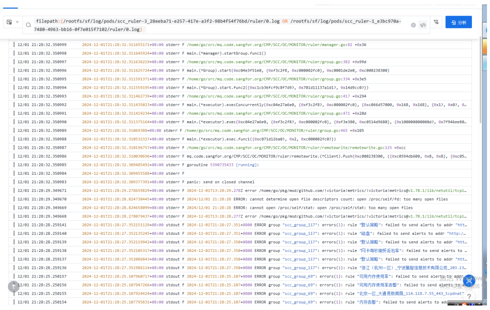
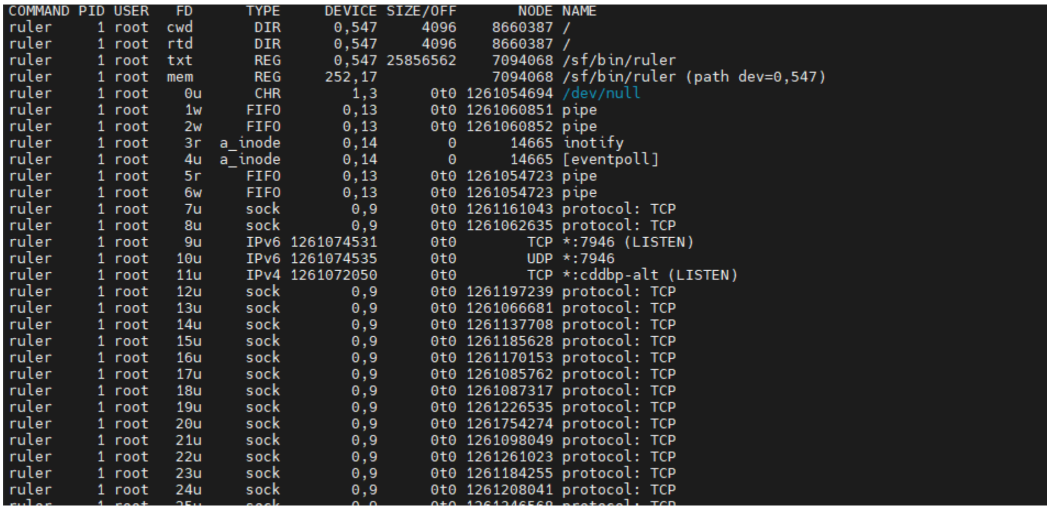
ruler 问题排查

Ruler 句柄数问题

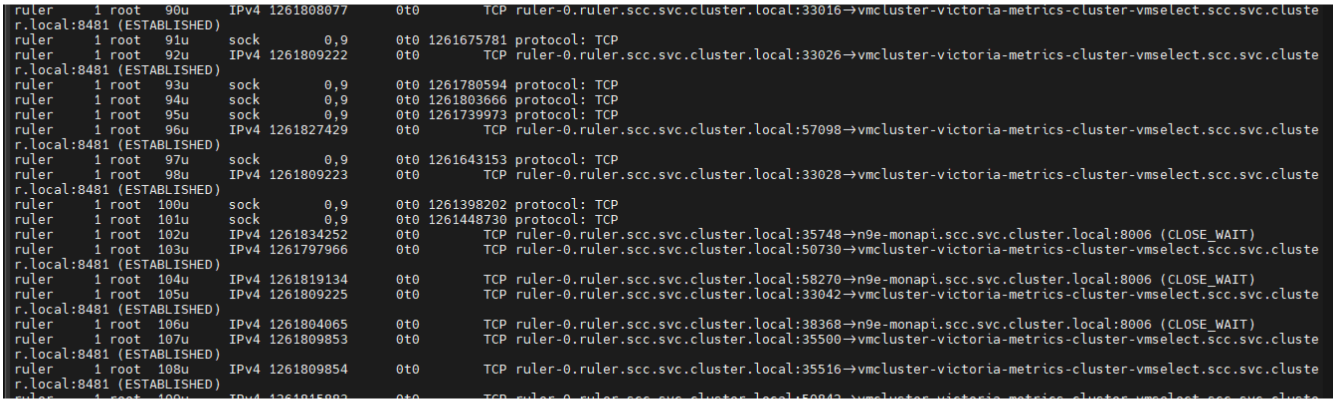


```
filepath:(/rootfs/sf/log/pods/scc_ruler-3_28eeba71-e257-417e-a3f2-98b4f54f76bd/ruler/0.log OR /rootfs/sf/log/pods/scc_ruler-1_e3bc970a-7480-4963-bb16-0f7e015f7102/ruler/0.log)

12/01 21:28:32.350999  2024-12-01T21:28:32.311655171+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/manager.go:82 +0x36
12/01 21:28:32.350998  2024-12-01T21:28:32.311645924+08:00 stderr F main.(*manager).startGroup.func1()
12/01 21:28:32.350997  2024-12-01T21:28:32.311636219+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/group.go:382 +0x99d
12/01 21:28:32.350996  2024-12-01T21:28:32.311625724+08:00 stderr F main.(*Group).start(0xc04e3f51e0, {0xf3c2f8, 0xc000002fc0}, 0xc0001de2e0, 0xc000238300)
12/01 21:28:32.350995  2024-12-01T21:28:32.311591371+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/group.go:334 +0x3e5
12/01 21:28:32.350994  2024-12-01T21:28:32.311559195+08:00 stderr F main.(*Group).start.func2({0xc1cb36fcf9c8f7d9?, 0x701d11137a1d1?, 0x14d9cc0?})
12/01 21:28:32.350993  2024-12-01T21:28:32.311462739+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/group.go:417 +0x294
12/01 21:28:32.350992  2024-12-01T21:28:32.311435023+08:00 stderr F main.(*executor).execConcurrently(0xc04e27a6e0, {0xf3c2f8?, 0xc000002fc0}, {0xc066d57000, 0x168, 0x168}, {0x1?, 0x0?, 0…
12/01 21:28:32.350991  2024-12-01T21:28:32.311419234+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/group.go:471 +0x20d
12/01 21:28:32.350990  2024-12-01T21:28:32.311375164+08:00 stderr F main.(*executor).exec(0xc04e27a6e0, {0xf3c2f8?, 0xc000002fc0}, {0xf3e380, 0xc0514d9680}, {0x100000000000d?, 0x7f946ee80…
12/01 21:28:32.350989  2024-12-01T21:28:32.31069388+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/group.go:465 +0x105
12/01 21:28:32.350988  2024-12-01T21:28:32.310532327+08:00 stderr F main.(*executor).exec.func1({0xc071d12ba0?, 0x2, 0xc000002fc0?})
12/01 21:28:32.350987  2024-12-01T21:28:32.310196757+08:00 stderr F /home/go/src/mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/remotewrite/remotewrite.go:125 +0xcc
12/01 21:28:32.350986  2024-12-01T21:28:32.310030696+08:00 stderr F mq.code.sangfor.org/CMP/SCC/OC/MONITOR/ruler/remotewrite.(*Client).Push(0xc000238300, {{0xc0594db600, 0x8, 0x8}, {0xc05…
12/01 21:28:32.350985  2024-12-01T21:28:32.309485492+08:00 stderr F goroutine 5390735433 [running]:
12/01 21:28:32.350984  2024-12-01T21:28:32.309455588+08:00 stderr F
12/01 21:28:32.350983  2024-12-01T21:28:32.309377391+08:00 stderr F panic: send on closed channel
12/01 21:28:29.349671  2024-12-01T21:28:29.278659829+08:00 stderr F 2024-12-01T13:28:29.278Z error /home/go/pkg/mod/github.com/!victoria!metrics/!victoria!metrics@v1.78.1/lib/netutil/tcpl…
12/01 21:28:29.349670  2024-12-01T21:28:28.824738445+08:00 stderr F 2024/12/01 21:28:28 ERROR: cannot determine open file descriptors count: open /proc/self/fd: too many open files
12/01 21:28:29.349669  2024-12-01T21:28:28.824658099+08:00 stderr F 2024/12/01 21:28:28 ERROR: cannot open /proc/self/stat: open /proc/self/stat: too many open files
12/01 21:28:29.349668  2024-12-01T21:28:28.278079437+08:00 stderr F 2024-12-01T13:28:28.277Z error /home/go/pkg/mod/github.com/!victoria!metrics/!victoria!metrics@v1.78.1/lib/netutil/tcpl…
12/01 21:28:28.259141  2024-12-01T21:28:27.352151129+08:00 stdout F 2024-12-01T21:28:27.351+0800 ERROR group "scc_group_117": errors(1): rule "默认策略": failed to send alerts to addr "htt…
12/01 21:28:28.259140  2024-12-01T21:28:27.352135245+08:00 stdout F 2024-12-01T21:28:27.351+0800 ERROR group "scc_group_117": errors(1): rule "磁盘": failed to send alerts to addr "http:/…
12/01 21:28:28.259139  2024-12-01T21:28:27.352119942+08:00 stdout F 2024-12-01T21:28:27.350+0800 ERROR group "scc_group_117": errors(1): rule "默认策略": failed to send alerts to addr "htt…
12/01 21:28:28.259138  2024-12-01T21:28:27.352105317+08:00 stdout F 2024-12-01T21:28:27.350+0800 ERROR group "scc_group_117": errors(1): rule "网卡每秒接受丢包率": failed to send alerts to …
12/01 21:28:28.259137  2024-12-01T21:28:27.352086843+08:00 stdout F 2024-12-01T21:28:27.350+0800 ERROR group "scc_group_117": errors(1): rule "默认策略": failed to send alerts to addr "htt…
12/01 21:28:28.259136  2024-12-01T21:28:27.351986114+08:00 stdout F 2024-12-01T21:28:27.350+0800 ERROR group "scc_group_117": errors(1): rule "浙江（杭州一区）_宁波腾智信息技术有限公司_203.13…
12/01 21:28:25.250157  2024-12-01T21:28:25.107960717+08:00 stdout F 2024-12-01T21:28:25.107+0800 ERROR group "scc_group_69": errors(1): rule "可用内存使用率": failed to send alerts to addr…
12/01 21:28:25.250156  2024-12-01T21:28:25.107947266+08:00 stdout F 2024-12-01T21:28:25.107+0800 ERROR group "scc_group_69": errors(1): rule "可用内存使用率告警": failed to send alerts to a…
12/01 21:28:25.250155  2024-12-01T21:28:25.107924424+08:00 stdout F 2024-12-01T21:28:25.107+0800 ERROR group "scc_group_69": errors(1): rule "北京一区_大通县教育局_114.118.7.55_443_tcpdnat"…
12/01 21:28:25.250154  2024-12-01T21:28:25.107795831+08:00 stdout F 2024-12-01T21:28:25.107+0800 ERROR group "scc_group_69": errors(1): rule "内存告警": failed to send alerts to add…
```

随着ruler的运行，根据 lsof -p 1 查看进程的打开的文件数，可见的在持续增长；



```
COMMAND PID USER   FD      TYPE     DEVICE SIZE/OFF       NODE NAME
ruler     1 root  cwd       DIR      0,547     4096    8660387 /
ruler     1 root  rtd       DIR      0,547     4096    8660387 /
ruler     1 root  txt       REG      0,547 25856562    7094068 /sf/bin/ruler
ruler     1 root  mem       REG     252,17             7094068 /sf/bin/ruler (path dev=0,547)
ruler     1 root   0u       CHR        1,3      0t0 1261054694 /dev/null
ruler     1 root   1w      FIFO       0,13      0t0 1261060851 pipe
ruler     1 root   2w      FIFO       0,13      0t0 1261060852 pipe
ruler     1 root   3r   a_inode       0,14        0      14665 inotify
ruler     1 root   4u   a_inode       0,14        0      14665 [eventpoll]
ruler     1 root   5r      FIFO       0,13      0t0 1261054723 pipe
ruler     1 root   6w      FIFO       0,13      0t0 1261054723 pipe
ruler     1 root   7u      sock        0,9      0t0 1261161043 protocol: TCP
ruler     1 root   8u      sock        0,9      0t0 1261062635 protocol: TCP
ruler     1 root   9u      IPv6 1261074531      0t0        TCP *:7946 (LISTEN)
ruler     1 root  10u      IPv6 1261074535      0t0        UDP *:7946
ruler     1 root  11u      IPv4 1261072050      0t0        TCP *:cddbp-alt (LISTEN)
ruler     1 root  12u      sock        0,9      0t0 1261197239 protocol: TCP
ruler     1 root  13u      sock        0,9      0t0 1261066681 protocol: TCP
ruler     1 root  14u      sock        0,9      0t0 1261137708 protocol: TCP
ruler     1 root  15u      sock        0,9      0t0 1261185628 protocol: TCP
ruler     1 root  16u      sock        0,9      0t0 1261170153 protocol: TCP
ruler     1 root  17u      sock        0,9      0t0 1261085762 protocol: TCP
ruler     1 root  18u      sock        0,9      0t0 1261087317 protocol: TCP
ruler     1 root  19u      sock        0,9      0t0 1261226535 protocol: TCP
ruler     1 root  20u      sock        0,9      0t0 1261754274 protocol: TCP
ruler     1 root  21u      sock        0,9      0t0 1261098049 protocol: TCP
ruler     1 root  22u      sock        0,9      0t0 1261261023 protocol: TCP
ruler     1 root  23u      sock        0,9      0t0 1261184255 protocol: TCP
ruler     1 root  24u      sock        0,9      0t0 1261208041 protocol: TCP
ruler     1 root  25u      sock        0,9      0t0 1261246568 protocol: TCP
```



```
ruler     1 root   90u     IPv4 1261808077      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:33016→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root   91u     sock        0,9      0t0 1261675781 protocol: TCP
ruler     1 root   92u     IPv4 1261809222      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:33026→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root   93u     sock        0,9      0t0 1261780594 protocol: TCP
ruler     1 root   94u     sock        0,9      0t0 1261803666 protocol: TCP
ruler     1 root   95u     sock        0,9      0t0 1261739973 protocol: TCP
ruler     1 root   96u     IPv4 1261827429      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:57098→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root   97u     sock        0,9      0t0 1261643153 protocol: TCP
ruler     1 root   98u     IPv4 1261809223      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:33028→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root  100u     sock        0,9      0t0 1261398202 protocol: TCP
ruler     1 root  101u     sock        0,9      0t0 1261448730 protocol: TCP
ruler     1 root  102u     IPv4 1261834252      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:35748→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler     1 root  103u     IPv4 1261797966      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:50730→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root  104u     IPv4 1261819134      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:58270→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler     1 root  105u     IPv4 1261809225      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:33042→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root  106u     IPv4 1261804065      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:38368→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler     1 root  107u     IPv4 1261809853      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:35500→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root  108u     IPv4 1261809854      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:35516→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
r.local:8481 (ESTABLISHED)
ruler     1 root  109u     IPv4 1261815883      0t0        TCP ruler-0.ruler.scc.svc.cluster.local:50842→vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluste
```

分析来看，ruler 在频繁地与 某个服务 建立 TCP 连接，并且频繁地打开和关闭 socket 套接字，由于建立连接的操作与关闭连接的操作频率不一致，从而使得连接数持续增加，最后超过 ulimit -Sn 和 ulimit -Hn 65535 的限制；大胆猜测所有的HTTP请求：

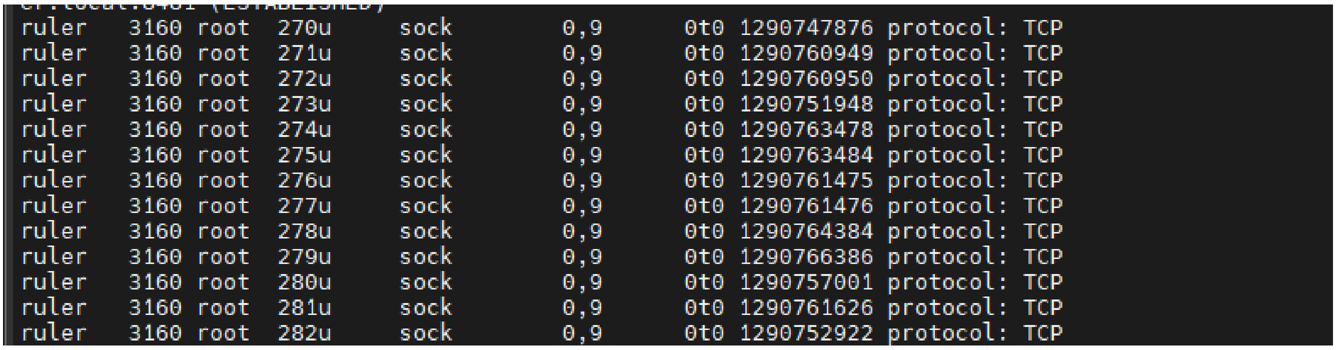
1、检查 vm 的相关请求的连接和代码

使用的自定义的连接池，并且连接都正常建立连接，看起来没啥问题

2、检查心跳接口相关的连接和代码

在ruler运行了一段时间后，通过 lsof -p [进程号] 查看其打开的文件连接，发现 vmselect 的TCP连接数较为稳定，连接数与 ruler 的在执行的组数基本是相同的

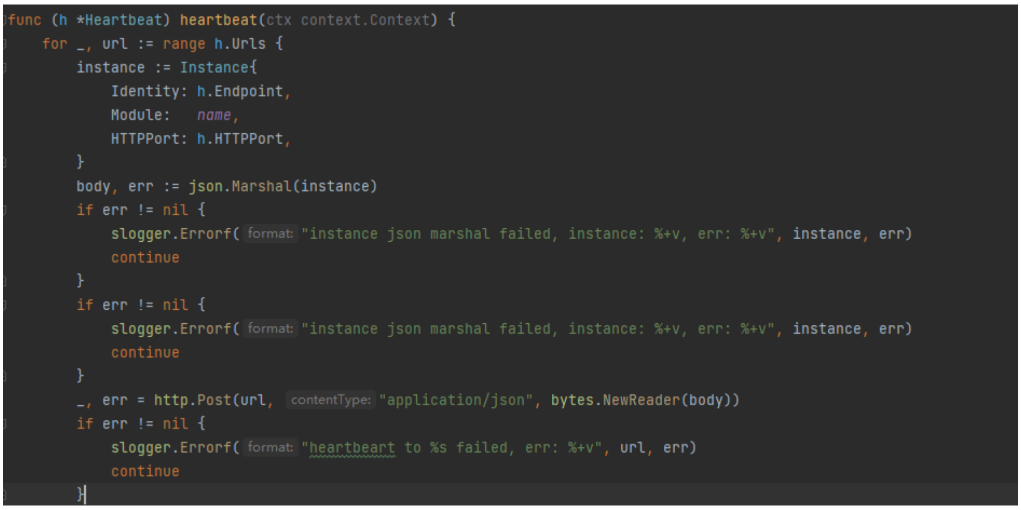
检查发现存在大量的 sock 连接，HTTP请求会建立TCP连接，从而打开socket进行读写，从这个角度去排查；



```
ruler   3160 root  270u     sock         0,9       0t0 1290747876 protocol: TCP
ruler   3160 root  271u     sock         0,9       0t0 1290760949 protocol: TCP
ruler   3160 root  272u     sock         0,9       0t0 1290760950 protocol: TCP
ruler   3160 root  273u     sock         0,9       0t0 1290751948 protocol: TCP
ruler   3160 root  274u     sock         0,9       0t0 1290763478 protocol: TCP
ruler   3160 root  275u     sock         0,9       0t0 1290763484 protocol: TCP
ruler   3160 root  276u     sock         0,9       0t0 1290761475 protocol: TCP
ruler   3160 root  277u     sock         0,9       0t0 1290761476 protocol: TCP
ruler   3160 root  278u     sock         0,9       0t0 1290764384 protocol: TCP
ruler   3160 root  279u     sock         0,9       0t0 1290766386 protocol: TCP
ruler   3160 root  280u     sock         0,9       0t0 1290757001 protocol: TCP
ruler   3160 root  281u     sock         0,9       0t0 1290761626 protocol: TCP
ruler   3160 root  282u     sock         0,9       0t0 1290752922 protocol: TCP
```

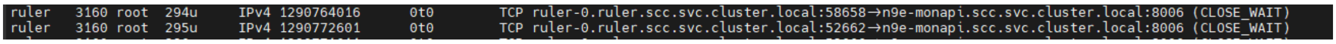
检视 心跳相关的代码，每30s执行一次，使用timer.ticker 进行定时调用，错误也处理了，一个简单的post心跳上报请求；

resp 也不读取，直接 _ 进行忽略即可，也不用close调，看起来没啥问题



```go
func (h *Heartbeat) heartbeat(ctx context.Context) {
    for _, url := range h.Urls {
        instance := Instance{
            Identity: h.Endpoint,
            Module:   name,
            HTTPPort: h.HTTPPort,
        }
        body, err := json.Marshal(instance)
        if err != nil {
            slogger.Errorf( format: "instance json marshal failed, instance: %+v, err: %+v", instance, err)
            continue
        }
        if err != nil {
            slogger.Errorf( format: "instance json marshal failed, instance: %+v, err: %+v", instance, err)
            continue
        }
        _, err = http.Post(url,  contentType: "application/json", bytes.NewReader(body))
        if err != nil {
            slogger.Errorf( format: "heartbeart to %s failed, err: %+v", url, err)
            continue
        }
```

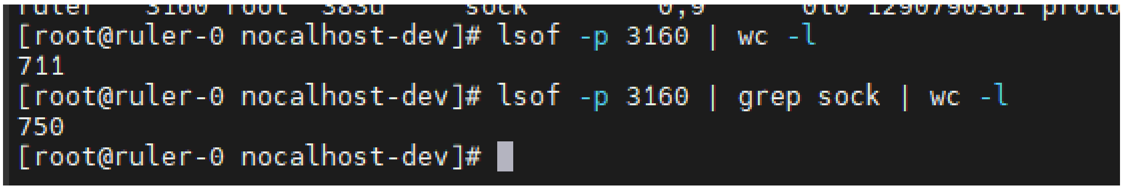
在 lsof -p [进程号] 中偶然发现与 n9e存在 CLOSE_WAIT 的连接，并且 src port 也不一样，也就是意味着心跳接口的 TCP 连接在重建？



```
ruler   3160 root  294u     IPv4 1290764016      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:58658→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  295u     IPv4 1290772601      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:52662→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
```

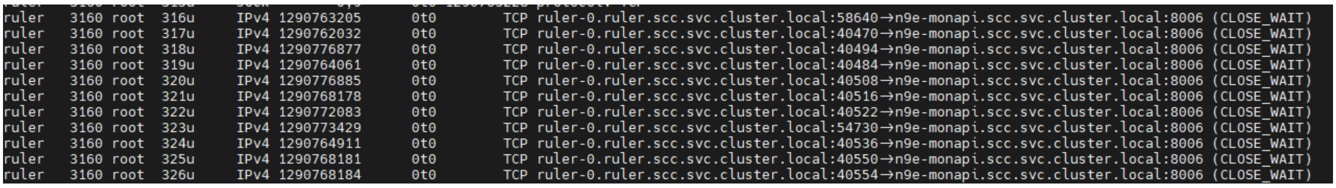
大胆猜测，http.Post 这个调用没有使用到长连接？

减少interval时间为1s，执行10m，再次 lsof 看下，发现存在大量的 sock 连接，整体打开的fd数上涨到了 711 多



```
[root@ruler-0 nocalhost-dev]# lsof -p 3160 | wc -l
711
[root@ruler-0 nocalhost-dev]# lsof -p 3160 | grep sock | wc -l
750
[root@ruler-0 nocalhost-dev]# 
```

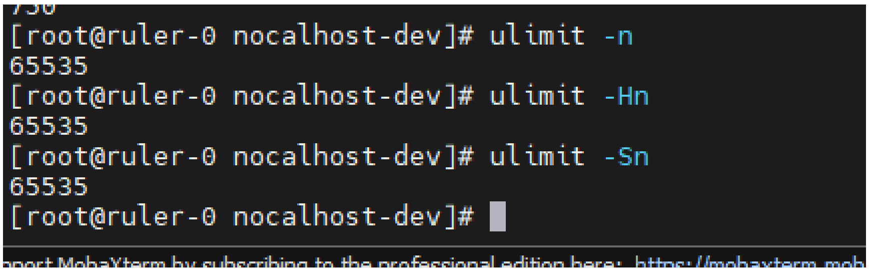
并且出现了大量的 close_wait 连接



```
ruler   3160 root  316u     IPv4 1290763205      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:58640→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  317u     IPv4 1290762032      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40470→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  318u     IPv4 1290776877      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40494→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  319u     IPv4 1290764061      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40484→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  320u     IPv4 1290776885      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40508→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  321u     IPv4 1290768178      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40516→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  322u     IPv4 1290772083      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40522→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  323u     IPv4 1290773429      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:54730→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  324u     IPv4 1290764911      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40536→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  325u     IPv4 1290768181      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40550→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
ruler   3160 root  326u     IPv4 1290768184      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:40554→n9e-monapi.scc.svc.cluster.local:8006 (CLOSE_WAIT)
```

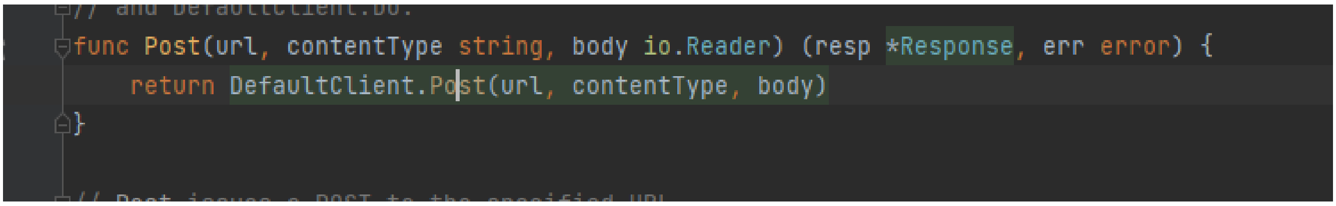
基本就能定位到是这个段代码的问题了，线上增长缓慢是因为interval为30s，fd会慢增加；

查看容器配置的最大可打开文件数为 65535，超过就会进行报错；

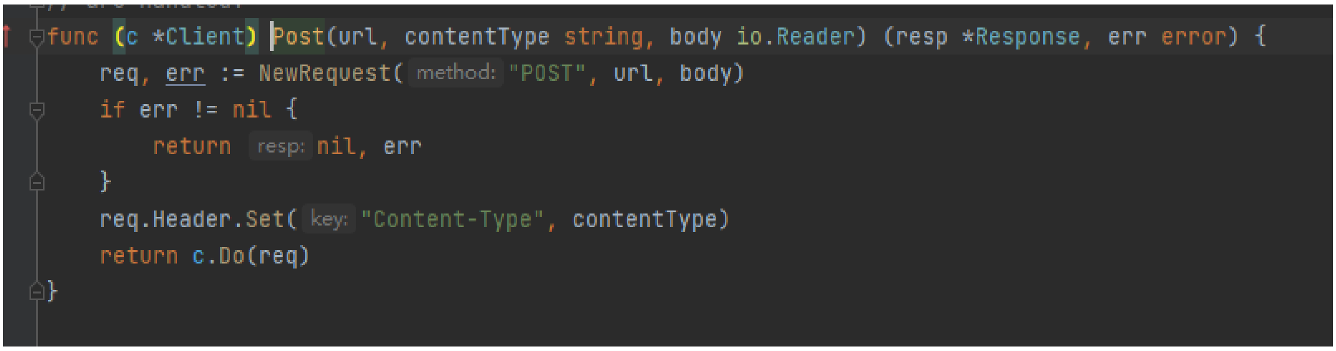


```
730
[root@ruler-0 nocalhost-dev]# ulimit -n
65535
[root@ruler-0 nocalhost-dev]# ulimit -Hn
65535
[root@ruler-0 nocalhost-dev]# ulimit -Sn
65535
[root@ruler-0 nocalhost-dev]# 
pport MobaXterm by subscribing to the professional edition here: https://mobaxterm.mob
```

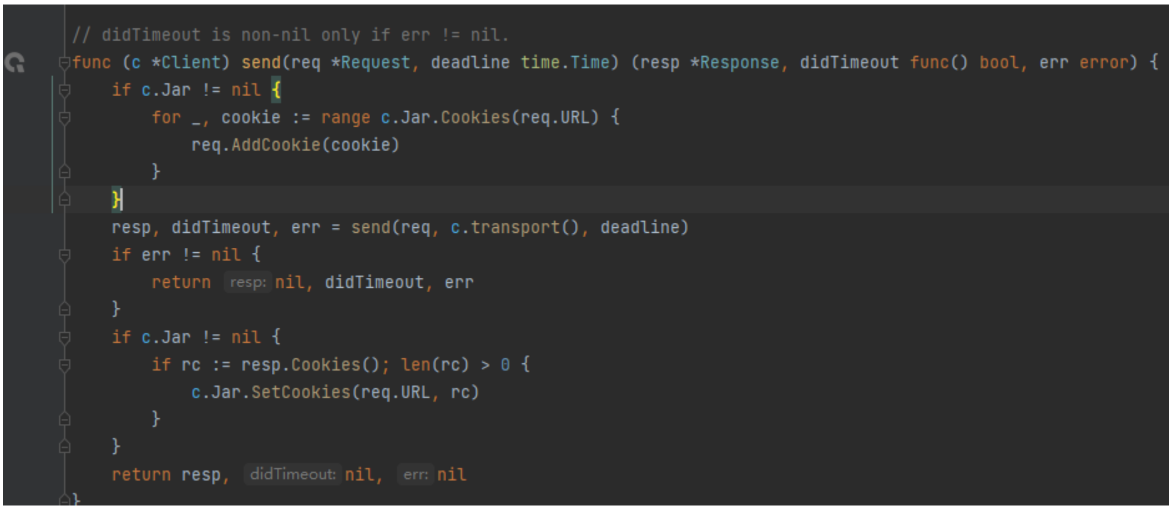
具体看下 net/http 的源码，



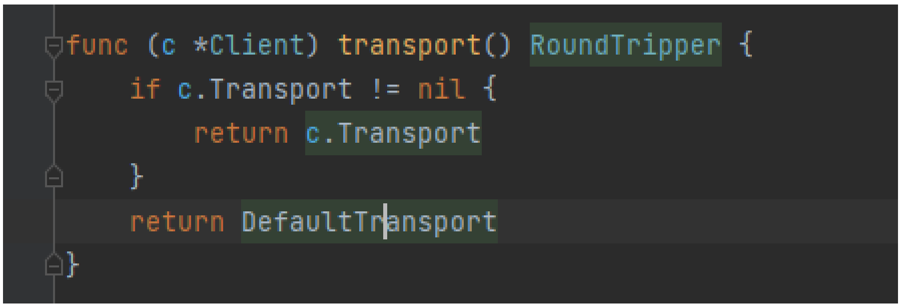
```go
// and DefaultClient.Do.
func Post(url, contentType string, body io.Reader) (resp *Response, err error) {
	return DefaultClient.Post(url, contentType, body)
}

// Post issues a POST to the specified URL.
```



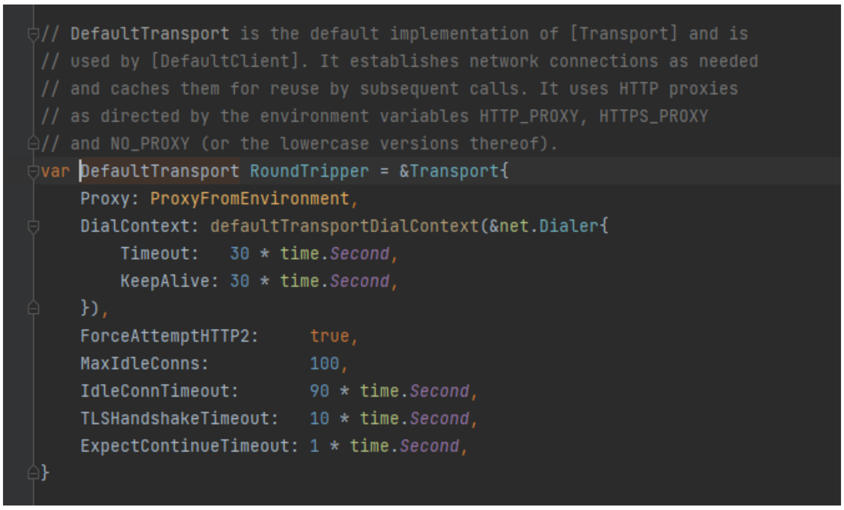
```go
// are handled.
func (c *Client) Post(url, contentType string, body io.Reader) (resp *Response, err error) {
	req, err := NewRequest( method: "POST", url, body)
	if err != nil {
		return  resp: nil, err
	}
	req.Header.Set( key: "Content-Type", contentType)
	return c.Do(req)
}
```



```go
// didTimeout is non-nil only if err != nil.
func (c *Client) send(req *Request, deadline time.Time) (resp *Response, didTimeout func() bool, err error) {
	if c.Jar != nil {
		for _, cookie := range c.Jar.Cookies(req.URL) {
			req.AddCookie(cookie)
		}
	}
	resp, didTimeout, err = send(req, c.transport(), deadline)
	if err != nil {
		return  resp: nil, didTimeout, err
	}
	if c.Jar != nil {
		if rc := resp.Cookies(); len(rc) > 0 {
			c.Jar.SetCookies(req.URL, rc)
		}
	}
	return resp,  didTimeout: nil,  err: nil
}
```



```go
func (c *Client) transport() RoundTripper {
	if c.Transport != nil {
		return c.Transport
	}
	return DefaultTransport
}
```

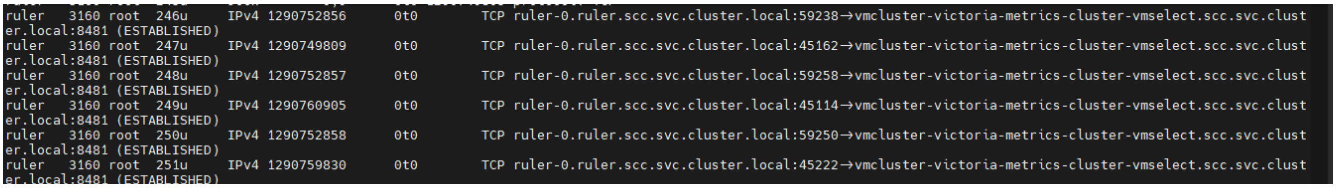


```go
// DefaultTransport is the default implementation of [Transport] and is
// used by [DefaultClient]. It establishes network connections as needed
// and caches them for reuse by subsequent calls. It uses HTTP proxies
// as directed by the environment variables HTTP_PROXY, HTTPS_PROXY
// and NO_PROXY (or the lowercase versions thereof).
var DefaultTransport RoundTripper = &Transport{
	Proxy: ProxyFromEnvironment,
	DialContext: defaultTransportDialContext(&net.Dialer{
		Timeout:   30 * time.Second,
		KeepAlive: 30 * time.Second,
	}),
	ForceAttemptHTTP2:     true,
	MaxIdleConns:          100,
	IdleConnTimeout:       90 * time.Second,
	TLSHandshakeTimeout:   10 * time.Second,
	ExpectContinueTimeout: 1 * time.Second,
}
```

http.Post的方法，使用了默认的连接池，也就是会有TCP连接被 reuse 的，但从实际的情况（大量的close_wait）来看，又没复用连接，为什么呢？

现在能大概猜测是没复用到长连接的问题，所以修改一下代码，使用短连接 connection为close看一下效果；

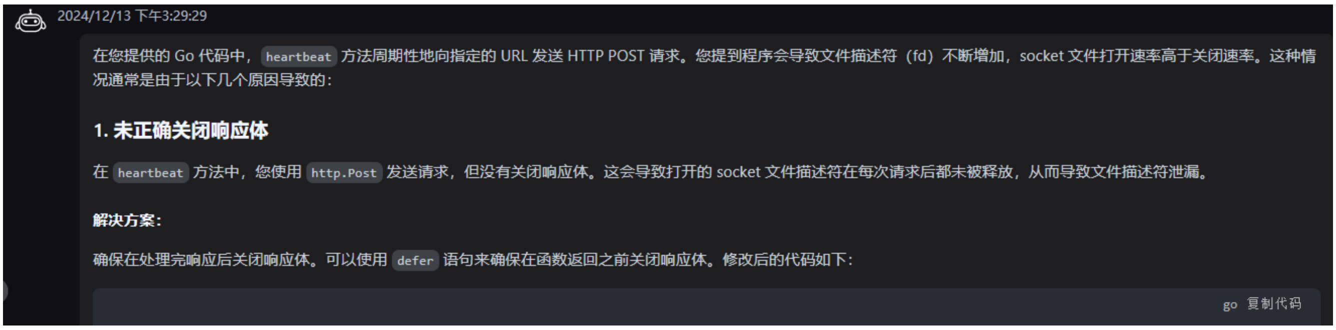
改完跑一段时间， lsof -p [进程号] 发现很正常，都是与 vmselect 建立的 ESTABLISH 连接，并且连接数固定在 50 左右；



```
ruler    3160 root  246u    IPv4 1290752856      0t0      TCP ruler-0.ruler.scc.svc.cluster.local:59238->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.clust
er.local:8481 (ESTABLISHED)
ruler    3160 root  247u    IPv4 1290749809      0t0      TCP ruler-0.ruler.scc.svc.cluster.local:45162->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.clust
er.local:8481 (ESTABLISHED)
ruler    3160 root  248u    IPv4 1290752857      0t0      TCP ruler-0.ruler.scc.svc.cluster.local:59258->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.clust
er.local:8481 (ESTABLISHED)
ruler    3160 root  249u    IPv4 1290760905      0t0      TCP ruler-0.ruler.scc.svc.cluster.local:45114->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.clust
er.local:8481 (ESTABLISHED)
ruler    3160 root  250u    IPv4 1290752858      0t0      TCP ruler-0.ruler.scc.svc.cluster.local:59250->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.clust
er.local:8481 (ESTABLISHED)
ruler    3160 root  251u    IPv4 1290759830      0t0      TCP ruler-0.ruler.scc.svc.cluster.local:45222->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.clust
er.local:8481 (ESTABLISHED)
```

范围就能确定是长连接没复用，导致不断建立新的TCP连接，导致socket被不断打开的问题，TCP 连接建立的速度比关闭的速度要快导致的；

ok，google和AI一下代码，看看有什么思路，重点： “未关闭响应体可能导致fd泄漏”



2024/12/13 下午3:29:29

在您提供的 Go 代码中，`heartbeat` 方法周期性地向指定的 URL 发送 HTTP POST 请求。您提到程序会导致文件描述符（fd）不断增加，socket 文件打开速率高于关闭速率。这种情况通常是由于以下几个原因导致的：

1. 未正确关闭响应体

在 `heartbeat` 方法中，您使用 `http.Post` 发送请求，但没有关闭响应体。这会导致打开的 socket 文件描述符在每次请求后都未被释放，从而导致文件描述符泄漏。

解决方案：

确保在处理完响应后关闭响应体。可以使用 `defer` 语句来确保在函数返回之前关闭响应体。修改后的代码如下：

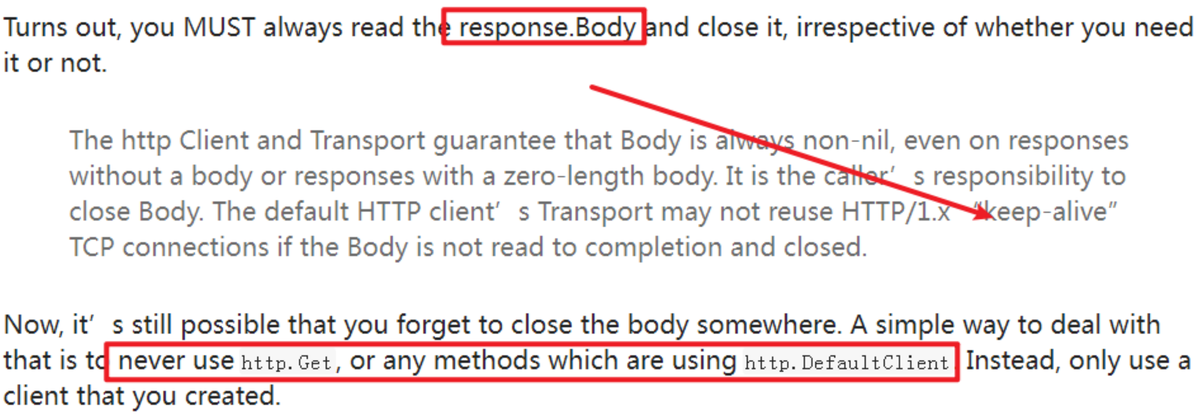
？！没有使用到 resp 进行 io.Read 也需要关闭吗？这个时候就得 google 一下发现了新世界

TIL: Go Response Body MUST be closed, even if you don’ t read it

https://manishrjain.com/must-close-golang-http-response

文章中提到了两个点：

- net/http 源码包中的 response.body 的描述，如果上一个body没有被读取或者close，则这个 TCP 连接，不会被 reuse
- 永远不要使用 HTTP.GET 以及 HTTP.DefaultClient，而应该使用自己创建的 httpclient

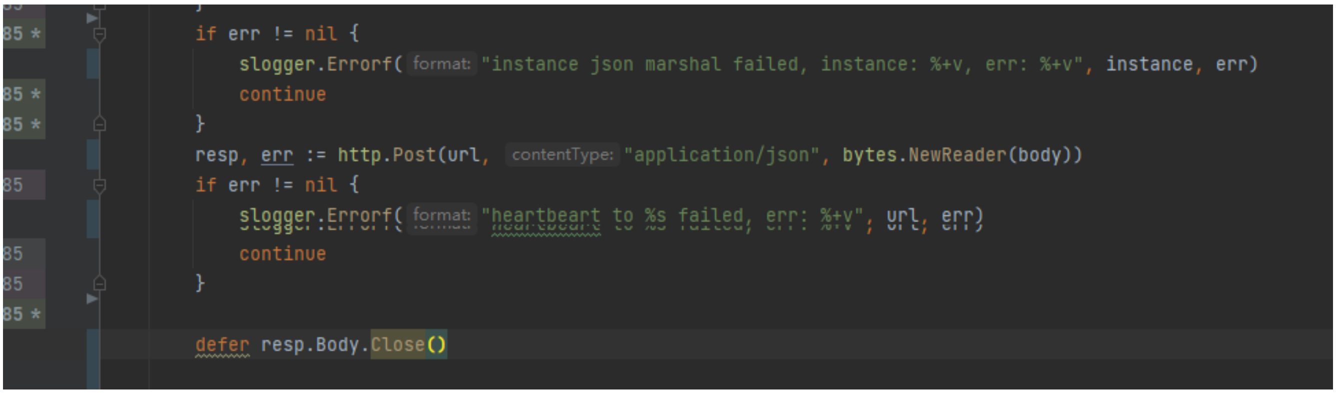


Turns out, you MUST always read the response.Body and close it, irrespective of whether you need it or not.

> The http Client and Transport guarantee that Body is always non-nil, even on responses without a body or responses with a zero-length body. It is the caller’ s responsibility to close Body. The default HTTP client’ s Transport may not reuse HTTP/1.x “keep-alive” TCP connections if the Body is not read to completion and closed.

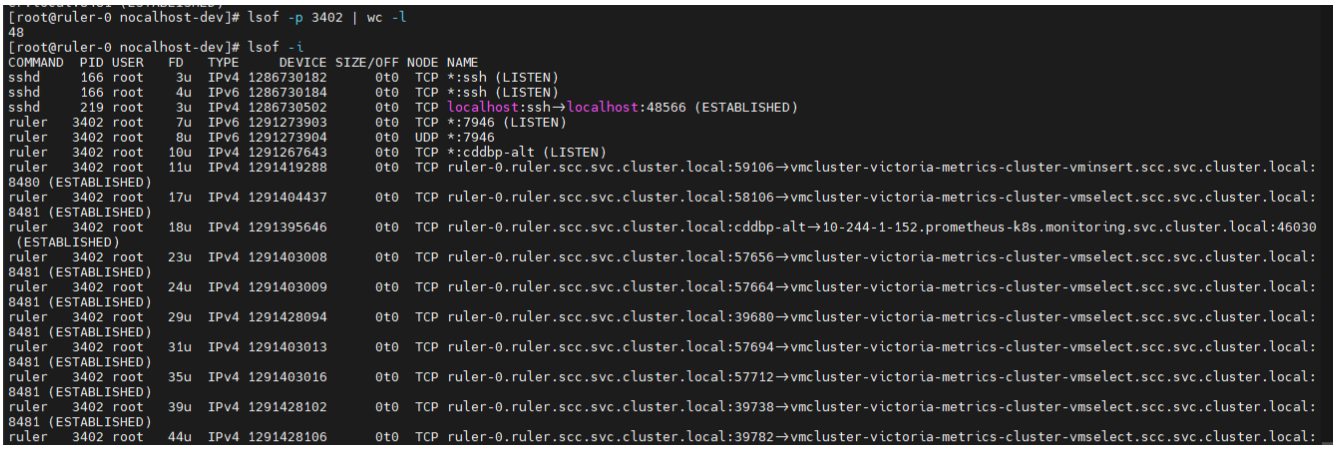
Now, it’ s still possible that you forget to close the body somewhere. A simple way to deal with that is to never use `http.Get`, or any methods which are using `http.DefaultClient`. Instead, only use a client that you created.

测试，修改 resp ，进行 defer close

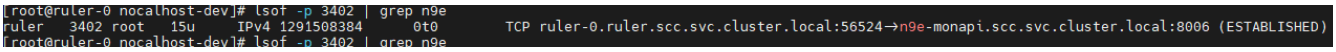


```
if err != nil {
    slogger.Errorf( format: "instance json marshal failed, instance: %+v, err: %+v", instance, err)
    continue
}
resp, err := http.Post(url,  contentType: "application/json", bytes.NewReader(body))
if err != nil {
    slogger.Errorf( format: "heartbeart to %s failed, err: %+v", url, err)
    continue
}

defer resp.Body.Close()
```

执行 ruler，运行半小时，检查 lsof -p [进程号] ，发现连接数正常了！，并且也有 ESTABLISH 的连接，需要疯狂 grep 才可能看到



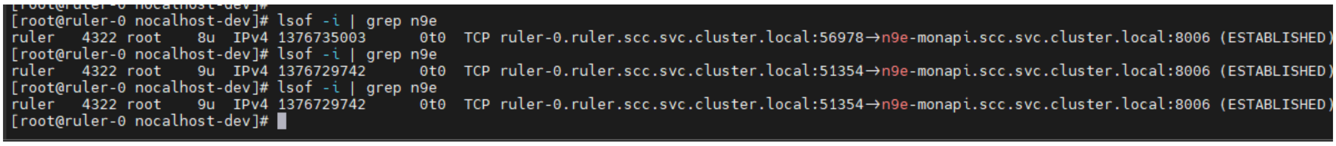
```
[root@ruler-0 nocalhost-dev]# lsof -p 3402 | wc -l
48
[root@ruler-0 nocalhost-dev]# lsof -i
COMMAND  PID USER   FD   TYPE     DEVICE SIZE/OFF NODE NAME
sshd     166 root    3u  IPv4 1286730182      0t0  TCP *:ssh (LISTEN)
sshd     166 root    4u  IPv6 1286730184      0t0  TCP *:ssh (LISTEN)
sshd     219 root    3u  IPv4 1286730502      0t0  TCP localhost:ssh->localhost:48566 (ESTABLISHED)
ruler   3402 root    7u  IPv6 1291273903      0t0  TCP *:7946 (LISTEN)
ruler   3402 root    8u  IPv6 1291273904      0t0  UDP *:7946
ruler   3402 root   10u  IPv4 1291267643      0t0  TCP *:cddbp-alt (LISTEN)
ruler   3402 root   11u  IPv4 1291419288      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:59106->vmcluster-victoria-metrics-cluster-vminsert.scc.svc.cluster.local:
8480 (ESTABLISHED)
ruler   3402 root   17u  IPv4 1291404437      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:58106->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   18u  IPv4 1291395646      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:cddbp-alt->10-244-1-152.prometheus-k8s.monitoring.svc.cluster.local:46030
 (ESTABLISHED)
ruler   3402 root   23u  IPv4 1291403008      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:57656->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   24u  IPv4 1291403009      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:57664->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   29u  IPv4 1291428094      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:39680->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   31u  IPv4 1291403013      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:57694->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   35u  IPv4 1291403016      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:57712->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   39u  IPv4 1291428102      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:39738->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
8481 (ESTABLISHED)
ruler   3402 root   44u  IPv4 1291428106      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:39782->vmcluster-victoria-metrics-cluster-vmselect.scc.svc.cluster.local:
```



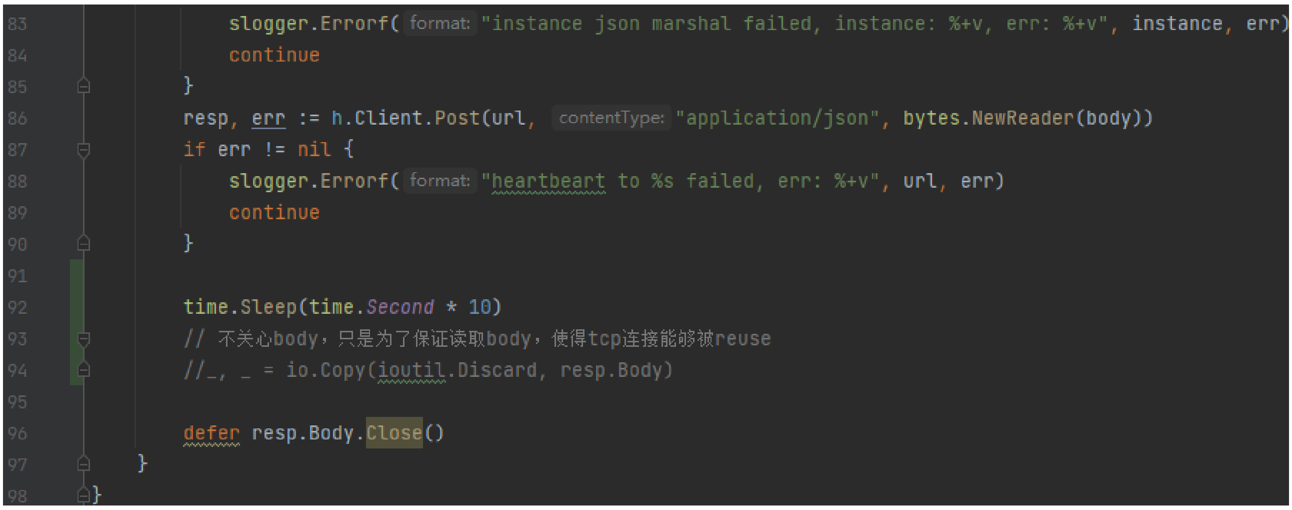
```
[root@ruler-0 nocalhost-dev]# lsof -p 3402 | grep n9e
ruler   3402 root   15u     IPv4 1291508384      0t0       TCP ruler-0.ruler.scc.svc.cluster.local:56524->n9e-monapi.scc.svc.cluster.local:8006 (ESTABLISHED)
[root@ruler-0 nocalhost-dev]# lsof -p 3402 | grep n9e
```

上面代码的写法还是有一些问题，如果你需要复用连接，则需要 读取完body，并且，关闭body，这样才是复用的；

否则，连接会建立后立刻被关闭掉，我们可以验证一下，在defer之前time.Sleep一下，看下连接是否处于建立中，在10s内，tcp连接的src port 为 56978，过了10s后，更换为了 51354，并看起来与猜想一致



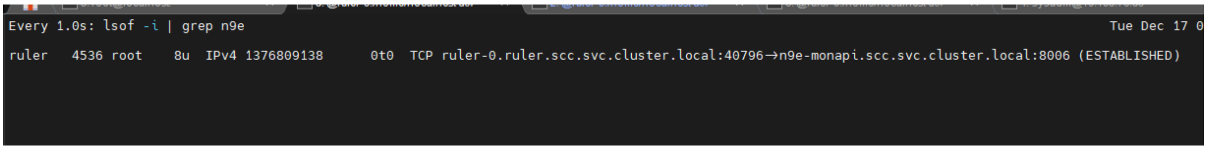
```
[root@ruler-0 nocalhost-dev]# lsof -i | grep n9e
ruler    4322 root    8u  IPv4 1376735003      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:56978->n9e-monapi.scc.svc.cluster.local:8006 (ESTABLISHED)
[root@ruler-0 nocalhost-dev]# lsof -i | grep n9e
ruler    4322 root    9u  IPv4 1376729742      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:51354->n9e-monapi.scc.svc.cluster.local:8006 (ESTABLISHED)
[root@ruler-0 nocalhost-dev]# lsof -i | grep n9e
ruler    4322 root    9u  IPv4 1376729742      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:51354->n9e-monapi.scc.svc.cluster.local:8006 (ESTABLISHED)
[root@ruler-0 nocalhost-dev]#
```



```
            slogger.Errorf( format: "instance json marshal failed, instance: %+v, err: %+v", instance, err)
            continue
        }
        resp, err := h.Client.Post(url,  contentType: "application/json", bytes.NewReader(body))
        if err != nil {
            slogger.Errorf( format: "heartbeart to %s failed, err: %+v", url, err)
            continue
        }

        time.Sleep(time.Second * 10)
        // 不关心body，只是为了保证读取body，使得tcp连接能够被reuse
        //_, _ = io.Copy(ioutil.Discard, resp.Body)

        defer resp.Body.Close()
    }
}
```

最后，我们使用比较正确的写法，读取完body，并且，关闭body， 验证一下是否一致复用一个连接

发现 src port 一直是 40796，说明复用的是同一个tcp连接

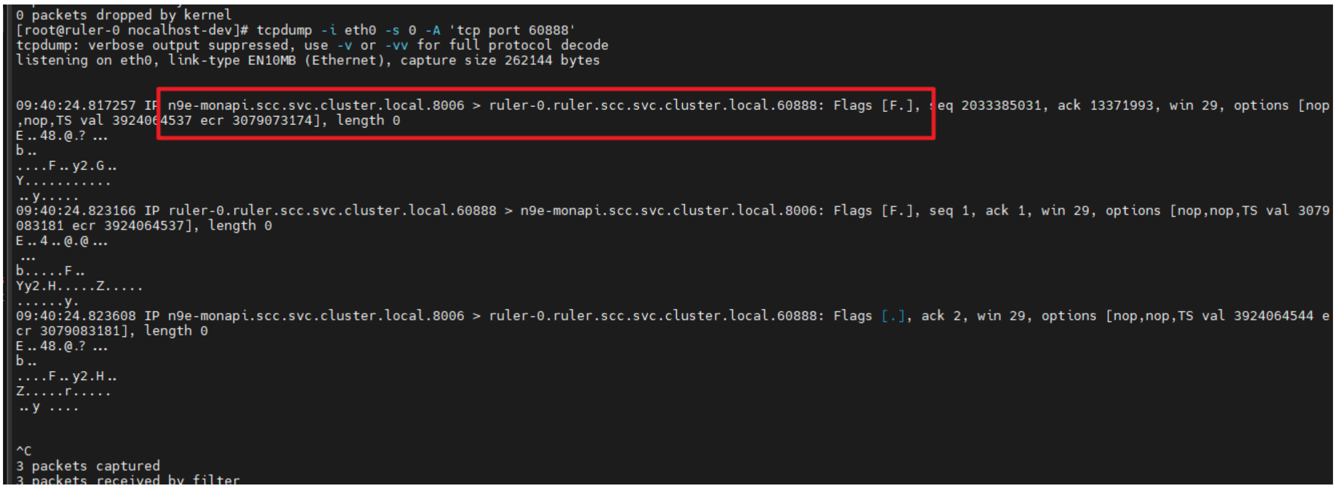


```
Every 1.0s: lsof -i | grep n9e                                                                                          Tue Dec 17 0

ruler    4536 root    8u  IPv4 1376809138      0t0  TCP ruler-0.ruler.scc.svc.cluster.local:40796→n9e-monapi.scc.svc.cluster.local:8006 (ESTABLISHED)
```

按照实际场景测试，30s触发一次心跳，发现tcp连接又不复用了，调整为5s又复用了，再次调整为 10s，发现有时候复用有时候不复用？

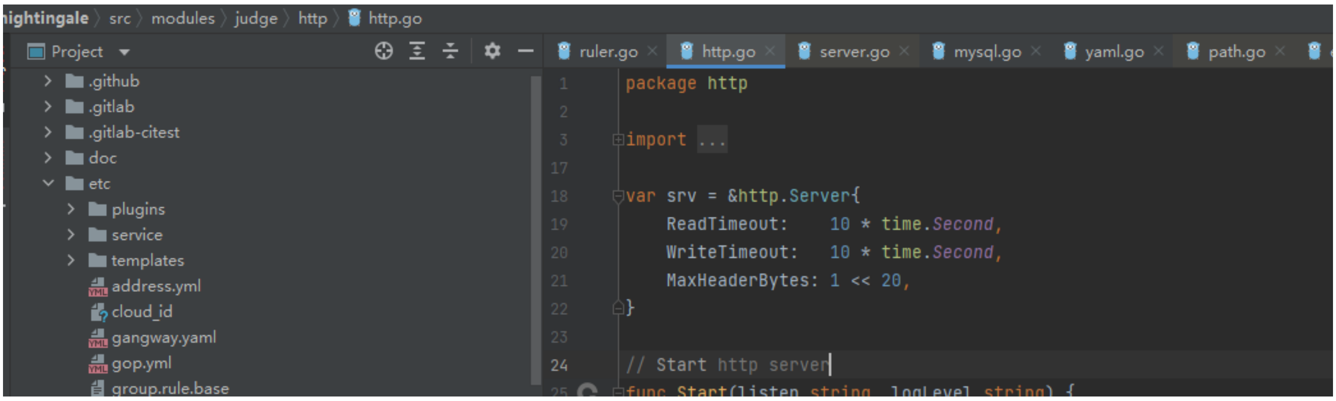
猜测是跟客户端的连接时间有关，检查一下客户端的初始化代码。检查了 transport 的idleconnTime 为 90s，也没啥问题，只能抓包看是哪边把连接关了

发现 n9e 给 ruler 发送了一个 FIN 关闭连接的报文，证明：是服务端主动关闭的连接



```
0 packets dropped by kernel
[root@ruler-0 nocalhost-dev]# tcpdump -i eth0 -s 0 -A 'tcp port 60888'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes


09:40:24.817257 IP n9e-monapi.scc.svc.cluster.local.8006 > ruler-0.ruler.scc.svc.cluster.local.60888: Flags [F.], seq 2033385031, ack 13371993, win 29, options [nop
,nop,TS val 3924064537 ecr 3079073174], length 0
E..48.@.? ...
b..
....F..y2.G..
Y..........
..y.....
09:40:24.823166 IP ruler-0.ruler.scc.svc.cluster.local.60888 > n9e-monapi.scc.svc.cluster.local.8006: Flags [F.], seq 1, ack 1, win 29, options [nop,nop,TS val 3079
083181 ecr 3924064537], length 0
E..4..@.@ ...
 ...
b.....F..
Yy2.H.....Z.....
......y.
09:40:24.823608 IP n9e-monapi.scc.svc.cluster.local.8006 > ruler-0.ruler.scc.svc.cluster.local.60888: Flags [.], ack 2, win 29, options [nop,nop,TS val 3924064544 e
cr 3079083181], length 0
E..48.@.? ...
b..
....F..y2.H..
Z.....r.....
 ..y ....


^C
3 packets captured
3 packets received by filter
```

检查 n9e 的代码，发现设置了http的read和write的timeout为 10s， 破案



```
package http

import ...

var srv = &http.Server{
    ReadTimeout:    10 * time.Second,
    WriteTimeout:   10 * time.Second,
    MaxHeaderBytes: 1 << 20,
}

// Start http server
func Start(listen string, logLevel string) {
```

结论

TIL: Go Response Body MUST be closed, even if you don’ t read it

1、即使未读取 resp，也必须要进行关闭

2、心跳和探活等接口使用http请求时，需要特别注意这种场景

3、无论你是否需要，都必须始终读 Body 并将其关闭

后话：

为什么不建议使用 http.defaultClient ?

- 初始化 defaultclient 的代码为： “var DefaultClient = &Client{}” ，也就是 timeout 会被默认设置为0，就是没有超时时间，导致TCP连接被挂起，从而导致fd泄漏

推荐阅读：

- https://medium.com/@nate510/don-t-use-go-s-default-http-client-4804cb19f779