pprof工具使用

|  |  |  |
| --- | --- | --- |
| 版本 | 内容 | 修改人 / 时间 |
| 1.0 | 创建 | 赵明明 / 2018.04.16 |
| 1.1 | 更新Go1.10对pprof的支持 | 赵明明 / 2018.04.20 |

1. 背景

Go是一门注重性能的语言，但是我们在写完代码之后，怎么才能查看我们写的代码在性能方面做得如何呢？此时Go中提供了内置的工具-pprof来帮助我们分析代码。

二、pprof能帮助我们做什么

pprof 主要能帮助我们分析代码的内存使用、cpu占用方面。

三、程序中如何使用pprof

在代码中，我们一共有两种方式使用pprof，

（1）、使用runtime/pprof

使用此包时，需要开发者指定一个文件路径，pprof将服务运行时采集的数据写入到此文件中。

想要获取cpu方面数据，使用

file,\_ := os.Create(“./cpuProfile”)

pprof.StartCPUProfile(file) 开始对程序进行cpu分析。

defer pprof.StartCPUProfile() 停止对程序进行分析

想要获取内存分析数据，使用

pprof.WriteHeapProfile(file)

注意：使用此种方式时，会消耗一定的cpu和内存。

（2）、使用 net/http/pprof

使用此包时，只需要在程序main包中，导入

import \_ “net/http/pprof”

在该包init初始化函数中，初始化了五个使用httpDefaultMux 注册的path 分别为：

debug/pprof 展示一个总的分析页面

debug/pprof/cmdline 展示分析的工具名称

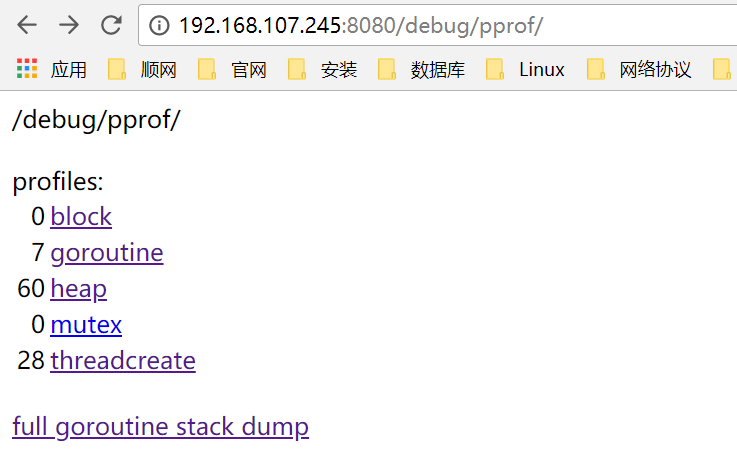
debug/pprof/profile 下载一个30秒的cpu文件，指定时间在路径后加入？seconds=10

debug/pprof/symbol 查找请求中列出的程序计数器

debug/pprof/trace 以二进制形式执行跟踪。默认时间为1s，指定时间同上。

如果程序中没有使用http监听，使用此包下，需要建立一个http的监听，如果使用了自定义的mux，则需要自己注册上面五个path(五个path对应的函数名，均为大写)

在debug/pprof路径下还有几个子页面



block 阻塞

goroutine goroutine创建数量，以及创建的位置

heap 内存

mutex 锁

threadcreate 线程创建。

full goroutine stack dump 完整的goroutine堆栈转储

（每个子页面对应着runtime/pprof内的一个函数，有兴趣的同志，可以看看，入口在net/http/pprof第227行。）

注：net/http/pprof， 其内部也是调用了runtime/pprof的方法。

四、使用go tool pprof命令，分析数据

准备：

使用go tool pprof命令前，先安装graphviz软件，用于生成系统图。

地址为： https://pan.baidu.com/s/12BlFJx8Vy8Ouscv1lhUN6A 密码：ogtm

并将bin目录，加到系统变量path中。

开始 ：

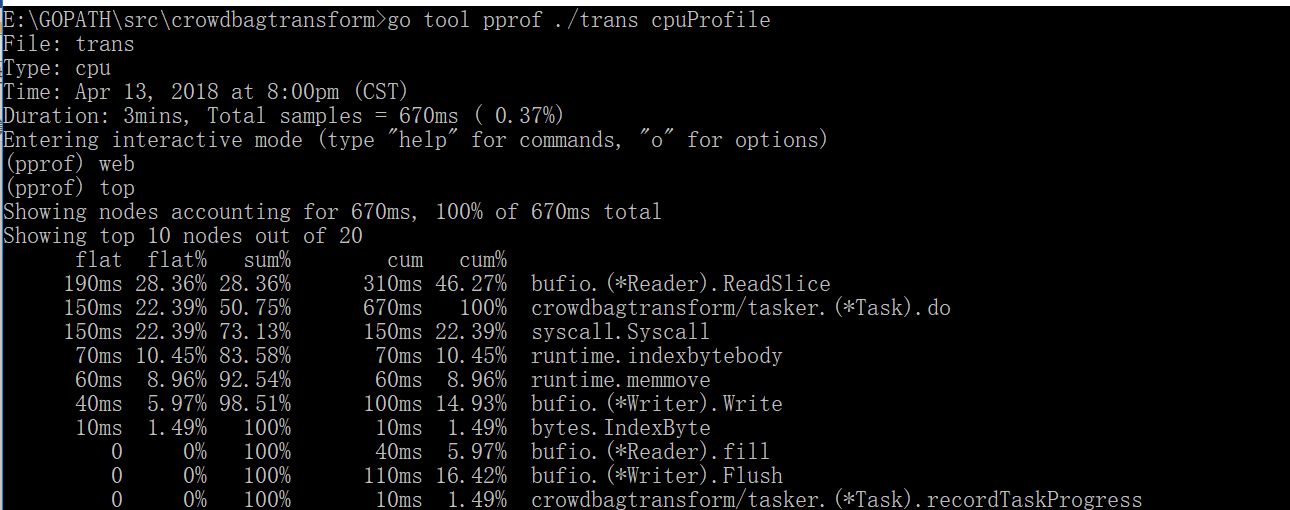
通过（3程序中使用pprof）以上方式获取的文件或是http地址，都可以通过 go tool pprof命令分析，这里介绍最简单的一种go tool pprof使用方式

go tool pprof [options] [binary] <source> ...

binary为二进制的应用文件。

source为分析文件的来源（可以是文件名称、可以是http地址）

cpu例如：



url地址为：http://you\_ip:your\_port/debug/pprof/profile

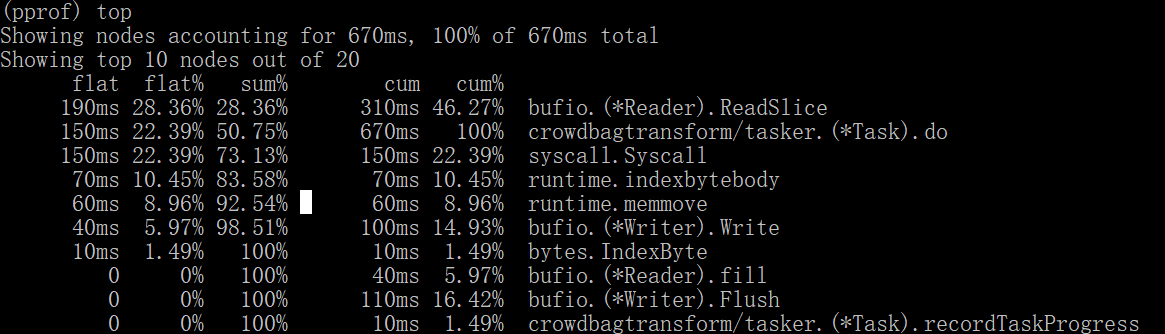
第一行表示二进制文件名称

第二行表示分析的类型（cpu、inuse\_space）

第三行为时间

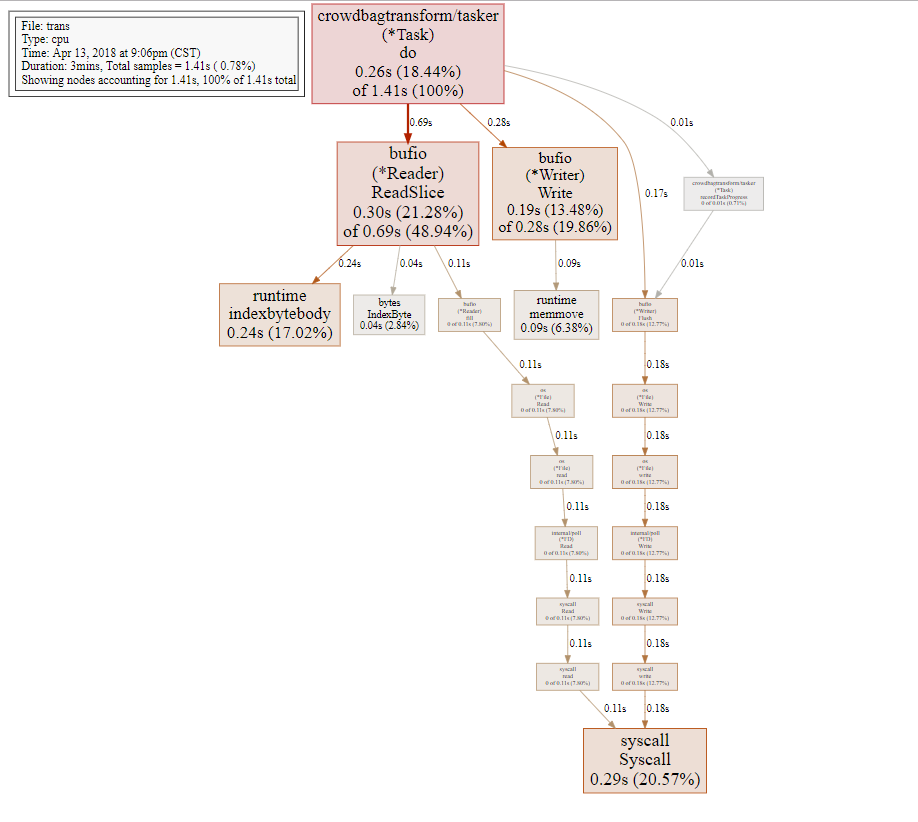
第四行表示此文件生成的时间，上图中 670ms表示，此文件采集数据的样本时间。 0.37%表示此670ms在总采集时间内所占的比例。

pprof中一个常用的命令为，topN，它列出了程序耗时的时间。



第一行为占用样本的时间，第二行为占用样本时间的百分比，第三行为前几列占用的总和，第四行、第五行表示函数以及函数的子函数运行所占用的时间和比例。

pprof还有一个web 命令，将上述信息，生成一个.svg文件，并在浏览器中显示，生成一个函数调用图



箭头表示函数调用关系，箭头上的数字表示被调用函数的时间。

web 参数后面还可加上函数名称，只打印此函数的调用关系图。

如果想打印更详细到代码级别，可以使用**list命令**，list命令后面必须跟着一个正则表达式，就能查看函数的代码以及每行代码的消耗。也可以使用**weblist命令**，将代码打印到web（同时显示源代码和汇编代码）。

其他的命令还有**gif、png、svg**生成对应格式的图片或者gif图。

内存使用情况与cpu使用相同，

url地址换为： http://you\_ip:you\_port/debug/pprof/heap

五、火焰图（Flame Graphs）

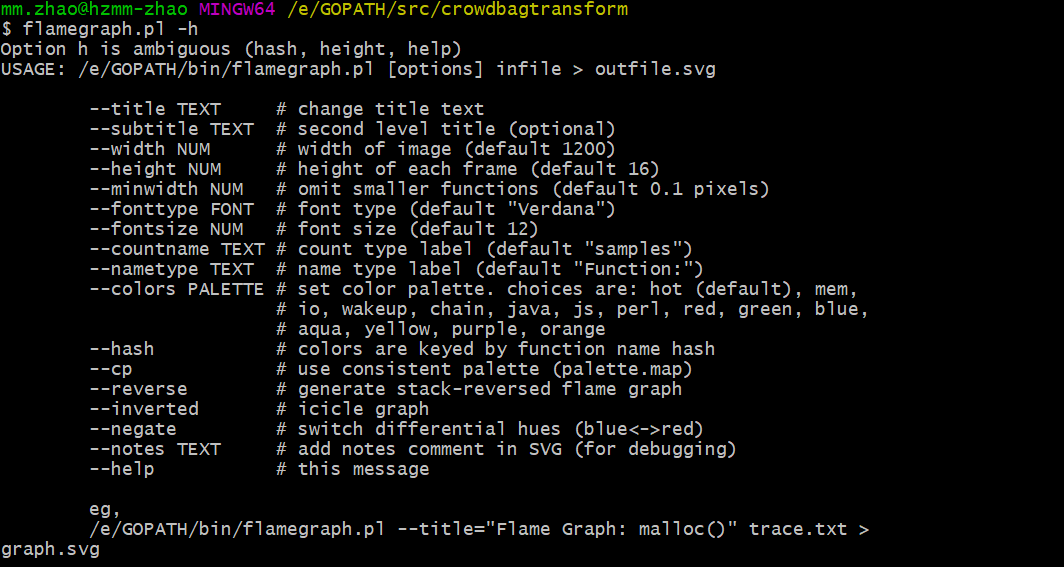
火焰图是Brendan D.Gregg发明的一种性能分析图表，火焰图的颜色并没有深层次的意义，只是作者当初制作时，使用了最温暖的颜色，所以看起来很像火焰。这里我们介绍的go的一种工具[go-torch,](https://github.com/uber/go-torch)这是uber开源的一个工具，可以直接读取go程序产生的性能分析文件，并生成一个.svg的文件，.svg可以直接使用浏览器打开。

安装

首先安装go-torch（github.com/uber/go-torch）直接使用go get命令即可。按照go-torch readme文档中所说，我们还需要安装FlameGraph脚本，flameGraph安装步骤如下：



下载完成后，进入到FlameGraph文件夹内，将flamegraph.pl文件copy到gopath内，或者是path内。cmd内输入flamegraph.pl –h。出现以下界面则为安装成功



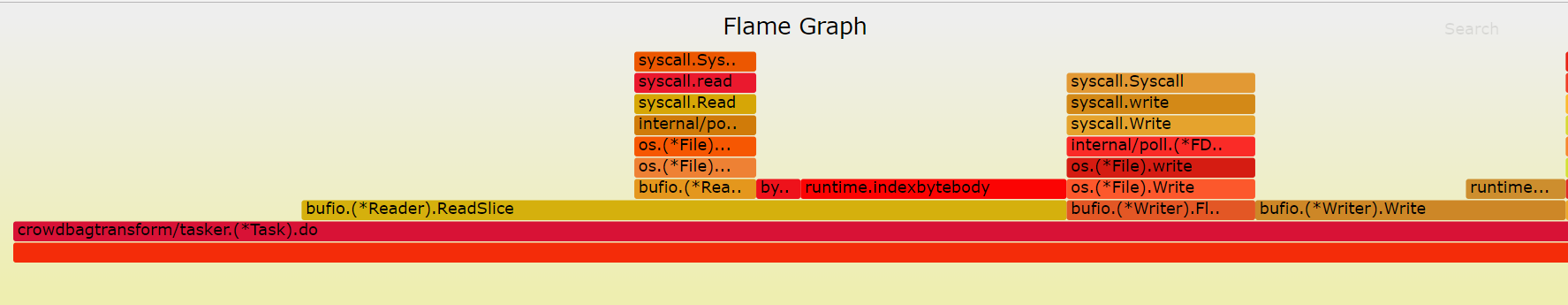
安装成功之后，我们就可以使用go-torch来生成火焰图了。

go-torch的使用方法与go tool pprof使用方法接近， 例如：

go tool pprof [options] [binary] <source> ...

go-torch [options] [binary] <source> ...

命令成功后，会在当前目录下生成一个.svg文件，直接双击打开，在浏览器中就会显示对此文件的火焰图。



火焰图的调用顺序是从下往上的，顺序代表了函数的调用关系。

go-torch 常用命令：

-u url 要访问的 URL，这里只是主机和端口部分

--seconds 修改http下载文件的秒数

更多命令请查看go-torch –h

六、Go1.10.1对pprof的支持

1. Go1.10.1对pprof的改变

主要增加web ui的功能，生成分析文件之后，使用以下命令

go tool pprof –http=:8090 profile

默认会打开浏览器，展示原在pprof下使用web命令显示的界面，此时看页面左

上角，view按钮点击后，有展示各种图示，但是没有火焰图的选项。

如果想在页面中使用火焰图，我们需要下载google编写的原生pprof工具，地址为github.com/google/pprof，编译安装到本地（go get）。下载完成后，使用命令pprof -http=:8090 profile（这个命令的体验感并不是很强，并不会默认打开浏览器，也可能是我配置错了），输入命令后，在浏览器中输入命令中监听的端口号，点击view，选择火焰图（Flame Graph）选项，我们的面前就能展示出一幅美丽火焰图。

七、错误集

（1）、使用go-torch时，报

Failed: could not generate flame graph: fork/exec E:\GOPATH\bin\flamegraph.pl: %1 is not a valid Win32 application

这个错误时，

按照 <https://github.com/uber/go-torch/issues/60>上的修改方式，修改完毕后，重新编译后，在使用go-torch命令。

参考网址

<https://golang.org/>

<https://tonybai.com/2018/02/17/some-changes-in-go-1-10/>

<https://blog.golang.org/profiling-go-programs>

<https://github.com/uber/go-torch>

<http://cizixs.com/2017/09/11/profiling-golang-program>