Debugging concurrent programs in Go

F0SD3M
04022023
cat /etc/about_me

- From Ukraine :
- Gopher, OSS contributor
- father of 🥺 👶
- debuggers fan
- Speaker at many conferences

andrii@hachyderm.io
Concurrency Programming is Challenging!
Debugging concurrent programs is Hard!
8 stages of debugging

1. That can’t happen.
2. That doesn’t happen on my machine.
3. That shouldn’t happen.
4. Why does that happen?
5. Oh, I see 😐
6. How did that ever work? 🤣
7. Who wrote this 🍤 😛 😛
8. Oh wait, that was me. (From reddit)
Debugging sequential programs

dlvc test -- test.run TestFibonacciBig
(dlv) b main_test.go:6
Breakpoint 1 set at 0x115887f for github.com/andriisoldatenko/debug_test.TestFibonacciBig() ./main_test.go:6
(dlv) c
> github.com/andriisoldatenko/debug_test.TestFibonacciBig() ./main_test.go:6 (hits goroutine(17):1 total:1) (PC: 0x115887f)
    1:  package main
    2: 
    3: import "testing"
    4: 
    5: func TestFibonacciBig(t *testing.T) {
    6:     var want int64 = 55
    7:     got := FibonacciBig(10)
    8:     if got.Int64() != want {
    9:         t.Errorf("Invalid Fibonacci value for N: %d, got: %d, want: %d", 10, got.Int64(), want)
   10:     }
   11: }
(dlv)
Debugging concurrent programs

```go
package main

import (
    "fmt"
    "time"
)

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}

func main() {
    go say("world")
    say("hello")
}
```

@andrii@hachyderm.io
What is a goroutine?

```go
type g struct {
    stack              stack
    stackguard0        uintptr
    stackguard1        uintptr
    _panic             uintptr
    _defer             uintptr
    m                  uintptr
    sched              gobuf
    syscallsp          uintptr
    syscallpc          uintptr
    stktopsp           uintptr
    param              unsafe.Pointer
    atomicstatus       uintptr
    stackLock          uintptr
    goid               uintptr
    schedlink          uintptr
    waitsince          uintptr
    waitreason         uintptr
    preempt            bool
    preemptStop        bool
    preemptShrink      bool
    asyncSafePoint     bool
    panicfault         bool
    gcscandone         bool
    throwsplit         bool
    activeStackChans   bool
    raceignore         uintptr
    syscallblocktraced bool
    syscallitticks     uintptr
    traceseq           uintptr
    tracecallsp        uintptr
    lockedm            uintptr
    sig                 uintptr
    writebuf           []byte
    sigcode0           uintptr
    sigcode1           uintptr
    sigpc              uintptr
    gopc               uintptr
    ancestors          []*uint64
    startpc            uintptr
    racectx            uintptr
    waiting            uintptr
    cgoCtx             uintptr
    labels             unsafe.Pointer
    timer              *time.Timer
    selectDone         uintptr
    goAssistBytes      int64
}
```

[https://tpaschalis.github.io/goroutines-size/](https://tpaschalis.github.io/goroutines-size/)
Do you know why gorotines?
Question to audience!
How can I debug my concurrent Go program?
package main

import {
    "fmt"
    "time"
}

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}

func main() {
    go say("world")
    say("hello")
}

hello
world
world
hello
world
hello
world
hello
world
world
hello
hello
Developers which are using coloured logs

Developers which are using prints
```go
func main() {
    c := coloredgoroutine.Colors(os.Stdout)
    fmt.Fprintln(c, "Hi, I am go routine", goid.ID(), "from main routine")

    count := 10
    var wg sync.WaitGroup
    wg.Add(count)

    for i := 0; i < count; i++ {
        i := i
        go func() {
            fmt.Fprintln(c, "Hi, I am go routine", goid.ID(), "from loop i =", i)
            wg.Done()
        }()
    }
    wg.Wait()
}
```
visualize goroutines 🐞?

```
(base) → colored-goroutines git:(main) × go run main_colored.go
Hi, I am go routine 1 from main routine
Hi, I am go routine 15 from loop i = 9
Hi, I am go routine 12 from loop i = 6
Hi, I am go routine 13 from loop i = 7
Hi, I am go routine 14 from loop i = 8
Hi, I am go routine 8 from loop i = 2
Hi, I am go routine 11 from loop i = 5
Hi, I am go routine 10 from loop i = 4
Hi, I am go routine 9 from loop i = 3
Hi, I am go routine 6 from loop i = 0
Hi, I am go routine 7 from loop i = 1
```

https://github.com/xiegeo/coloredgoroutine
visualize goroutines

https://divan.dev/posts/go_concurrency_visualize/
scheduling events
Print scheduling events?

$ GODEBUG=schedtrace=5000 <binary>

SCHED 0ms: gomaxprocs=28 idleprocs=25 threads=4 spinningthreads=1 idlethreads=0 read
# command-line-arguments
SCHED 0ms: gomaxprocs=28 idleprocs=25 threads=5 spinningthreads=1 idlethreads=0 read
# command-line-arguments
SCHED 0ms: gomaxprocs=28 idleprocs=25 threads=5 spinningthreads=1 idlethreads=0 read
SCHED 0ms: gomaxprocs=28 idleprocs=25 threads=5 spinningthreads=1 idlethreads=0 read
hello
world
Channel 1: world
Channel 2: hello
using debuggers:

- delve
- gdb
How to set breakpoint inside goroutine?

```go
package main

import (  
  "fmt"
)

func say(s string, r chan string) {  
  fmt.Println(s)  
  r <- s
}

func main() {  
  chan1 := make(chan string)  
  chan2 := make(chan string)

  go say("world", chan1)  
  go say("hello", chan2)

  res1 := <-chan1  
  res2 := <-chan2

  fmt.Printf("Channel 1: %s\nChannel 2: %s\n", res1, res2)
}
```

How to set breakpoint inside goroutine?

```go
> main.say() ./main.go:9 (hits goroutine(7):1 total:1) (PC: 0x10c46c9)

        "fmt"
4:     
5:     )
6:     
7:     func say(s string, r chan string) {
8:             fmt.Println(s)
=> 9:             r <- s
10:     }
11:     
12:     func main() {
13:             chan1 := make(chan string)
14:             chan2 := make(chan string)
```
How to debug channel?

```go
text
package main

func main() {
    ch := make(chan int, 4)
    ch <- 1
    ch <- 2
    ch <- 3
    ch <- 4
    close(ch)
}
```
How to debug channel?

```go
> main.main() ./main.go:7 (PC: 0x10279fb38)
 2:
 3:
 4: func main() {
 5:   ch := make(chan int, 4)
 6:   ch <- 1
 7:   ch <- 2
 8:   ch <- 3
 9:   ch <- 4
10:  close(ch)
11: }
(dlv) p ch
chan int {
  qcount: 1,
  dataqsz: 4,
  buf: *[4]int [1,0,0,0],
  elemsize: 8,
  closed: 0,
  elemtype: *runtime._type {size: 8, ptrdata: 0, hash: 341333390,
    fieldAlign: 8, kind: 2, equal: runtime.memequal64, gcdata: *0, str: 331,
    sendx: 1,
    recvx: 0,
    recvq: waitq<int> {
      first: *sudog<int> nil,
      last: *sudog<int> nil,},
  sendq: waitq<int> {
      first: *sudog<int> nil,
      last: *sudog<int> nil,},
  lock: runtime.mutex { lockRankStruct: runtime.lockRankStruct {},
    key: 0,},
(dlv)
```
How to debug channel?

```
(dlv) n
> main.main() ./main.go:8 (PC: 0x10279fb48)
  3:
  4: func main() {
  5:   ch := make(chan int, 4)
  6:   ch <- 1
  7:   ch <- 2
  8:   ch <- 3
  9:   ch <- 4
 10:  close(ch)
 11: }
(dlv) p ch
chan int {
    qcount: 2,
    dataqsiz: 4,
    buf: *[4]int [1,2,0,0],
    elemsize: 8,
    closed: 0,
    elemtype: *runtime.type {size: 8, ptrdata: 0, hash: 3413333333,
     fieldAlign: 8, kind: 2, equal: runtime.memequal64, gndata: *0, str: 33
     sendx: 2,
     recvx: 0,
     recvq: waitq<int> {
         first: *sudog<int> nil,
         last: *sudog<int> nil},
     sendq: waitq<int> {
         first: *sudog<int> nil,
         last: *sudog<int> nil},
     lock: runtime.mutex {
         lockRankStruct: runtime.lockRankStruct {},
         key: 0,}
}
dlv send to channel value similar to set variable.

(dlv) set ch <- 3
Command failed: 1:4: expected 'EOF', found '<-'
(dlv) set a = 2
Command failed: could not find symbol value for a
(dlv) set a := 2
Command failed: 1:3: expected 'EOF', found ':='
(dlv) set
Command failed: 1:1: expected operand, found 'EOF'
(dlv) help set
Changes the value of a variable.

[goroutine <n>] [frame <m>] set <variable> = <value>

https://github.com/go-delve/delve/issues/2117
Sending output to pager...
(dlv) goroutine
Thread 7543681 at ./pages.go:20
Goroutine 163:
    Runtime: ./pages.go:20 main.loginHandler (0x102f0f19c)
    User: ./pages.go:20 main.loginHandler (0x102f0f19c)
Start: /opt/homebrew/Cellar/go/1.19.1/libexec/src/net/http/server.go:1842 net/http.(*conn
Labels: "path": "/login", "request": "real"
Profile labels

labels := pprof.Labels("worker", "purge")
pprof.Do(ctx, labels, func(ctx context.Context) {
    // Do some work...
    
go update(ctx) // propagates labels in ctx.
})
goroutines labels

```go
go func(p string, rid int64) {
    labels := pprof.Labels("request", "automated", "page", p, "rid", str
    pprof.Do(context.Background(), labels, func(_ context.Context) {
        makeRequest(activeConns, c, p, rid)
    })
}(page, i)
```
Or you can use Debugger Middleware

```
router.HandleFunc("/", debugger.Middleware(homeHandler, func(r *http.Request) [][]string {
    return [][]string{
        "path", r.RequestURI,
    }
})
```
Or you can use Directly

Original:

```go
func sum(a, b int) int {
    return a+b
}
```

Replacement:

```go
func sum(a, b int) int {
    debugger.SetLabels(func() []string {
        return []string{
            "a", strconv.Itoa(a),
            "b", strconv.Itoa(b),
        }
    })

    return a+b
}
```

https://github.com/dlsniper/debugger
goroutines dlv

(dlv) goroutines -l
* Goroutine 1 - User: ./main.go:8 main.main (0x10279fb48) (thread 7511929)
Goroutine 2 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/runtime/proc.go:364
Goroutine 3 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/runtime/proc.go:364
Goroutine 4 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/runtime/proc.go:364
[4 goroutines]
(dlv)
goroutines labels golang

[6 goroutines]
(dlv) goroutines -l -with label page=/about
  Goroutine 7 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/r
    Labels: "page":"/about", "request":"automated", "rid":"1"
  Goroutine 11 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/r
    Labels: "page":"/about", "request":"automated", "rid":"5"
  Goroutine 167 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/r
    Labels: "page":"/about", "request":"automated", "rid":"1"
  Goroutine 168 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/r
    Labels: "page":"/about", "request":"automated", "rid":"1"
  Goroutine 173 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/r
    Labels: "page":"/about", "request":"automated", "rid":"5"
  Goroutine 174 - User: /opt/homebrew/Cellar/go/1.19.1/libexec/src/r
    Labels: "page":"/about", "request":"automated", "rid":"5"
[6 goroutines]
(dlv)
You can play on you own using example project:

https://github.com/dlsniper/serverdemo

dlv debug --build-flags="-ldflags=-s -tags=debugger" *.go
Gdb and golang

go build -ldflags=-compressdwarf=false -gcflags=all="-N -l" -o main main.go
Gdb and goroutines

Thread 3 hit Breakpoint 1, main.say (s=..., r=0xc000100060) at /Users/andri
8     fmt.Println(s)

(gdb) info goroutines
  1 waiting runtime.gopark
  2 waiting runtime.gopark
  3 waiting runtime.gopark
  4 waiting runtime.gopark
  5 waiting runtime.gopark
* 6 running main.say
* 7 running fmt.Println

(gdb) c
Gdb and goroutines

(gdb) bt
#0  main.say (s=..., r=0xc0001000c0) at /Users/andrii/workspaces
#1  0x0000000001063ce1 in runtime.goexit () at /usr/local/Cell
#2  0x00000000010f4ac6 in string.* ()
#3  0x000000000000000005 in ?? ()
#4  0x000000c001000c0 in ?? ()
#5  0x0000000000000000 in ?? ()

(gdb) goroutine 1 bt
#0  runtime.gopark (unlockf={void (runtime.g *, void *, bool *,
#1  0x0000000001005aca in runtime.chanrecv (c=0xc000100060, ep
#2  0x00000000010058ab in runtime.chanrecv1 (c=0xc000100060, ep
#3  0x00000000010c4829 in main.main () at /Users/andrii/workspa
Deadlocks happen and are painful to debug.
How to detect deadlocks

```go
package main

func main() {
    ch := make(chan string)
    ch <- "hello deadlock"
}
```

defatal error: all goroutines are asleep - deadlock!

goroutine 1 [chan send]:
main.main()
/Users/andrii/workspace/src/github.com/andriisoldatenko/gowayfest/main_deadlock.go:5 +0x50
exit status 2
Real world examples
complicated scenario & tools

https://github.com/sasha-s/go-deadlock

https://github.com/cockroachdb/cockroach/issues/7972
Data races:

```
func main() {
    c := make(chan bool)  //_chan bool
    m := make(map[string]string)  //_map[string]string
    go func() {
        m["1"] = "a"  // First conflicting access.
        c <- true
    }()
    m["2"] = "b"  // Second conflicting access.
    <-c
    for k, v := range m {  // range map[string]string
        fmt.Println(k, v)
    }
}
```
Data races:
go run -race race_demo.go

Previous write at 0x00c000124180 by main goroutine:
  runtime.mapaccess2_faststr()
       /opt/homebrew/Cellar/go/1.19.1/libexec/src/runtime/map_faststr.go:108 +0x43c
  main.main()
       /Users/andrii/fosdem/go_devroom/race_demo.go:12 +0xfc

Goroutine 7 (running) created at:
  main.main()
       /Users/andrii/fosdem/go_devroom/race_demo.go:8 +0xe0

==============
2 b
1 a

Found 1 data race(s)
exit status 66
7 simple rules for debugging concurrency applications

- Never assume a particular order of execution
- Implement concurrency at the highest level possible
- Don’t forget Go only detects when the program as a whole freezes, not when a subset of goroutines get stuck.
- STRACE
7 simple rules for debugging concurrency applications

- conditional breakpoints your best friend
- DEBUG=schedtrace=5000
- go-deadlock
- And last but not least use Debugger!
References 1

- https://github.com/golang/go/wiki/LearnConcurrency
- https://rakyll.org/go-cloud/
- https://yourbasic.org/golang/detect-deadlock/
References 2

- https://blog.minio.io/debugging-go-routine-leaks-
a1220142d32c

- https://golang.org/src/cmd/link/internal/ld/dwarf.go

- https://golang.org/src/runtime/runtime-gdb.py

- https://cseweb.ucsd.edu/~yiying/GoStudy-
  ASPLOS19.pdf

References 3

- https://go.dev/doc/articles/race_detector
Slides:

@asoldatenko
Thank You

@a_soldatenko

andriii@hachyderm.io
Questions ?