[技术小贴士]: lua cosocket

Nov 2, 2014

从毕业之后的第一份工作开始，就接触了[Nginx](http://nginx.com/)，之后便开始了对nginx的漫漫学习之旅。在读nginx源码的过程，我学到了很多操作系统、数据结构、程序设计方面的知识。Nginx是我人生中第一次认真读懂的源代码（以前读研时看的hadoop，cassandra都只是看了点皮毛，有很多地方都没看懂），对我而言，Nginx不仅仅是一个web服务器，它更是我技术之路中的启蒙老师。

来到上海开始新的工作之后，我在[qingtingFM](http://www.qingting.fm/)接触到了[openresty](http://openresty.org/)，这是[agentzh](http://blog.sina.com.cn/s/articlelist_1834459124_0_1.html)大神对nginx打的一个包。openresty的一个非常显著的特性就是支持用[lua](http://www.lua.org/)写nginx配置，通过lua模块，可以非常容易地在nginx上实现很多功能。最近在openresty邮件列表中看到一个使用lua cosocket同上游服务器进行tcp连接的实例，感觉比较有学习价值：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | **local** sock **=** ngx.socket.tcp()  sock:settimeout(1000)  **local** ok, err **=** sock:connect("127.0.0.1", 13900)  **if** **not** ok **then**  ngx.say("failed to connect: ", err)  **return**  **end**  **local** bytes, err **=** sock:send("I am nginx")  **if** **not** bytes **then**  ngx.say("failed to send query: ", err)  **return**  **end**    **local** chunk,err **=** sock:receive()  **if** **not** chunk **then**  ngx.say("failed to receive a chunk: ", err)  **return**  **end**    ngx.log(ngx.DEBUG,"result: ", chunk)  **local** ok, err **=** sock:setkeepalive(0, 1)  **if** **not** ok **then**  ngx.say("failed to put the connection into pool "  **..** "with pool capacity 500 "  **..** "and maximal idle time 60 sec")  **return**  **end** |

这段代码的目的是在nginx上建立一个向上游的tcp长连接（sock:setkeepalive函数将当前套接字放入lua的内建连接池中）。作者在邮件列表中说，连接到后端tcp服务器，过一会儿就断了，无法保持住这个长连接，不知道为什么。这段代码看起来没问题，于是我写了一个tcp服务器来验证：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | **func** main() {  service **:=** "0.0.0.0:13900"  tcpAddr, err **:=** net**.**ResolveTCPAddr("tcp", service)  **if** err **!=** nil {  panic(err)  }  listener, err2 **:=** net**.**ListenTCP("tcp", tcpAddr)  **if** err2 **!=** nil {  panic(err2)  }  **var** wg sync**.**WaitGroup  ch **:=** make(**chan** net**.**Conn)  **go** **func**(ch **chan** net**.**Conn) {  wg**.**Add(1)  **for** {  conn, err **:=** listener**.**Accept()  **if** err **!=** nil {  fmt**.**Println("accept error: ", err)  } **else** {  fmt**.**Println("accept conn")  ch **<-** conn  }  }  wg**.**Done()  }(ch)  i **:=** 0  **for** {  i**++**  conn **:=** **<-**ch  **go** **func**(c net**.**Conn, id **int**) {  **var** buf [512]**byte**  **for** {  **if** \_, err **:=** c**.**Read(buf[**:**]); err **!=** nil {  fmt**.**Println("consumer", id, "err: ", err)  **return**  }  fmt**.**Println("consumer", id, "receive msg:", **string**(buf[**:**]))  */\* 注意这一行 \*/*  **if** n, err **:=** c**.**Write([]**byte**("consumer: " **+** strconv**.**Itoa(id))); err **!=** nil {  fmt**.**Println("consumer", i, "write error: ", err)  } **else** {  fmt**.**Println("write bytes: ", n)  }  }  }(conn, i)  }  wg**.**Wait()  } |

使用这个go服务器，确实会发现nginx无法keep住这个长连接，而且发现返回的数据nginx也不能及时收到，总是timeout。 最后发现，其实问题是出在lua的sock:receive这个函数上。原来，这里的lua cosocket的receive语义是收到一个’\n’或者直到缓冲区满才返回（行缓冲），而我后端的服务器发送的数据没有以’\n’结尾，所以这里receive不返回，继续等待接收数据（注意，ngx的lua模块全部是使用cosocket，因此receive不返回并不会把整个ngx阻塞住）。如果要接收二进制数据，则需要指定receive的数据大小，例如：sock:receive(10)。因此如果是接收二进制数据，通常是先接收固定长度的头部，然后解析出body的长度，再去接收body。后来我又去拜读了agentzh大神的lua-resty-[mysql|redis|memcache]-module，证实了自己的猜想。

这里有一个pull request是lua cosocket支持BSD语义的recv： <https://github.com/openresty/lua-nginx-module/pull/290>

另外这里有一个现成的lua http库: [lua-resty-http-simple](https://github.com/bakins/lua-resty-http-simple)可以用于解析http请求。