go tool trace

for correct and effective concurrency

Let's talk about Go

Let's talk about `go`

A special tool for Go's needs

* and Chromium

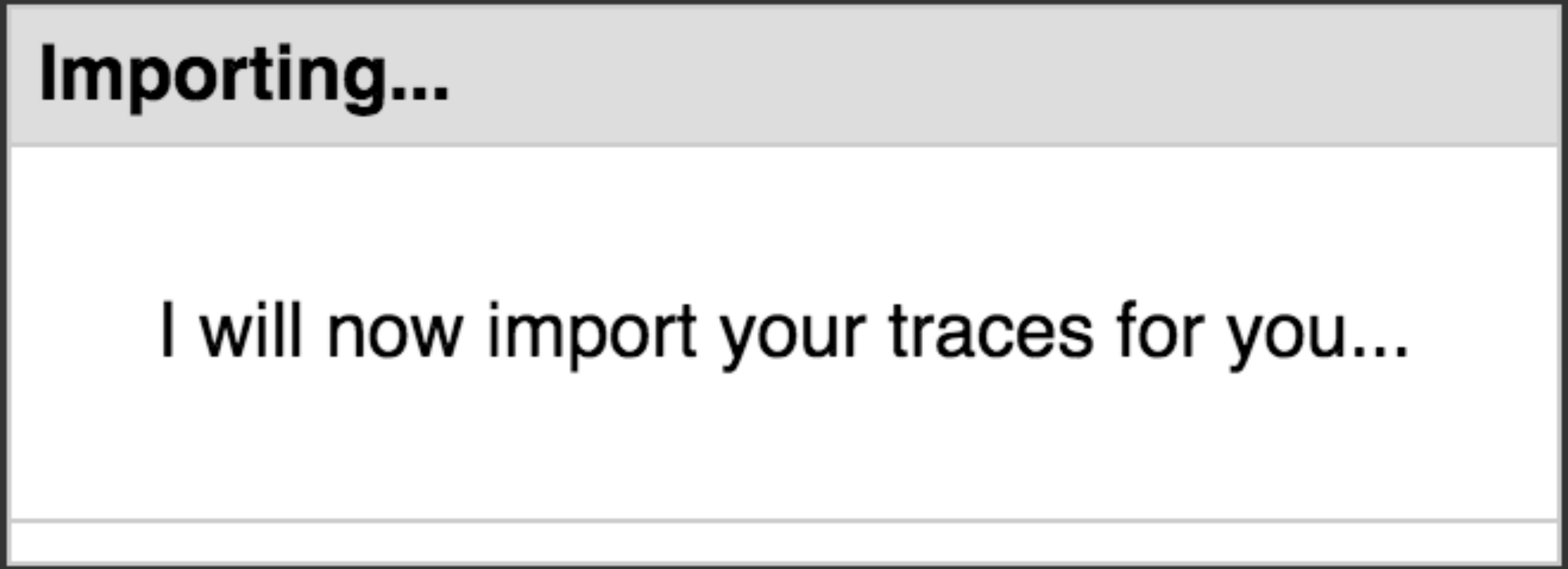
goroutine scheduling instrumented

Concurrency,
and how to manage it

Parallelism,
and how to exploit it

```
$ go doc runtime/trace
package trace // import "runtime/trace"

Go execution tracer. The tracer captures
a wide range of execution events ...

A trace can be analyzed later with
'go tool trace' command.
```

```
$ go doc cmd/trace
Trace is a tool for viewing trace files.

...

View the trace in a web browser:

    go tool trace trace.out
```


Importing...

I will now import your traces for you...

806 ms
808 ms
810 ms
812 ms
814 ms
816 ms
818 ms
820 ms
822 ms
824 ms
GC
GC
GC
G72
G7
G73
G74
G6

The tool in three demos:

1. A timing-dependent bug
2. What it doesn't show
3. Latency during GC

#1: A race condition

```
go test -race
go build -race
go install -race
```

```
==================
WARNING: DATA RACE
Read at 0x00c420141500 by goroutine 27:
  ...

Previous write at 0x00c420141500 by
goroutine 26:
  ...
```

a logical race, not a data race

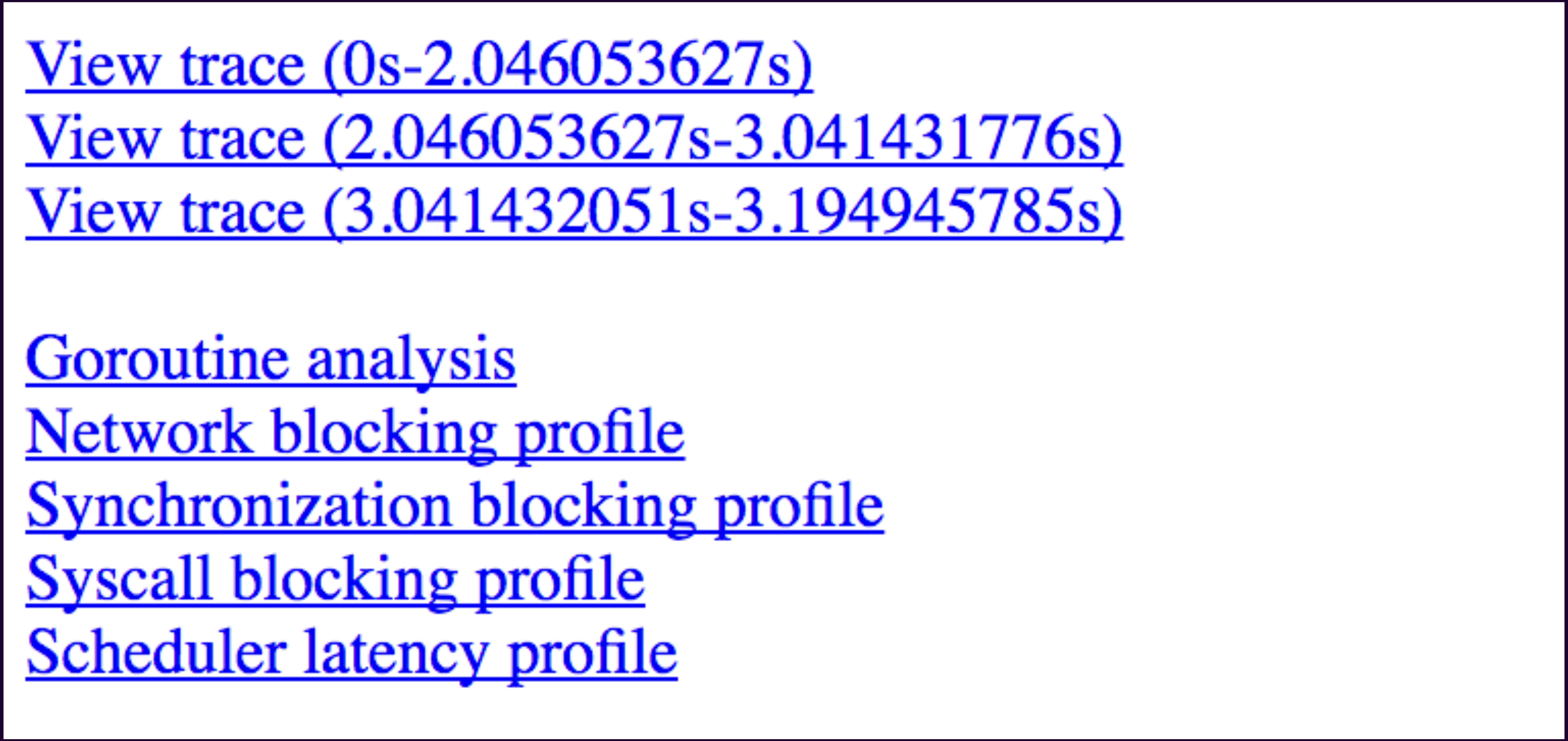
gRPC, HTTP/2, and flow control

trace
M
View Options
STATS (pid 1)
Heap:
Goroutines:
Threads:
PROCS (pid 0)
GC
Network
Syscalls
Proc 0
Proc 1
Proc 2
Proc 3
Proc 4
Proc 5
Proc 6
Proc 7
-0.2 s
0 s
0.2 s
0.4 s
0.6 s
0.8 s
1 s
1.2 s
1.4 s
1.6 s
1.8 s
2 s
2.2 s
File Size Stats
Nothing selected. Tap stuff.

stuff happening

stuff not happening

```
$ go test -trace=trace.out
$ go tool trace trace.out
```



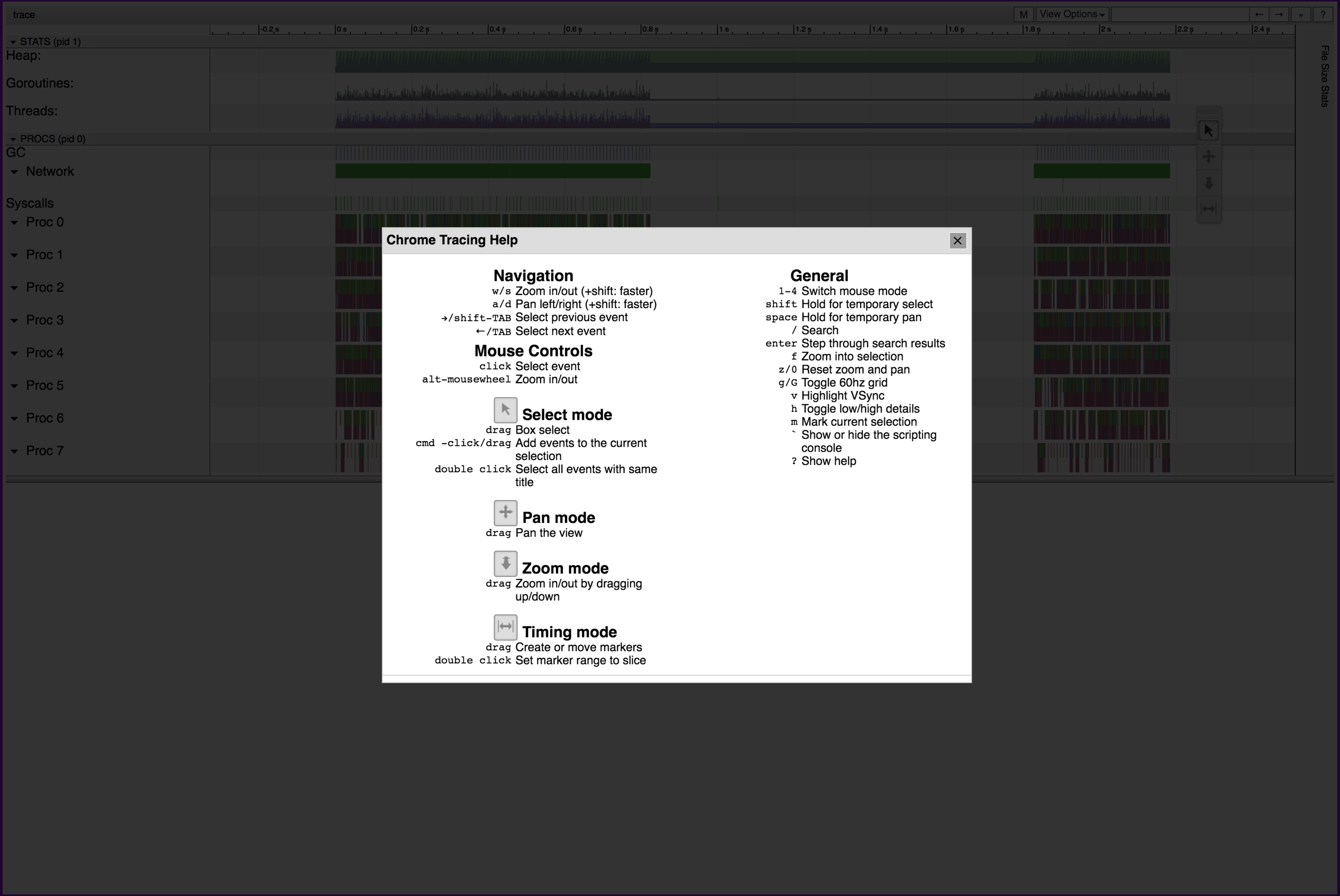
View trace (0s-2.046053627s)
View trace (2.046053627s-3.041431776s)
View trace (3.041432051s-3.194945785s)

Goroutine analysis
Network blocking profile
Synchronization blocking profile
Syscall blocking profile
Scheduler latency profile

"View trace"

trace
M
View Options
-0.2 s
0 s
0.2 s
0.4 s
0.6 s
0.8 s
1 s
1.2 s
1.4 s
1.6 s
1.8 s
2 s
2.2 s
STATS (pid 1)
Heap:
Goroutines:
Threads:
PROCS (pid 0)
GC
Network
Syscalls
Proc 0
Proc 1
Proc 2
Proc 3
Proc 4
Proc 5
Proc 6
Proc 7
1s
Nothing selected. Tap stuff.
File Size Stats

(from context.WithTimeout)



Chrome Tracing Help

Navigation

`w/s`	Zoom in/out (+shift: faster)
`a/d`	Pan left/right (+shift: faster)
`→/shift-TAB`	Select previous event
`←/TAB`	Select next event

Mouse Controls

`click`	Select event
`alt-mousewheel`	Zoom in/out

Select mode

`drag`	Box select
`cmd -click/drag`	Add events to the current selection
`double click`	Select all events with same title

Pan mode

`drag`	Pan the view

Zoom mode

`drag`	Zoom in/out by dragging up/down

Timing mode

`drag`	Create or move markers
`double click`	Set marker range to slice

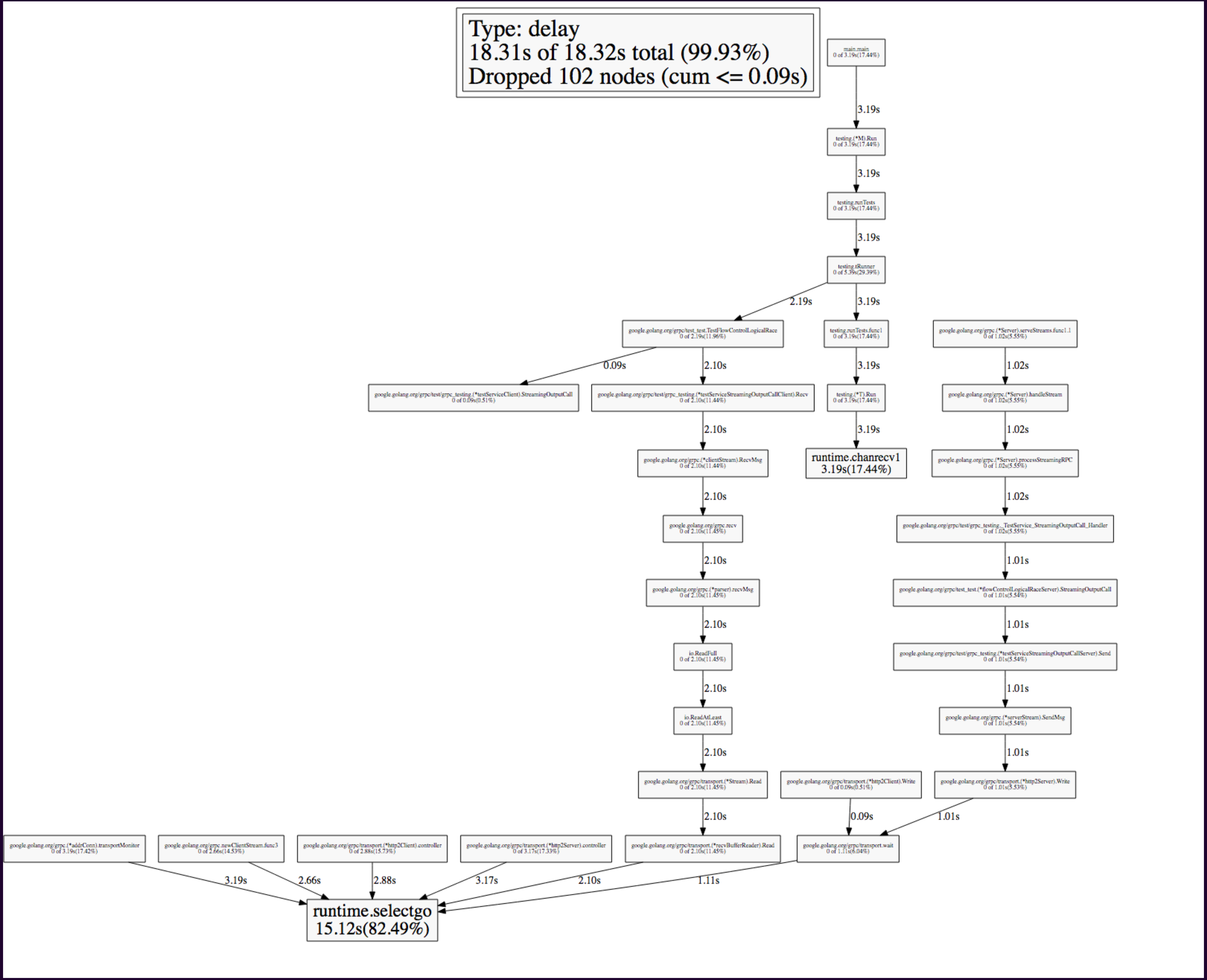
General

`1-4`	Switch mouse mode
`shift`	Hold for temporary select
`space`	Hold for temporary pan
`/`	Search
`enter`	Step through search results
`f`	Zoom into selection
`z/0`	Reset zoom and pan
`g/G`	Toggle 60hz grid
`v`	Highlight VSync
`h`	Toggle low/high details
`m`	Mark current selection
`` ` ``	Show or hide the scripting console
`?`	Show help

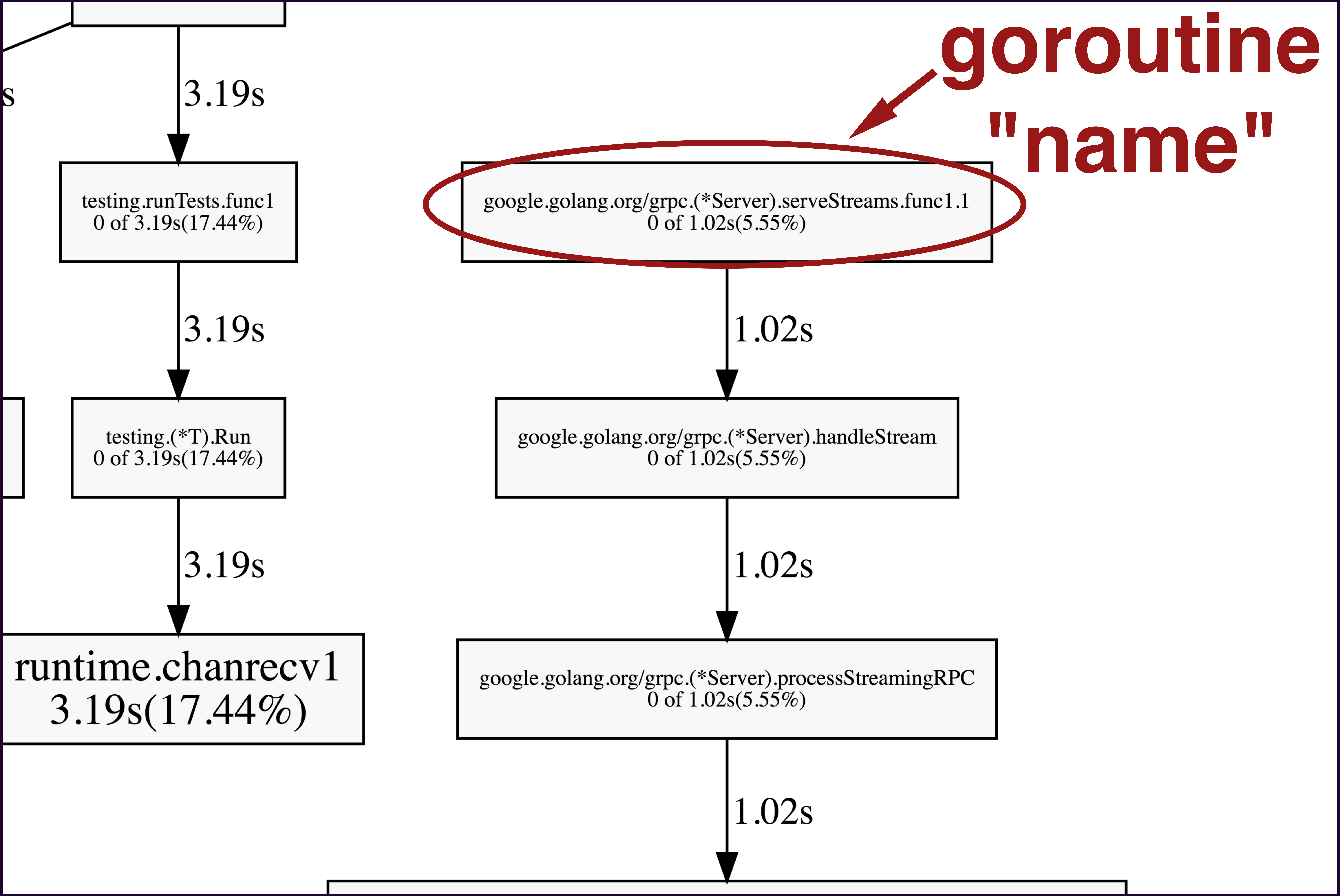
trace
M
View Options
806 ms
808 ms
810 ms
812 ms
814 ms
818 ms
820 ms
822 ms
824 ms
826 ms
STATS (pid 1)
Heap:
Goroutines:
Threads:
PROCS (pid 0)
GC
Network
Syscalls
Proc 0
Proc 1
Proc 2
Proc 3
Proc 4
Proc 5
Proc 6
Proc 7
File Size Stats
Nothing selected. Tap stuff.
20ms
Garbage collection

Goroutines running on OS threads

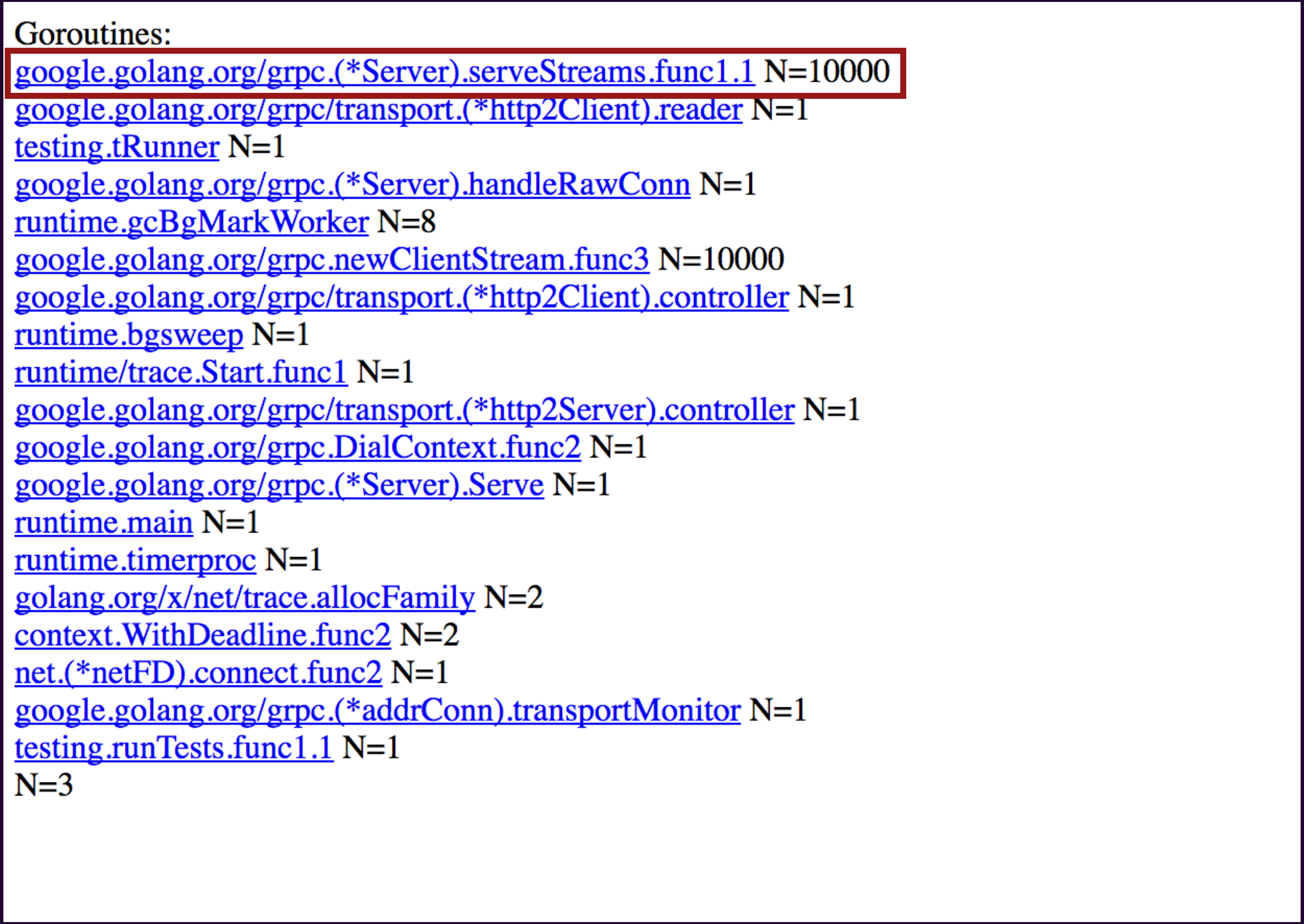
"Sync blocking profile"


Type: delay
18.31s of 18.32s total (99.93%)
Dropped 102 nodes (cum <= 0.09s)
main.main 0 of 3.19s(17.44%)
testing.(*M).Run 0 of 3.19s(17.44%)
testing.runTests 0 of 3.19s(17.44%)
testing.tRunner 0 of 5.39s(29.39%)
google.golang.org/grpc/test_test.TestFlowControlLogicalRace 0 of 2.19s(11.96%)
testing.runTests.func1 0 of 3.19s(17.44%)
google.golang.org/grpc.(*Server).serveStreams.func1.1 0 of 1.02s(5.55%)
google.golang.org/grpc/test/grpc_testing.(*testServiceClient).StreamingOutputCall 0 of 0.09s(0.51%)
google.golang.org/grpc/test/grpc_testing.(*testServiceStreamingOutputCallClient).Recv 0 of 2.10s(11.44%)
testing.(*T).Run 0 of 3.19s(17.44%)
google.golang.org/grpc.(*Server).handleStream 0 of 1.02s(5.55%)
google.golang.org/grpc.(*clientStream).RecvMsg 0 of 2.10s(11.44%)
runtime.chanrecv1 3.19s(17.44%)
google.golang.org/grpc.(*Server).processStreamingRPC 0 of 1.02s(5.55%)
google.golang.org/grpc.recv 0 of 2.10s(11.45%)
google.golang.org/grpc/test/grpc_testing._TestService_StreamingOutputCall_Handler 0 of 1.02s(5.55%)
google.golang.org/grpc.(*parser).recvMsg 0 of 2.10s(11.45%)
google.golang.org/grpc/test_test.(*flowControlLogicalRaceServer).StreamingOutputCall 0 of 1.01s(5.54%)
io.ReadFull 0 of 2.10s(11.45%)
google.golang.org/grpc/test/grpc_testing.(*testServiceStreamingOutputCallServer).Send 0 of 1.01s(5.54%)
io.ReadAtLeast 0 of 2.10s(11.45%)
google.golang.org/grpc.(*serverStream).SendMsg 0 of 1.01s(5.54%)
google.golang.org/grpc/transport.(*Stream).Read 0 of 2.10s(11.45%)
google.golang.org/grpc/transport.(*http2Client).Write 0 of 0.09s(0.51%)
google.golang.org/grpc/transport.(*http2Server).Write 0 of 1.01s(5.53%)
google.golang.org/grpc.(*addrConn).transportMonitor 0 of 3.19s(17.42%)
google.golang.org/grpc.newClientStream.func3 0 of 2.66s(14.53%)
google.golang.org/grpc/transport.(*http2Client).controller 0 of 2.88s(15.73%)
google.golang.org/grpc/transport.(*http2Server).controller 0 of 3.17s(17.33%)
google.golang.org/grpc/transport.(*recvBufferReader).Read 0 of 2.10s(11.45%)
google.golang.org/grpc/transport.wait 0 of 1.11s(6.04%)
runtime.selectgo 15.12s(82.49%)

google.golang.org/grpc.(*serverStream).SendMsg
0 of 1.01s(5.54%)
1.01s
).Read
google.golang.org/grpc/transport.(*http2Client).Write
0 of 0.09s(0.51%)
google.golang.org/grpc/transport.(*http2Server).Write
0 of 1.01s(5.53%)
0.09s
1.01s
eader).Read
google.golang.org/grpc/transport.wait
0 of 1.11s(6.04%)
one-second
unexpected delay


goroutine
"name"
3.19s
testing.runTests.func1
0 of 3.19s(17.44%)
3.19s
testing.(*T).Run
0 of 3.19s(17.44%)
3.19s
runtime.chanrecv1
3.19s(17.44%)
google.golang.org/grpc.(*Server).serveStreams.func1.1
0 of 1.02s(5.55%)
1.02s
google.golang.org/grpc.(*Server).handleStream
0 of 1.02s(5.55%)
1.02s
google.golang.org/grpc.(*Server).processStreamingRPC
0 of 1.02s(5.55%)
1.02s

"Goroutine analysis"



Goroutines:
google.golang.org/grpc.(*Server).serveStreams.func1.1 N=10000
google.golang.org/grpc/transport.(*http2Client).reader N=1
testing.tRunner N=1
google.golang.org/grpc.(*Server).handleRawConn N=1
runtime.gcBgMarkWorker N=8
google.golang.org/grpc.newClientStream.func3 N=10000
google.golang.org/grpc/transport.(*http2Client).controller N=1
runtime.bgsweep N=1
runtime/trace.Start.func1 N=1
google.golang.org/grpc/transport.(*http2Server).controller N=1
google.golang.org/grpc.DialContext.func2 N=1
google.golang.org/grpc.(*Server).Serve N=1
runtime.main N=1
runtime.timerproc N=1
golang.org/x/net/trace.allocFamily N=2
context.WithDeadline.func2 N=2
net.(*netFD).connect.func2 N=1
google.golang.org/grpc.(*addrConn).transportMonitor N=1
testing.runTests.func1.1 N=1
N=3

Goroutine	Total time, ns	Execution time, ns	Network wait time, ns	Sync block time, ns	Blocking syscall time, ns	Scheduler wait time, ns	GC sweeping time, ns	GC pause time, ns
8006	1002132537	54182	0	1002047334	0	31021	0	0
7540	1463723	1345897	0	0	0	117826	223532	490550
17035	1142502	1119065	0	0	0	23437	45775	0
565	1107174	1092100	0	0	0	15074	0	0
17167	1019955	699833	0	19565	2703	297854	32143	635731
35	956907	632455	0	252217	0	72235	0	0
17949	854637	679832	0	5154	12235	157416	23093	586337
7982	837615	674632	0	0	0	162983	47630	492979
5662	832185	620219	0	44491	0	167475	55213	418612
17964	831864	733236	0	0	0	98628	41124	418292
633	828565	688537	0	0	0	140028	57503	398841
16086	820569	693075	0	0	0	127494	59016	432313
7608	795322	596623	0	0	11913	186786	34915	518042
11394	790740	716259	0	0	687	73794	35212	431282
19298	787533	598503	0	0	0	189030	34845	494238
17658	774680	662557	0	0	6278	105845	49644	397100
19393	768517	653829	0	0	0	114688	42292	393480
18368	767028	616301	0	0	4285	146442	22498	599533
1114	764211	590116	0	0	0	174095	56888	378291
14811	764119	656739	0	0	6988	100392	60207	425485
15725	753947	479713	0	0	0	274234	29234	599487
2524	751747	601273	0	0	19154	131320	39909	444432
16325	745080	623040	0	0	0	122040	57736	381407
2251	739949	660840	0	0	0	79109	54434	328301
19710	732434	613715	0	0	4146	114573	36932	557860
7061	730464	598981	0	18741	8385	104357	28156	429358

trace
M
View Options
← → » ?
-0.1 s
0 s
0.1 s
0.2 s
0.3 s
0.4 s
0.5 s
0.6 s
0.7 s
0.8 s
0.9 s
1 s
1.1 s
File Size Stats
1s

PROCS (pid 0)
G8006 google.golang.org/grpc.
G7964 context.WithDeadline.fun…

Nothing selected. Tap stuff.

trace

M | View Options ▾ | ← | → | » | ?

-15 us | -10 us | -5 us | 0 us | 5 us | 10 us | 15 us | 20 us | 25 us | 30 us | 35 us | 40 us | 45

▾ PROCS (pid 0)

▾ G8006 google.golang.org/grpc.

G8006 google.golang.org/grpc.(*Server).serveStreams.func1.1

G7964 context.WithDeadline.fun…

File Size Stats

Nothing selected. Tap stuff.

trace
M
View Options
←
→
»
?
0 us
10 us
20 us
30 us
File Size Stats

PROCS (pid 0)

G8006 google.golang.org/grpc.

G8006 google.golang.org/grpc.(*Server).serveStreams.func1.1

G7964 context.WithDeadline.fun…

1 item selected: Slice (1)

Title	G8006 google.golang.org/grpc.(*Server).serveStreams.func1.1
Start	0.000 ms
Wall Duration	0.031 ms
Self Time	0.031 ms

Start Stack Trace

Title
google.golang.org/grpc.(*Server).serveStreams.func1.1:466

End Stack Trace

Title
runtime.selectgo:238
google.golang.org/grpc/transport.wait:586
google.golang.org/grpc/transport.(*http2Server).Write:648
google.golang.org/grpc.(*serverStream).SendMsg:564
google.golang.org/grpc/test/grpc_testing.(*testServiceStreamingOutputCallServer).Send:627
google.golang.org/grpc/test_test.(*flowControlLogicalRaceServer).StreamingOutputCall:3294
google.golang.org/grpc/test/grpc_testing._TestService_StreamingOutputCall_Handler:614

```
func wait(ctx context.Context, ...,
  proceed <-chan int) (int, error) {

  select {
  case <-ctx.Done():
    // ...
  case ...: // stream closed, etc
  case i := <-proceed:
    return i, nil
  }
}
```

```
t.sendQuotaPool.add(0)

tq, err := wait(s.ctx, ...,
  t.sendQuotaPool.acquire())

if err != nil && ... {
  t.sendQuotaPool.cancel()
  return
}

// calculate payload size ...
t.sendQuotaPool.add(tq-ps)
```

```
type quotaPool struct {
  c chan int

  mu    sync.Mutex
  quota int
}
```

```go
func newQuotaPool(q int) *quotaPool {
  qb := &quotaPool{
    c: make(chan int, 1),
  }
  if q > 0 {
    qb.c <- q
  } else {
    qb.quota = q
  }
  return qb
}
```

```go
func newQuotaPool(q int) *quotaPool {
  qb := &quotaPool{
    c: make(chan int, 1),
  }
  if q > 0 {
    qb.c <- q
  } else {
    qb.quota = q
  }
  return qb
}
```

```
func newQuotaPool(q int) *quotaPool {
  qb := &quotaPool{
    c: make(chan int, 1),
  }
  if q > 0 {
    qb.c <- q
  } else {
    qb.quota = q
  }
  return qb
}
```

```
func newQuotaPool(q int) *quotaPool {
  qb := &quotaPool{
    c: make(chan int, 1),
  }
  if q > 0 {
    qb.c <- q
  } else {
    qb.quota = q
  }
  return qb
}
```

```go
type quotaPool struct {
  c chan int // positive quota here

  mu    sync.Mutex
  quota int // zero or negative quota
}
```

```
func (qb *quotaPool) acquire()
  <-chan int {

  return qb.c
}
```

```go
func (qb *quotaPool) add(n int) {
  qb.mu.Lock(); defer qb.mu.Unlock()
  if qb.quota += n; qb.quota <= 0 {
    return
  }
  select {
  case qb.c <- qb.quota:
    qb.quota = 0
  default:
  }
}
```

```go
func (qb *quotaPool) cancel() {
  qb.mu.Lock(); defer qb.mu.Unlock()

  select {
  case n := <-qb.c:
    qb.quota += n
  default:
  }
}
```

```
Goroutine 10:

pool.add(0)
wait(..., pool.aq())
pool.add(tq-ps)
```

```
Goroutine 300:       Goroutine 400:

pool.add(0)
wait(..., pool.aq())
pool.cancel()
                     pool.add(0)
                     wait(..., pool.aq())
                     pool.add(tq-ps)
```

```
Goroutine 5000:      Goroutine 6000:

pool.add(0)
                     pool.add(0)
wait(..., pool.aq())
pool.add(tq-ps)
                     wait(..., pool.aq())
                     pool.add(tq-ps)
```

trace
M
View Options
STATS (pid 1)
Heap:
Goroutines:
Threads:
PROCS (pid 0)
GC
Network
Syscalls
Proc 0
Proc 1
Proc 2
Proc 3
Proc 4
Proc 5
Proc 6
Proc 7
0 us
887320 us
887330 us
887340 us
887350 us
887360 us
887370 us
887380 us
887390 us
887400 us
887410 us
887420 us
887430 us
887440 us
887450 us
887460 us
887470 us
G12 google.golang.org/grpc/tran...
G6 testing...
G11 google.golang.org...
G7963 go...
golang.org/grpc/transport.(*h...
G6 testing.tRunner
G11 google.golang.org/grpc/transport.(*h...
G18 google.golang.org/grpc.(*Serve...
G7962 google.golang.org/grpc.(*Server).serveStreams.func1.1
G7961 google.gola...
G11 google...
G18 google.golang.org/grpc.(*Serve...
G8006 google.golang.org/grpc.(*Server).se...
File Size Stats
Nothing selected. Tap stuff.
36µs

G7962, another goroutine running the same function

G8006, about to stall for one second

```
Goroutine 7962:       Goroutine 8006:

pool.add(0)
                      pool.add(0)
wait(..., pool.aq())
pool.cancel()
                      wait(..., pool.aq())
                      pool.cancel()
```

```
- func (qb *quotaPool) cancel() {
-   qb.mu.Lock(); defer qb.mu.Unlock()
-
-   select {
-   case n := <-qb.c:
-     qb.quota += n
-   default:
-   }
- }
```

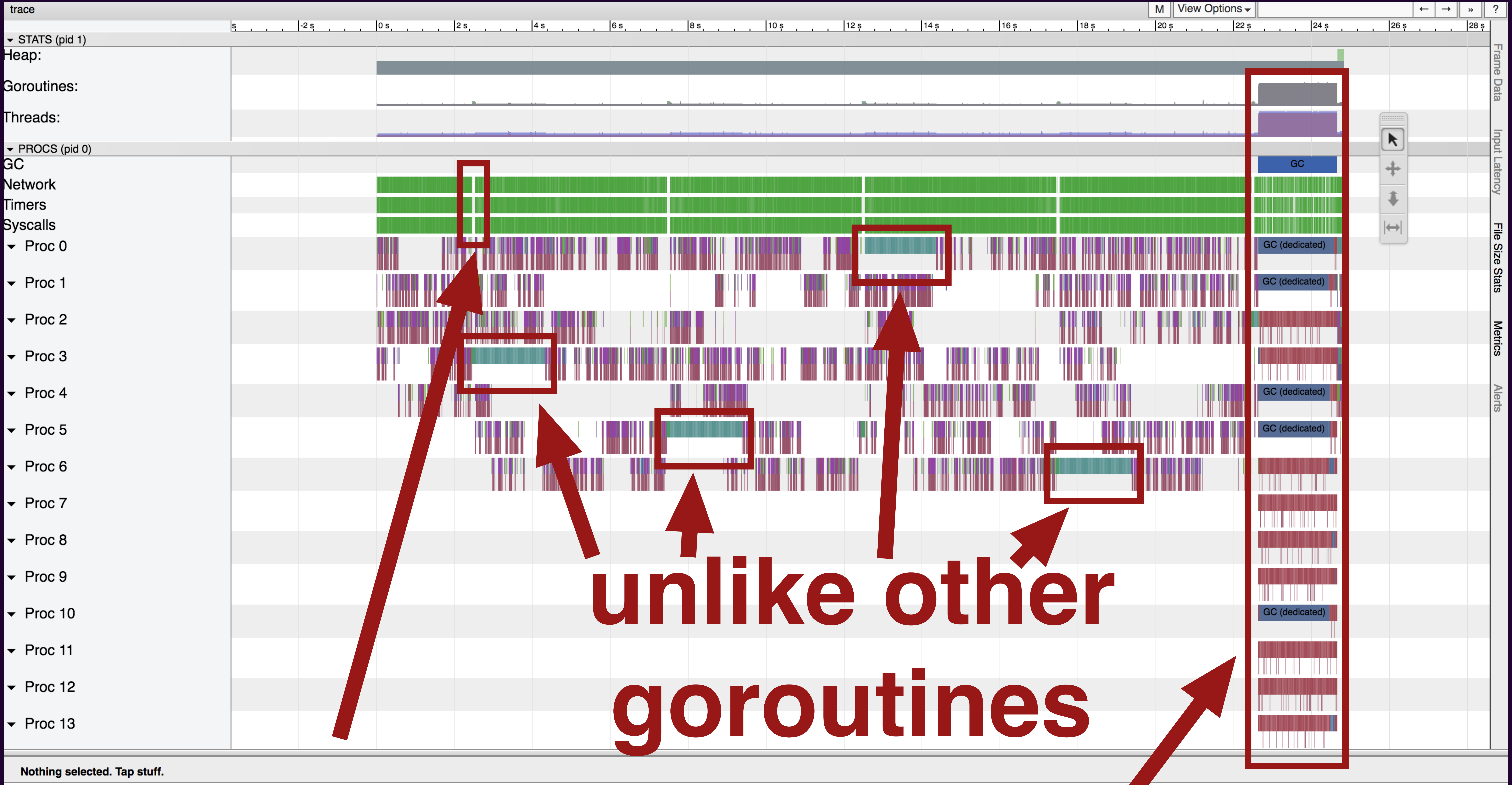
```go
func (qb *quotaPool) add(v int) {
	qb.mu.Lock(); defer qb.mu.Unlock()
	select {
	case n := <-qb.c:
		qb.quota += n
	default:
	}
	if qb.quota += v; qb.quota > 0 {
		qb.c <- qb.quota
		qb.quota = 0
	}
}
```

```
func (qb *quotaPool) add(v int) {
  qb.mu.Lock(); defer qb.mu.Unlock()
  select {
  case n := <-qb.c:
    qb.quota += n
  default:
  }
  if qb.quota += v; qb.quota > 0 {
    qb.c <- qb.quota
    qb.quota = 0
  }
}
```

#2: It's not a panacea

Three ways to get data:

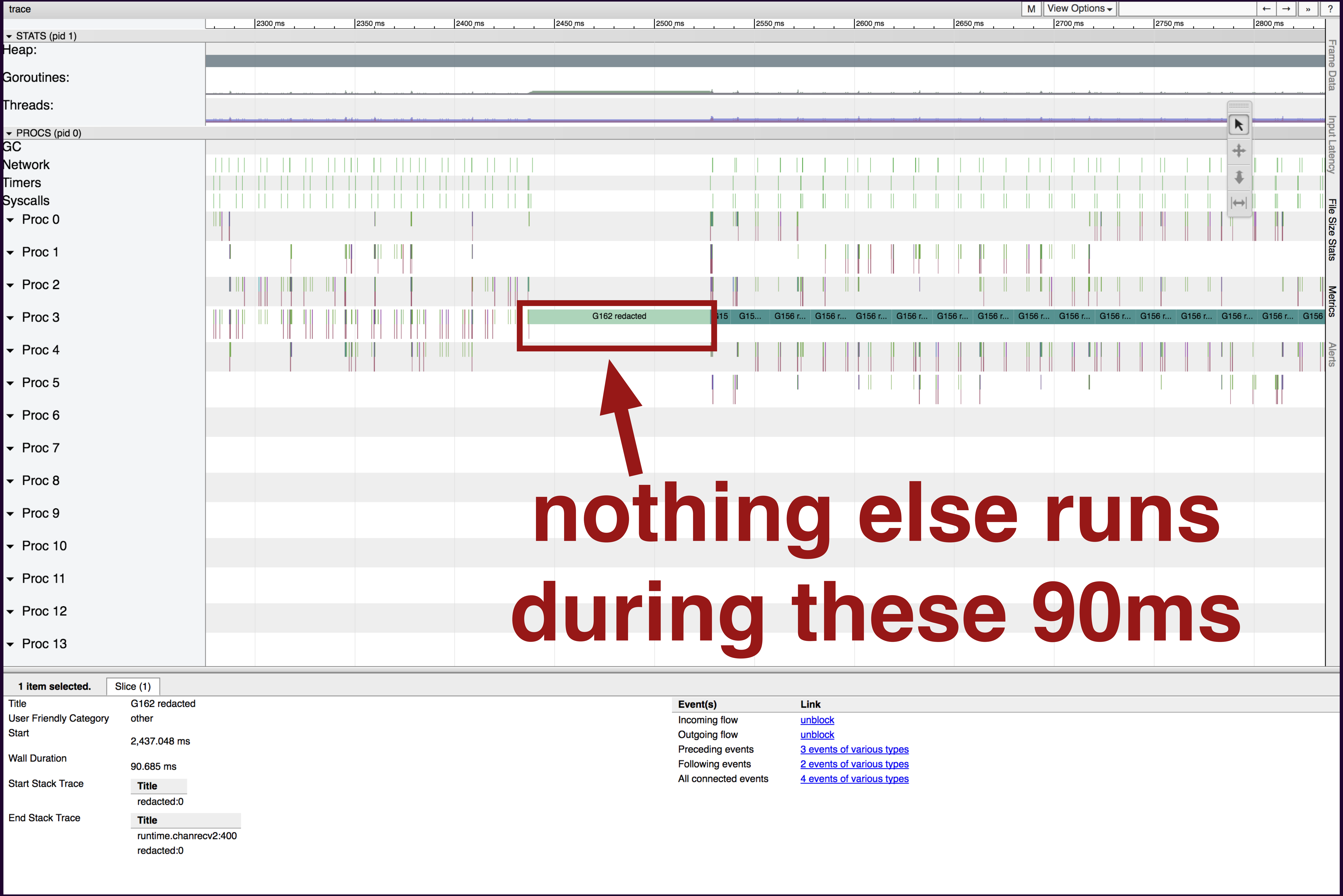
1. Testing with -trace flag
2. Direct runtime/trace use
3. net/http/pprof handlers



Nothing selected. Tap stuff.

suspicious gaps every five seconds

garbage collection



nothing else runs during these 90ms

1 item selected. | Slice (1)

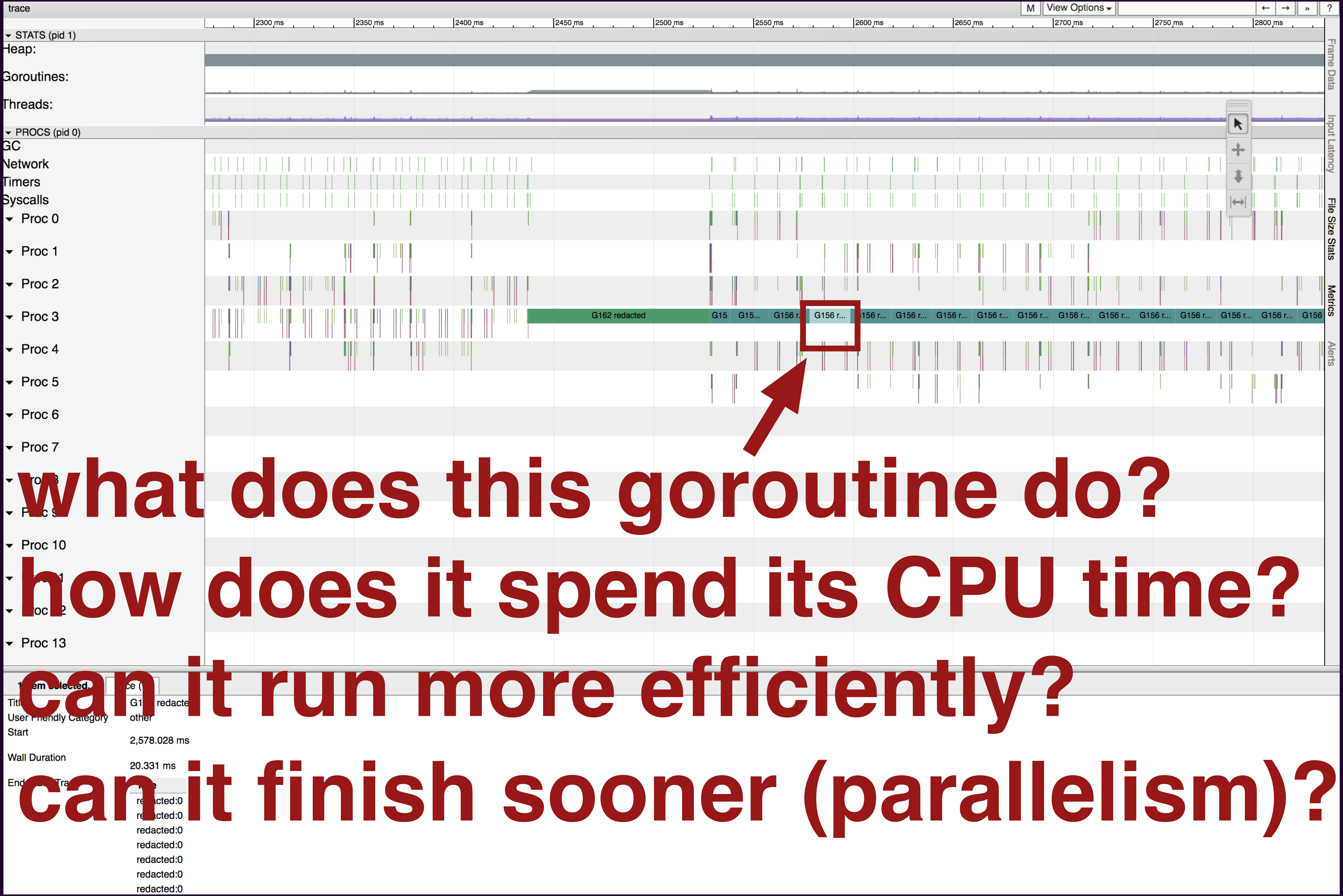
Title	G162 redacted
User Friendly Category	other
Start	2,437.048 ms
Wall Duration	90.685 ms
Start Stack Trace	Title redacted:0
End Stack Trace	Title runtime.chanrecv2:400 redacted:0

Event(s)	Link
Incoming flow	unblock
Outgoing flow	unblock
Preceding events	3 events of various types
Following events	2 events of various types
All connected events	4 events of various types

runtime.ReadMemStats

- and Go 1.8 or earlier
- and large (40GB) heap
- leads to long pauses

(fixed in Go 1.9)



Title G1 redacte
User Friendly Category other
Start 2,578.028 ms
Wall Duration 20.331 ms

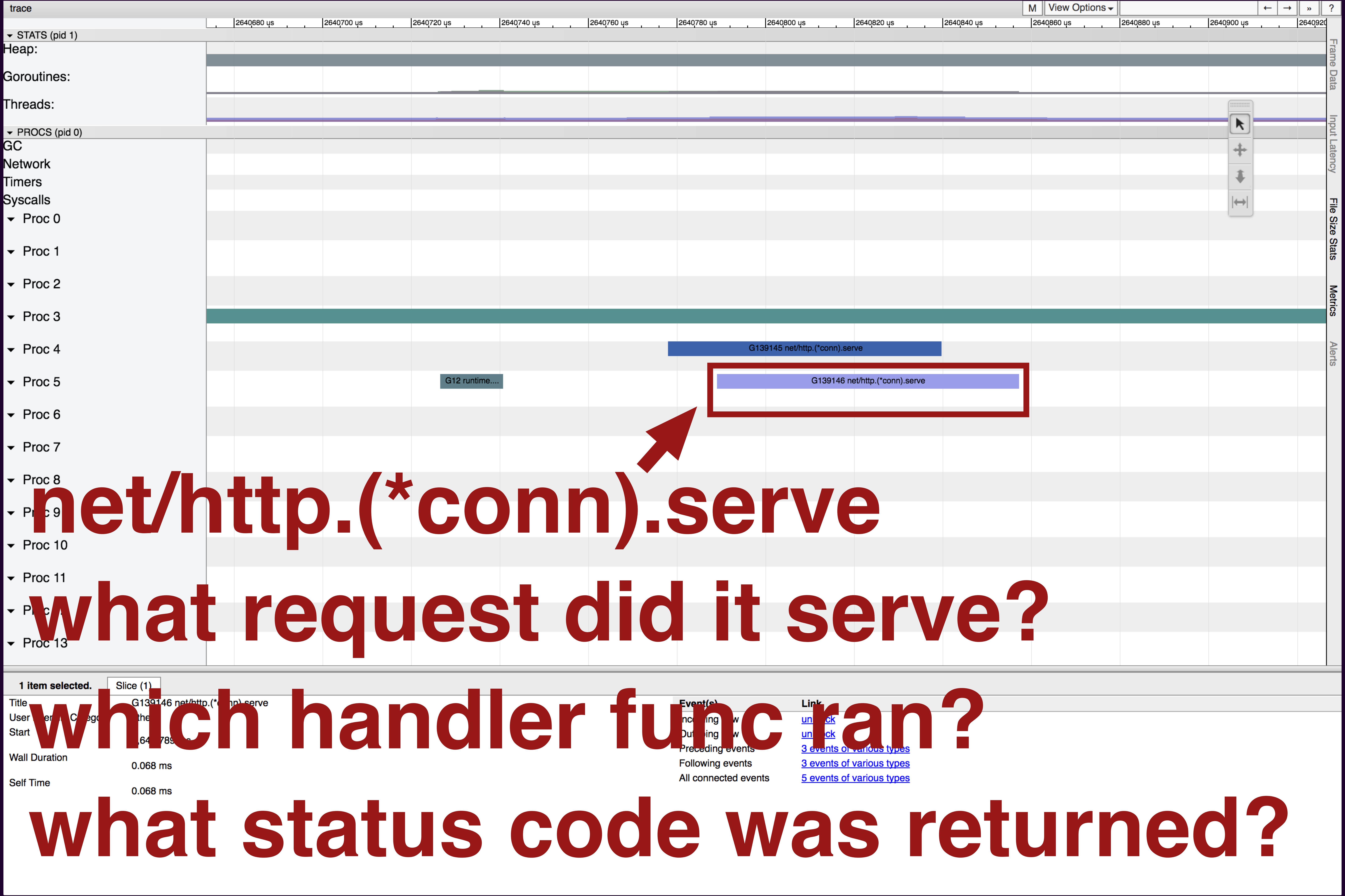
redacted:0
redacted:0
redacted:0
redacted:0
redacted:0
redacted:0
redacted:0

what does this goroutine do?

how does it spend its CPU time?

can it run more efficiently?

can it finish sooner (parallelism)?



Title G139146 net/http.(*conn).serve
Start
Wall Duration 0.068 ms
Self Time 0.068 ms

Event(s) Link
Preceding events 3 events of various types
Following events 3 events of various types
All connected events 5 events of various types

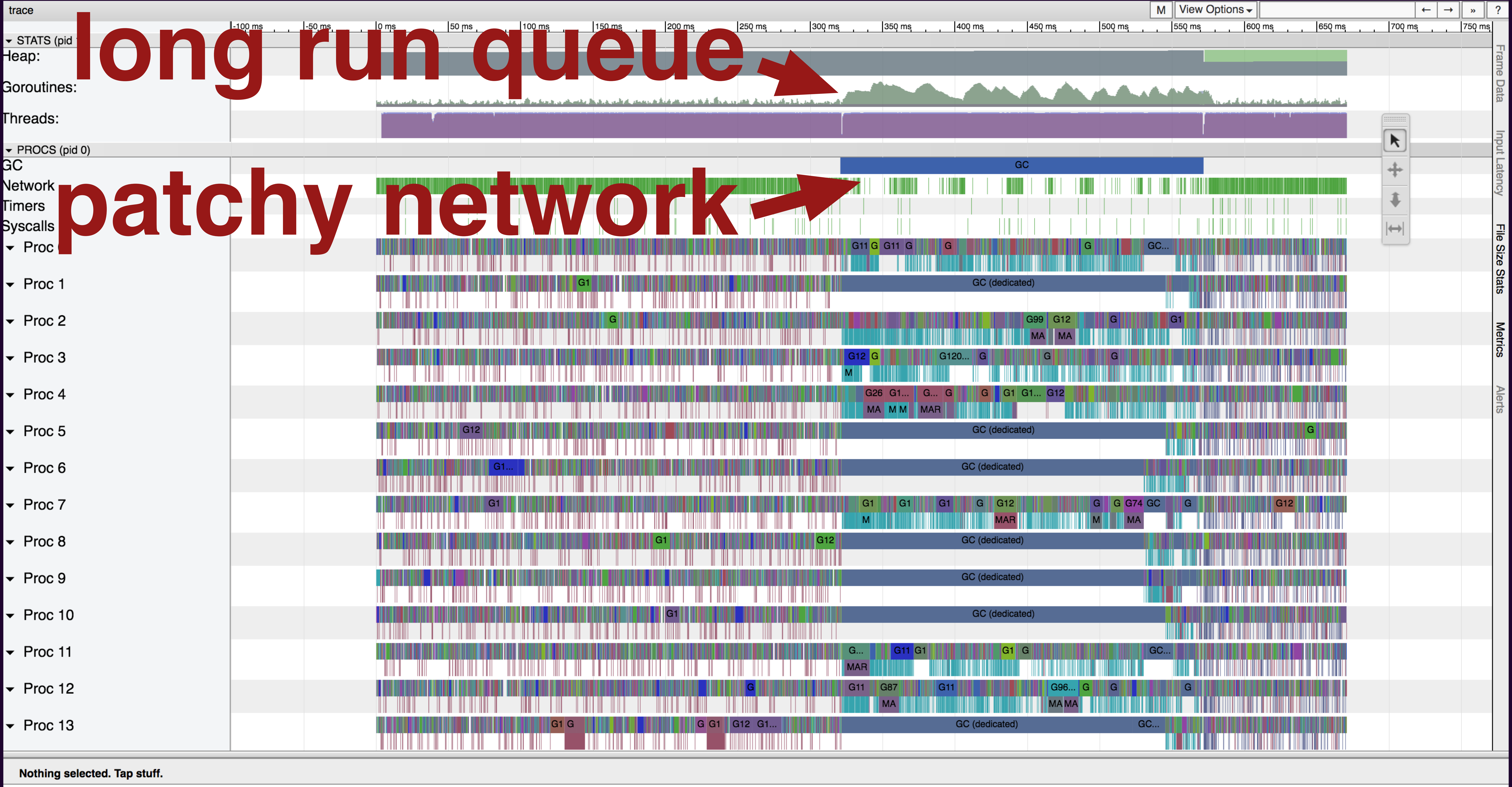
net/http.(*conn).serve

what request did it serve?

which handler func ran?

what status code was returned?

#3: The GC is still improving



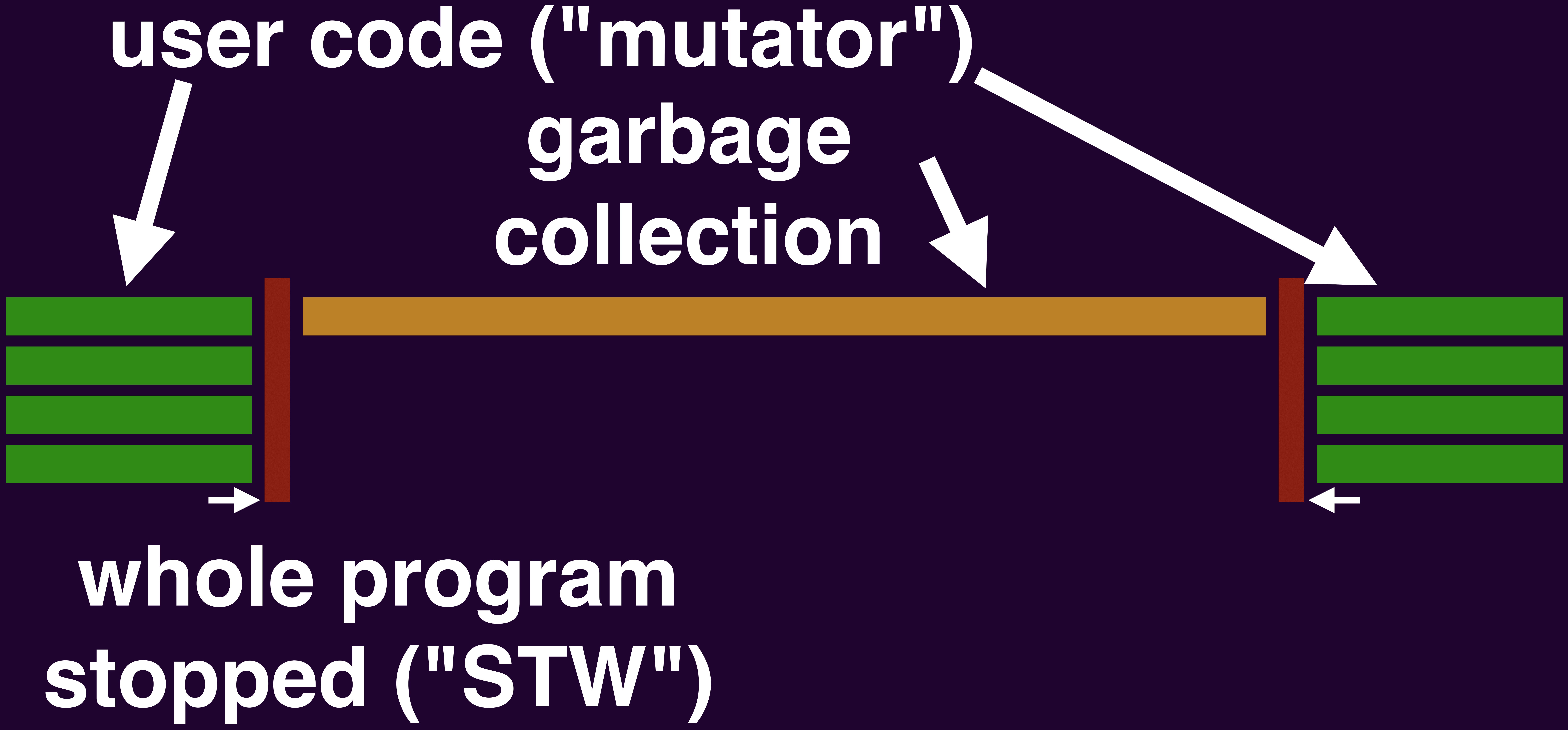
some behavior is *different* during GC

* this is probably bad, but how good should we expect?

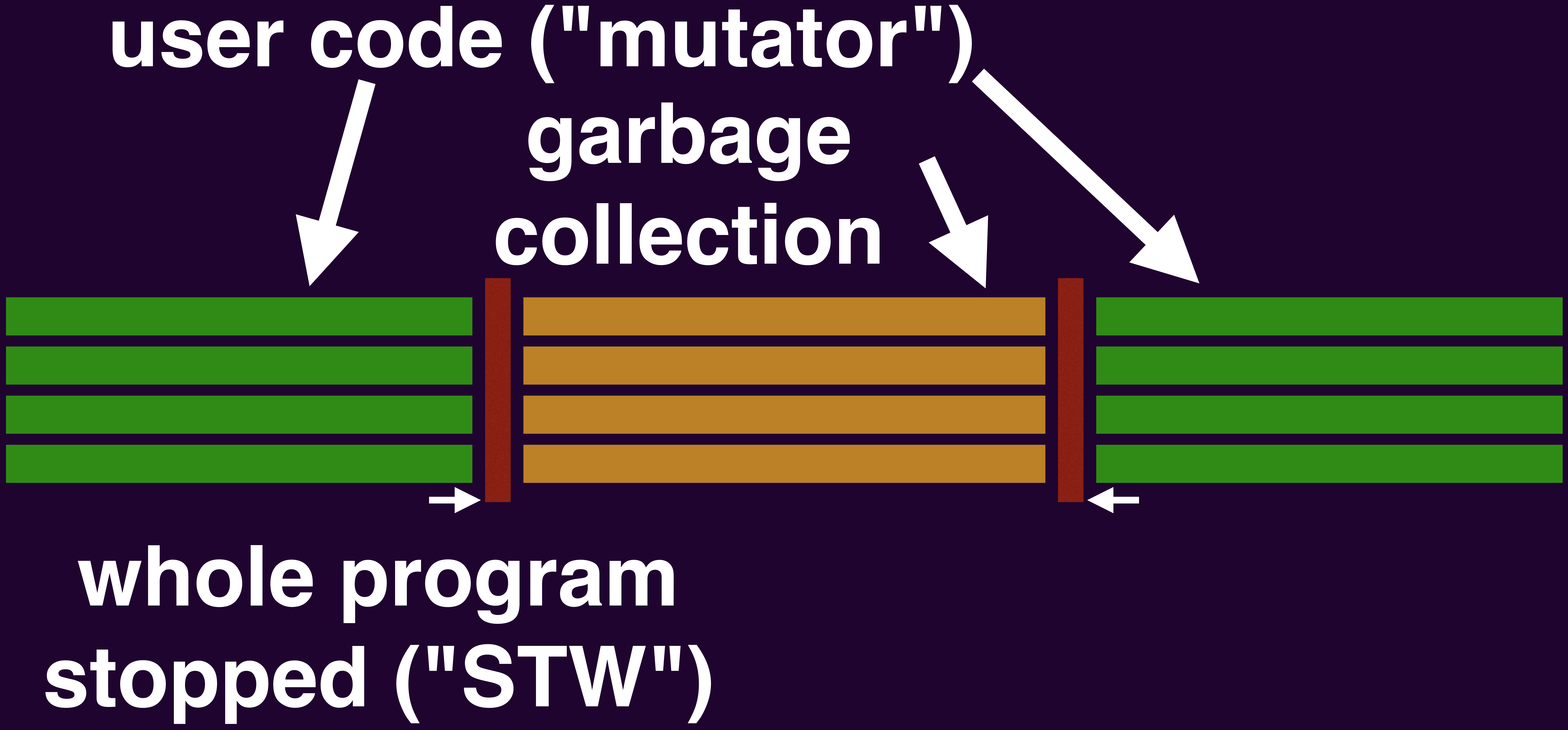
Go garbage collection timeline:

Go 1: program fully stopped
Go 1.1: GC uses parallel threads
Go 1.4: precise collection
Go 1.5: global pauses <10ms
Go 1.8: goroutine pauses <100µs

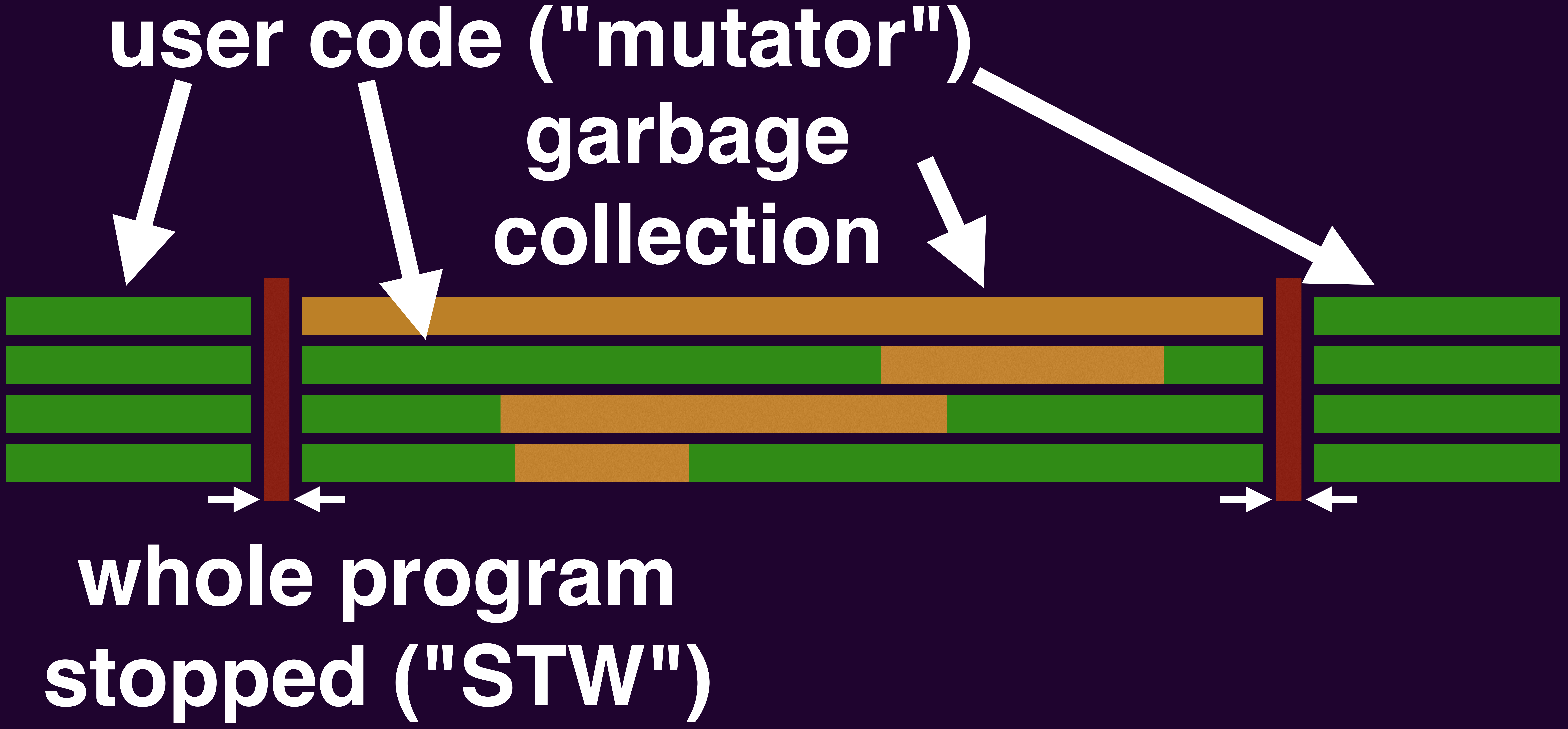
Go 1 GC


user code ("mutator")
garbage
collection
whole program
stopped ("STW")

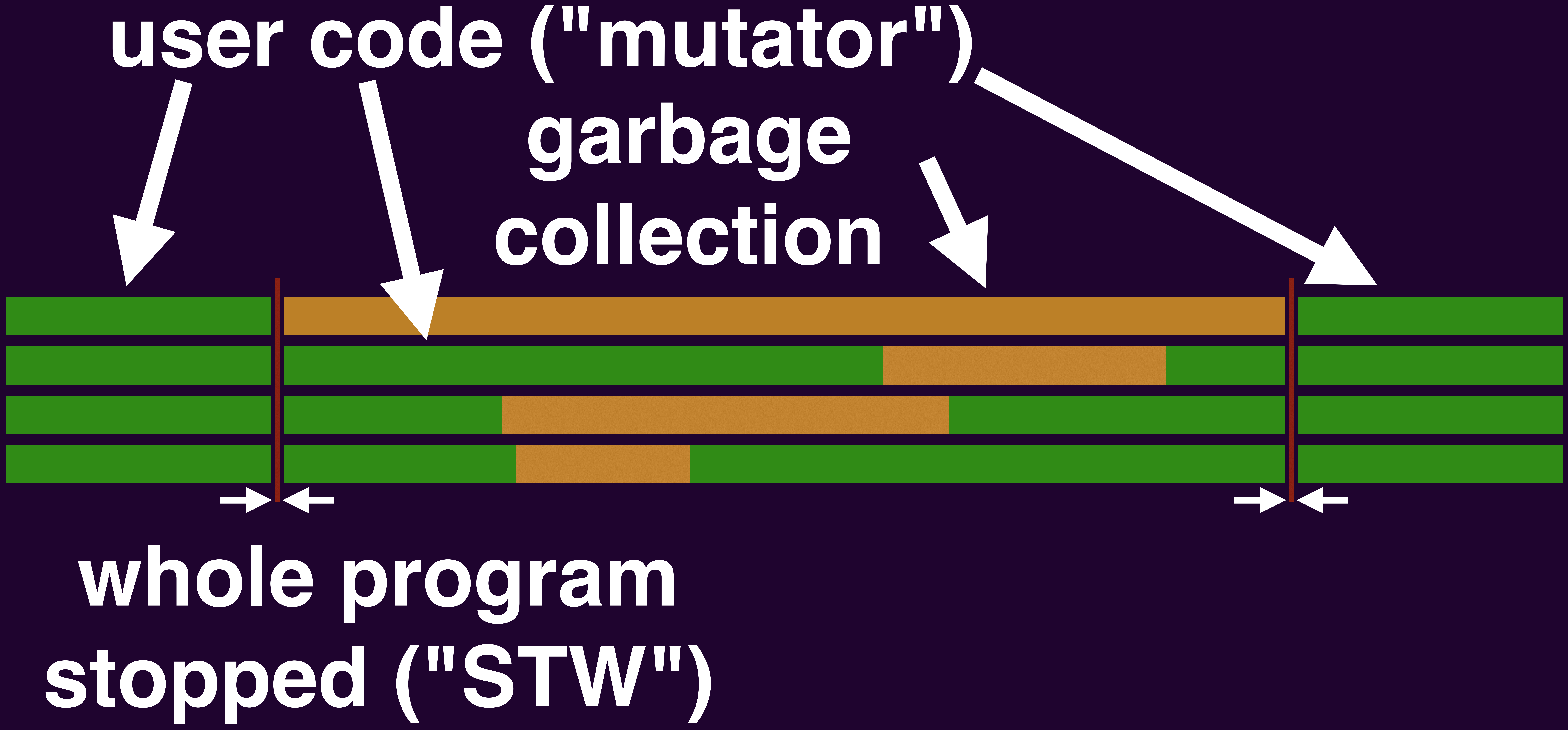
Go 1.1 GC


user code ("mutator")
garbage collection
whole program stopped ("STW")

Go 1.5 GC

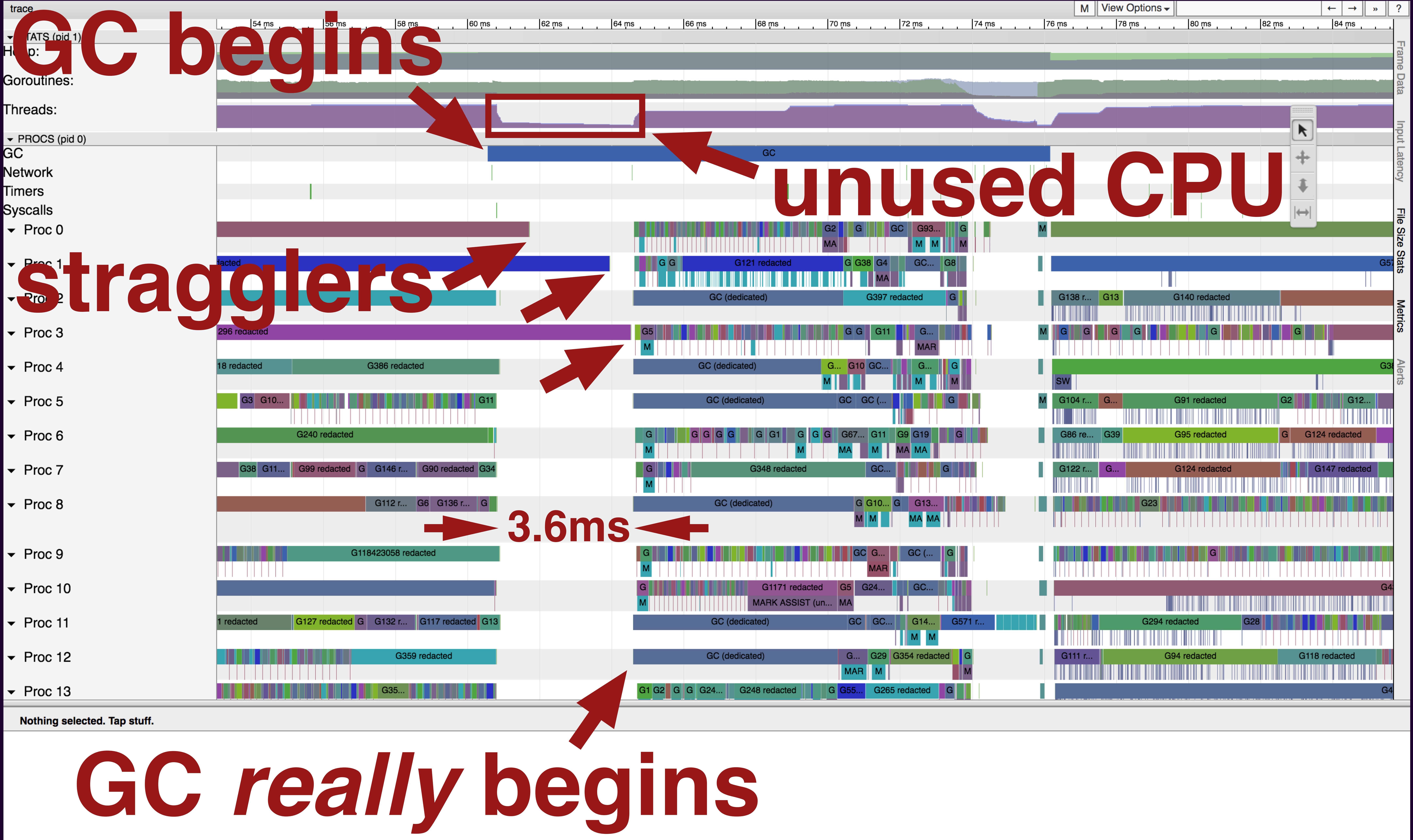

user code ("mutator")
garbage collection
whole program stopped ("STW")

Go 1.8 GC


user code ("mutator")
garbage collection
whole program stopped ("STW")

#3A: Stop-The-World pauses

Everything stops when the GC begins and ends mark phase
Stopping everything takes time


GC begins
unused CPU
stragglers
3.6ms
GC really begins
trace
View Options
STATS (pid 1)
Heap:
Goroutines:
Threads:
PROCS (pid 0)
GC
Network
Timers
Syscalls
Proc 0
Proc 1
Proc 2
Proc 3
Proc 4
Proc 5
Proc 6
Proc 7
Proc 8
Proc 9
Proc 10
Proc 11
Proc 12
Proc 13
GC (dedicated)
MARK ASSIST (un...
Nothing selected. Tap stuff.
Frame Data
Input Latency
File Size Stats
Metrics
Alerts

Goroutines can stop when...

✓ allocating memory
✓ calling functions
✓ communicating
✗ crunching numbers in a loop

Seen in:

- encoding/base64
- encoding/json
- .../golang/protobuf/proto

it's measurable for 1MB values

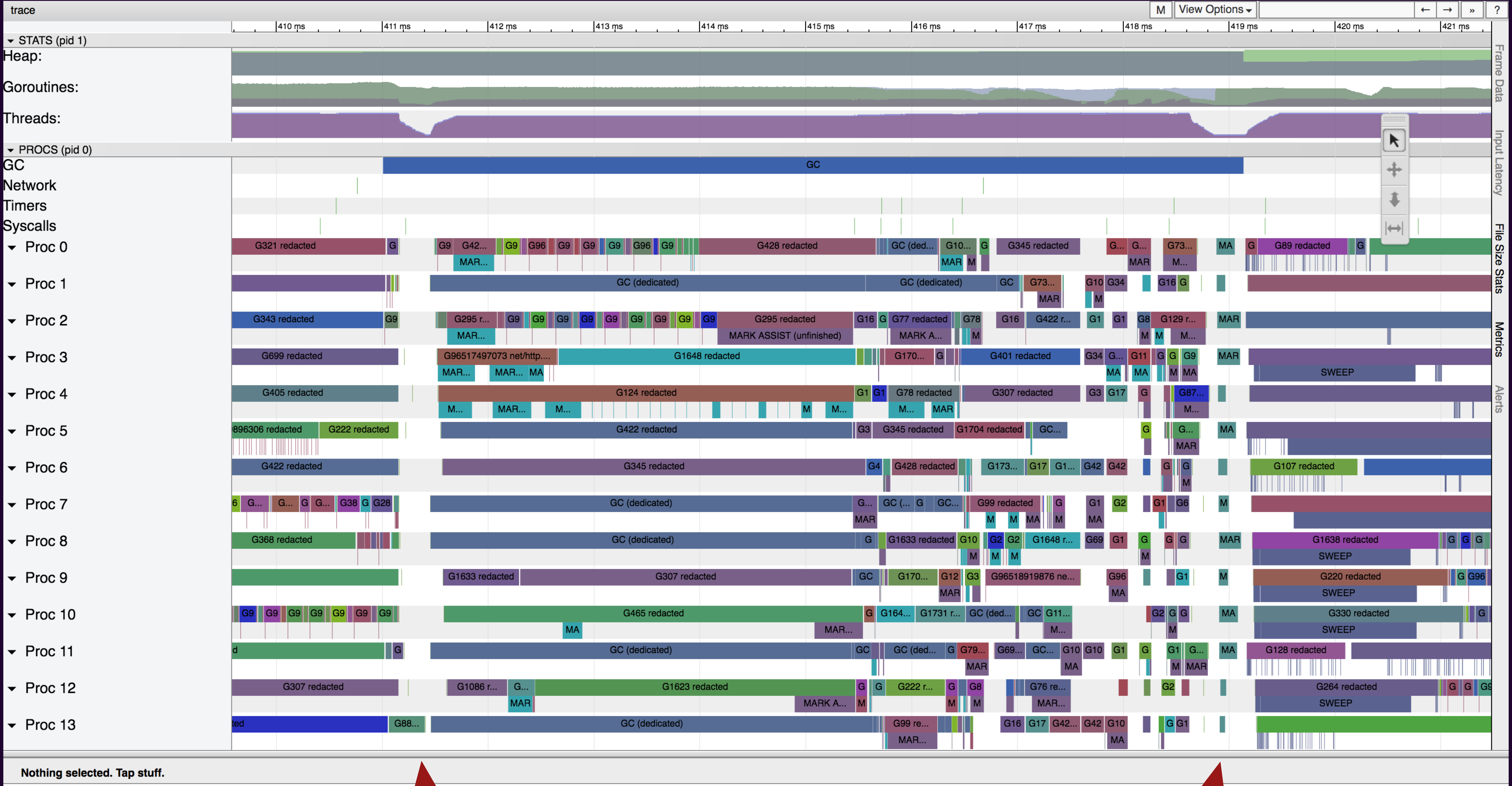
... if you're looking for it

*** check code lines before "End Stack Trace"**

Or write your own tight loop:

```
go func() {
  for i := 0; i < 1e9; i++ {
  }
}
```

golang.org/issue/10958
The Go 1.10 compiler should have a general, permanent fix
Workaround available now



quick pauses, less waste

#3B: Other awkward pauses

A mark/sweep GC:

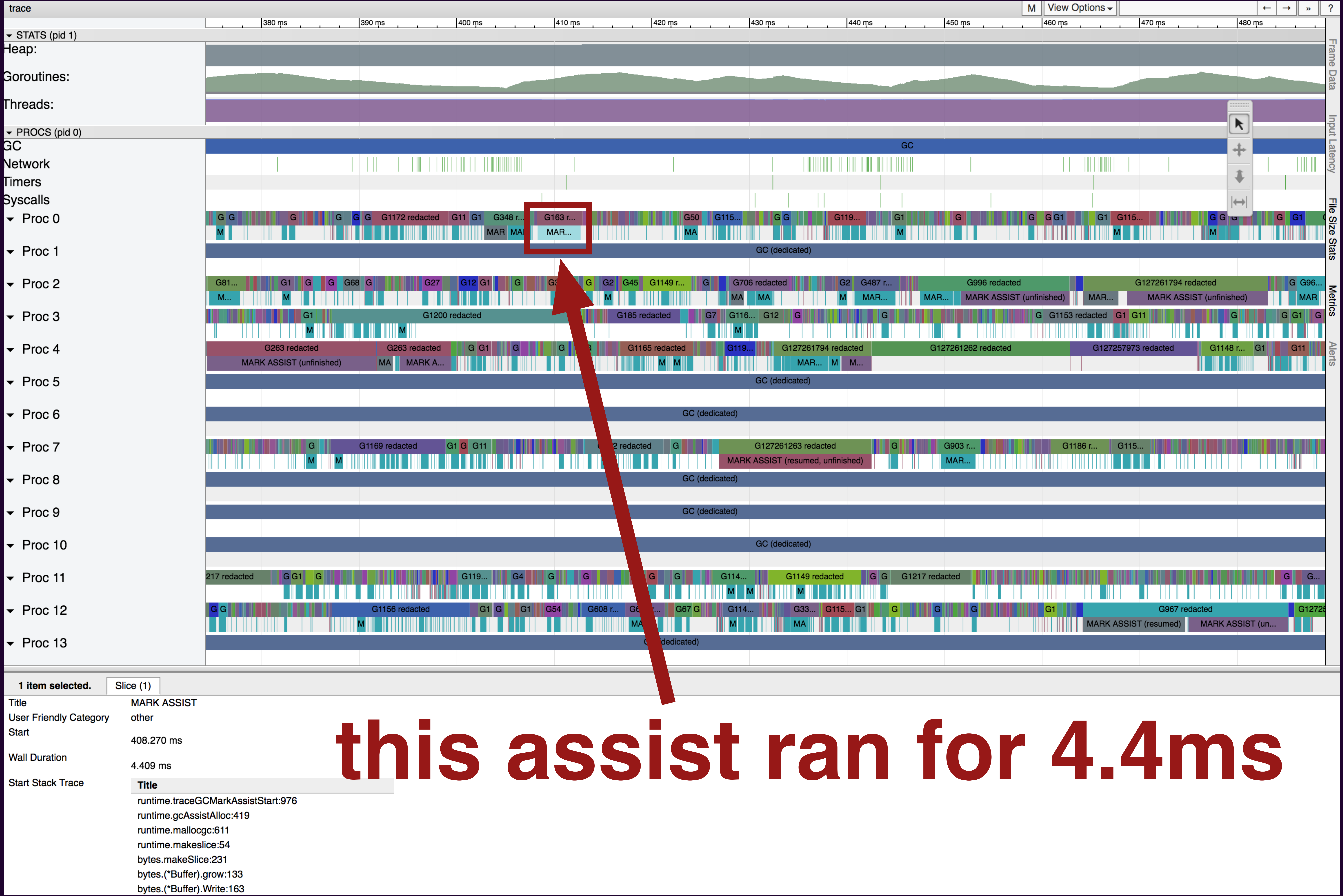
- "mark" finds in-use memory
- "sweep" reclaims the rest

GC needs to make progress
User code works against that
User code is forced to help out

user code in goroutines

Proc 11
Proc 12
Proc 13
G127260532 redacted
MARK ASSIST
runtime work below
read(2) syscall
helping the GC
* hold 'shift' to draw a selection box
1 item selected.
Slice (1)

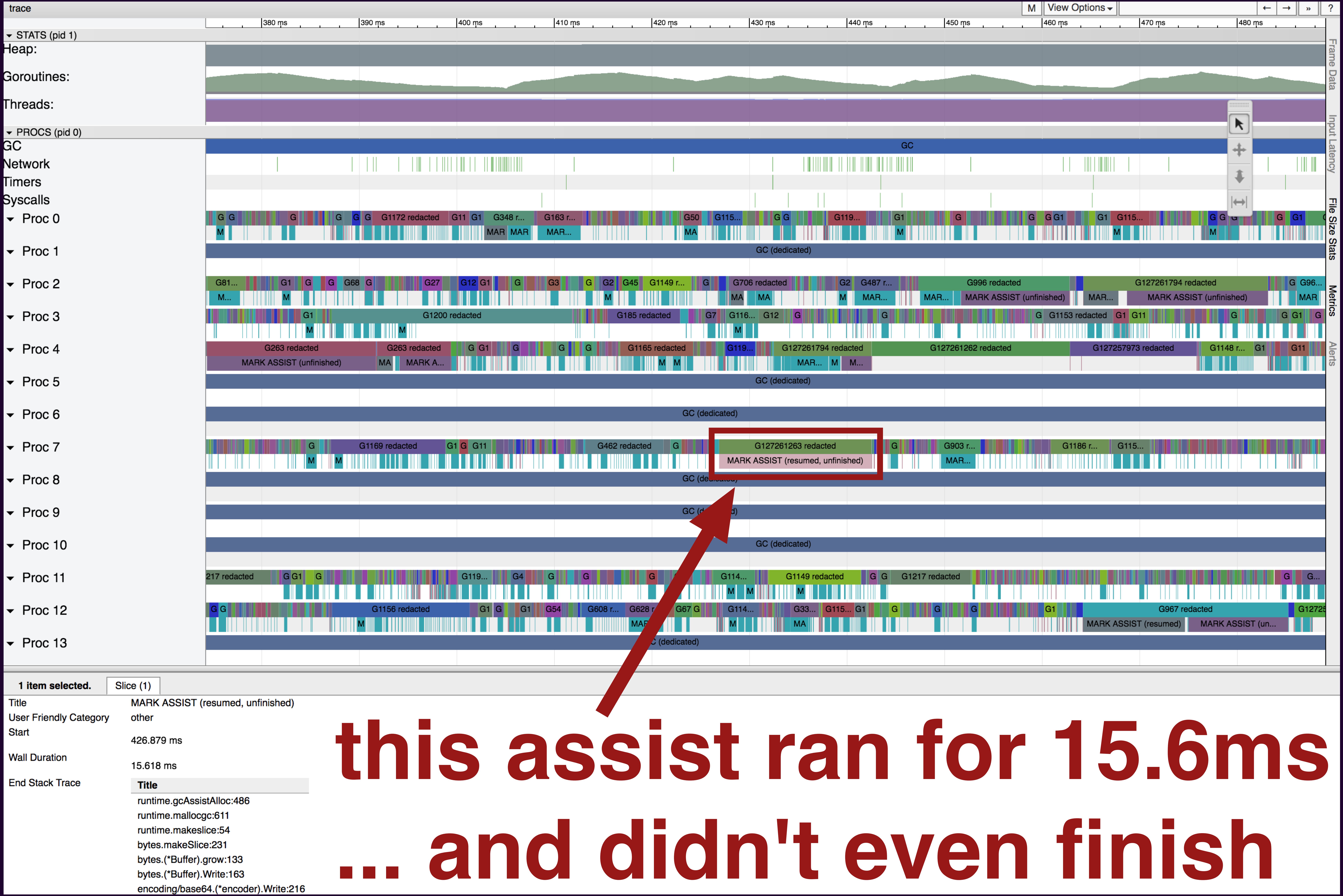
Title	syscall
User Friendly Category	other
Start	353.690 ms
Start Stack Trace	Title
	syscall.read:756
	syscall.Read:162
	internal/poll.(*FD).Read:121
	net.(*netFD).Read:204
	net.(*conn).Read:176
	bufio.(*Scanner).Scan:207
	redacted:0



1 item selected. Slice (1)

Title	MARK ASSIST
User Friendly Category	other
Start	408.270 ms
Wall Duration	4.409 ms
Start Stack Trace	**Title**
	runtime.traceGCMarkAssistStart:976
	runtime.gcAssistAlloc:419
	runtime.mallocgc:611
	runtime.makeslice:54
	bytes.makeSlice:231
	bytes.(*Buffer).grow:133
	bytes.(*Buffer).Write:163

this assist ran for 4.4ms



1 item selected. Slice (1)

Title	MARK ASSIST (resumed, unfinished)
User Friendly Category	other
Start	426.879 ms
Wall Duration	15.618 ms
End Stack Trace	

```
Title
runtime.gcAssistAlloc:486
runtime.mallocgc:611
runtime.makeslice:54
bytes.makeSlice:231
bytes.(*Buffer).grow:133
bytes.(*Buffer).Write:163
encoding/base64.(*encoder).Write:216
```

this assist ran for 15.6ms

… and didn't even finish

Most assists are well-deserved
But they start suddenly
Sweeping requires assists too
Don't allocate in critical paths?

go tool trace

http://.../debug/pprof/trace

How can the tool help you?

1. See time-dependent issues
2. Complements other profiles
3. Find latency improvements

Be prepared:

practice using the tools

@rhyshiltner Thank you!