginpprof.GoroutineHandler.func1
         0     0%   100%          3 11.11%  github.com/gin-contrib/sessions.Sessions.func1
         0     0%   100%          3 11.11%  github.com/gin-gonic/gin.(*Context).Next
```

list 具体某一条

```
(pprof) list Read
Total: 27
ROUTINE ======================== bufio.(*Reader).Read in /usr/local/go/src/bu
         0          2 (flat, cum)  7.41% of Total
         .          .    221:           }
         .          .    222:           // One read.
         .          .    223:           // Do not use b.fill, which will loop
         .          .    224:           b.r = 0
         .          .    225:           b.w = 0
         .          2    226:           n, b.err = b.rd.Read(b.buf)
         .          .    227:           if n < 0 {
         .          .    228:                   panic(errNegativeRead)
         .          .    229:           }
         .          .    230:           if n == 0 {
         .          .    231:                   return 0, b.readErr()
ROUTINE ======================== golang.org/x/crypto/ssh.(*buffer).Read in /h
         0          2 (flat, cum)  7.41% of Total
         .          .     89:           if b.closed {
         .          .     90:                   err = io.EOF
         .          .     91:                   break
         .          .     92:           }
         .          .     93:           // out of buffers, wait for producer
         .          2     94:           b.Cond.Wait()
```

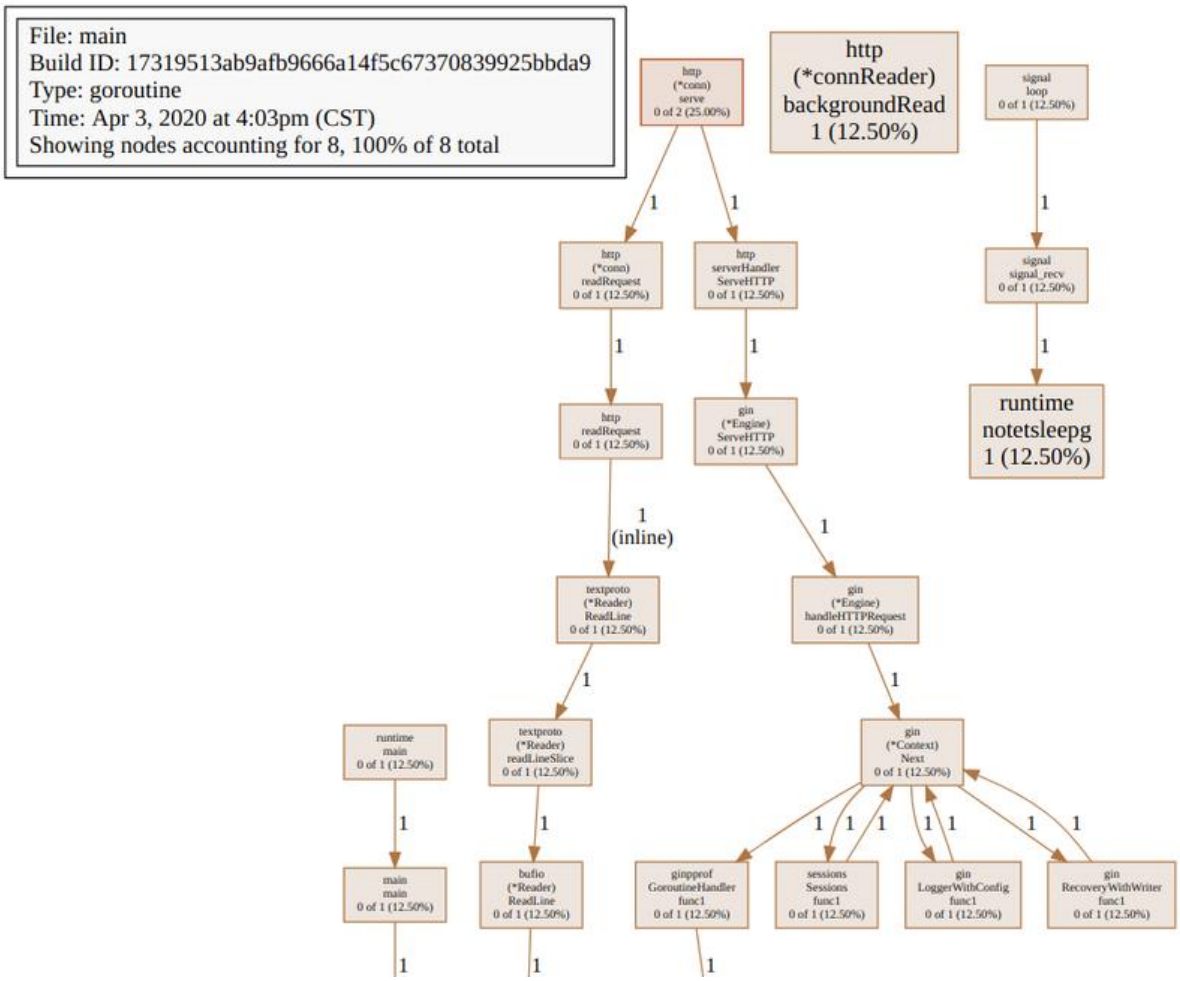
web命令，虽然这个命令的名字叫“web”，但它的实际行为是产生一个 .svg 文件，并调用你的系里设置的默认打开 .svg 的程序打开它。如果遇到 failed to execute dot. Is Graphviz installed? Error: exec: "dot": executable file not found in $PATH 错误，需要安装Graphviz ,ubuntu 下安装直接 su o apt-get install graphviz。好了，我在软件中做了一些ssh的操作，这些操作会引起后台开启协程在浏览器中会看到下图：

原文链接：golang pprof 调试 goroutine



这里图片不是很清晰，主要想说明pprof调试确定协程泄露的方法。解决掉协程泄露失控的问题后，下：

原文链接：golang pprof 调试 goroutine



感谢 https://blog.wolfogre.com/posts/go-ppof-practice

原文链接：golang pprof 调试 goroutine