Go1.14 Timer Reset 导致进程假死

2020年3月29日

|  |
| --- |
| package main  import (  "encoding/json"  "fmt"  "net/http"  \_ "net/http/pprof"  "sync"  "time"  )  const PoolTimeout = time.Millisecond  var mutex = sync.Mutex{}  var counter int  var last time.Time  var timers = sync.Pool{  New: func() interface{} {  t := time.NewTimer(PoolTimeout)  t.Stop()  return t  },  }  func main() {  const bind = "54321"  fmt.Println("Bind port = ", bind)  http.HandleFunc("/VIID/Faces", callabck)  if err := http.ListenAndServe(":"+bind, nil); err != nil {  fmt.Println(err)  }  }  func callabck(w http.ResponseWriter, r \*http.Request) {  timer := timers.Get().(\*time.Timer)  timer.Reset(PoolTimeout)  var m map[string]interface{}  if err := json.NewDecoder(r.Body).Decode(&m); err != nil {  http.Error(w, "Unmarshal json failed", http.StatusBadRequest)  return  }  now := time.Now()  mutex.Lock()  if now.Unix()-last.Unix() > 0 {  counter++  last = now  fmt.Println(counter, " \t ", now.Format(time.RFC3339))  }  mutex.Unlock()  <-timer.C  timers.Put(timer)  w.WriteHeader(http.StatusOK)  } |

|  |
| --- |
| Example Code |

# 1、平台及Go版本

操作系统：Windows 7 64位

Go版本：go version go1.14.1 windows/amd64

似乎Go1.14才有这个问题，Go1.13（测试过Go1.13.9）没有这个问题。

# 2、问题复现步骤

（a）、执行go build main.go生成可执行程序main.exe，该进程监听本机54321端口；

（b）、向本机54321端口的/VIID/Faces接口发起大量POST请求；

（c）、待main.exe运行一段时间后（几分钟至几十分钟不等），该进程假死，进程内有一个线程独占1个CPU核心，如图1所示；该线程在执行runtime.checkTimers，如图2所示。

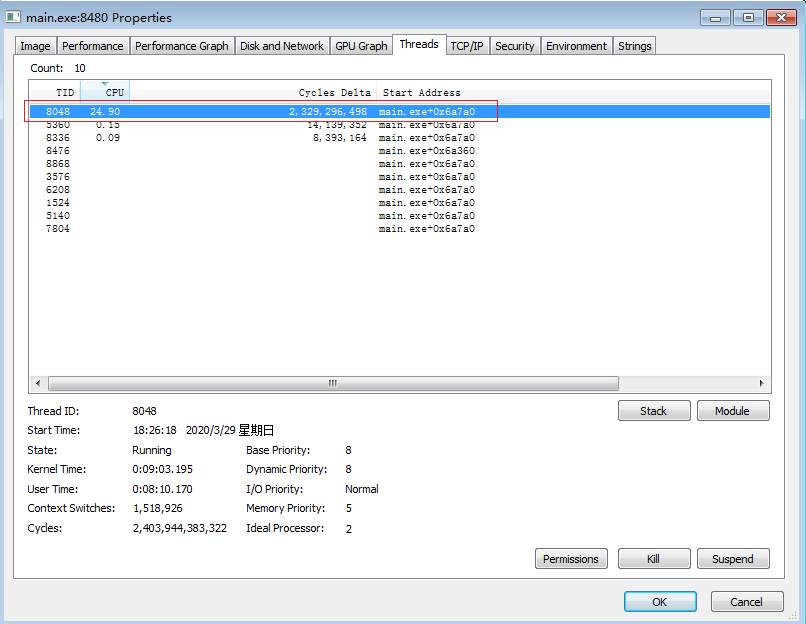


图1

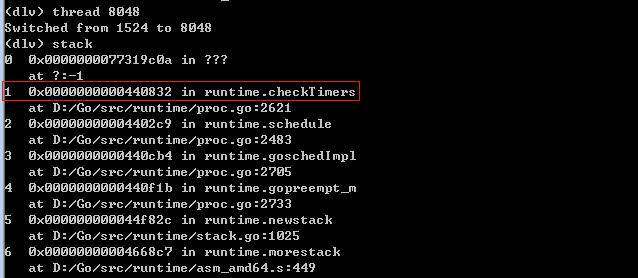


图2