

47. 案例篇



服務器總是時不時丟包, 我該怎麼辦?(上)

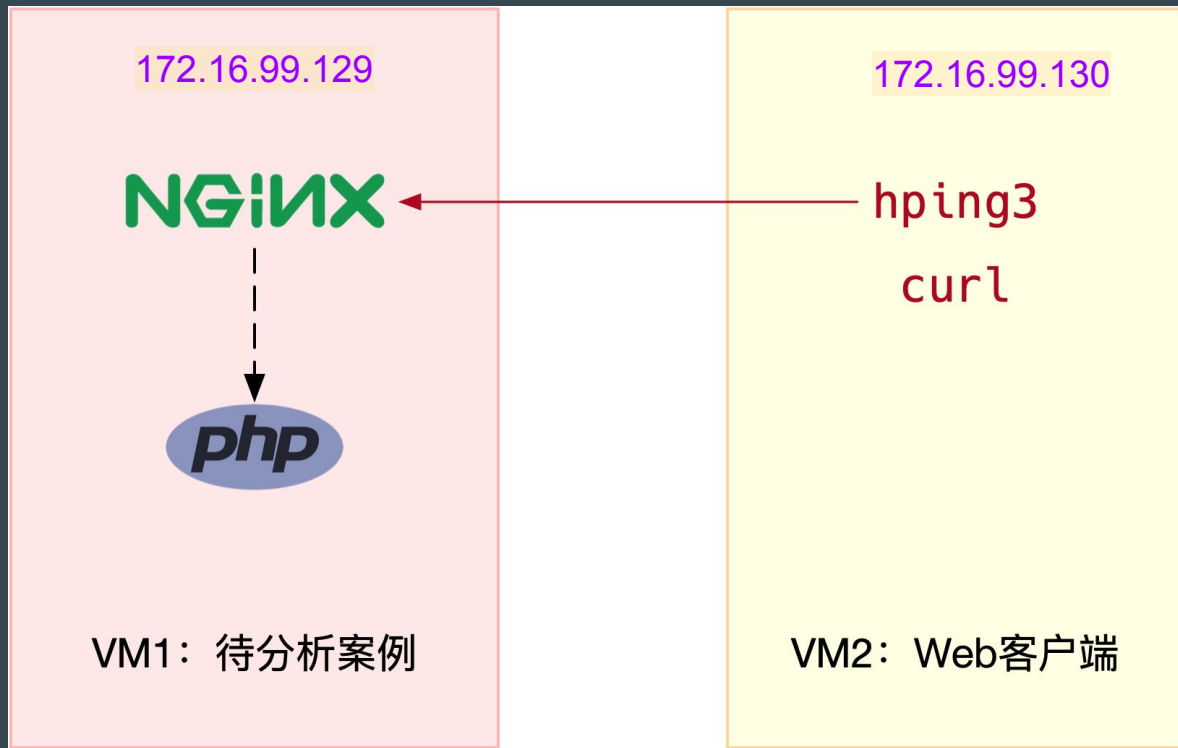
Jerry

- 為什麼會丟包
- 案例分析
- 小結
- 思考

為什麼會丟包

- 在傳輸的過程中，因為各種原因，而導致封包沒送達應用。
- 丟包率
 - 被丟掉的封包占比。
- 對 TCP 來說，丟包會造成壅塞和重傳，進而造成延遲增加和吞吐降低。

案例分析



環境

- VM-1 (Nginx + PHP):
 - `docker run --name nginx --hostname nginx --privileged -p 80:80 -itd feisky/nginx:drop`
- VM-2 (hping3 + curl)
 - `apt install docker.io curl hping3`

客戶端(VM2): 確認Nginx是否可訪問

- 使用 hping3
 - `hping3 -c 10 -S -p 80 172.16.99.129`
 - `-c 10` //發送 10 個請求
 - `-S` //使用 TCP SYN
 - `-p 80` //使用 80 port

RTT 起伏太大

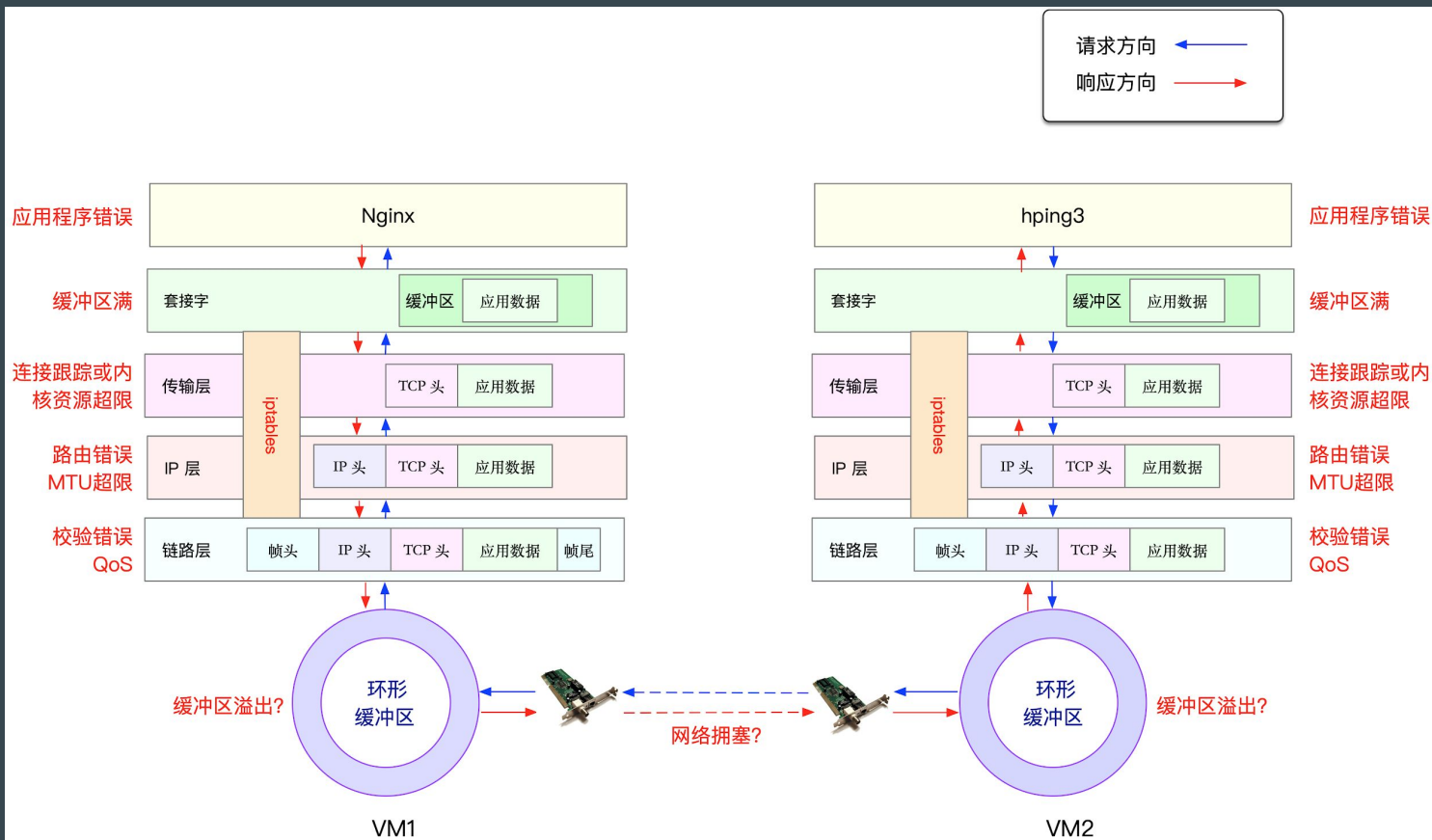
```
[root@demo-2:~# hping3 -c 10 -S -p 80 172.16.99.129
HPING 172.16.99.129 (ens33 172.16.99.129): S set, 40 headers + 0 data bytes
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=1 win=65535 rtt=2061.7 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=2 win=65535 rtt=1061.3 ms
DUP! len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=1 win=65535 rtt=2061.8 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=6 win=65535 rtt=3.1 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=5 win=65535 rtt=1035.7 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=4 win=65535 rtt=3024.1 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=9 win=65535 rtt=3.7 ms

--- 172.16.99.129 hping statistic ---
10 packets transmitted, 7 packets received, 30% packet loss
round-trip min/avg/max = 3.1/1321.6/3024.1 ms
```

只收到 7 個回覆, 30% 都掉包了

較高的 RTT 可能因為掉包重傳

可能的原因



鏈路層

- netstat -i

Ring Buffer溢出導致的丟包數

進入Ring Buffer後的丟包數

接收總數

總錯誤數

```
[root@nginx:/# netstat -i
Kernel Interface table
Iface      MTU      RX-OK    RX-ERR    RX-DRP    RX-OVR      TX-OK    TX-ERR    TX-DRP    TX-OVR    Flg
eth0       100      47       0         0 0       12        0         0         0        0 BMRU
lo         65536    0        0         0 0         0         0         0         0        0 LRU
```

```
[root@nginx:/# tc -s qdisc show dev eth0
qdisc netem 8002: root refcnt 2 limit 1000 loss 30%
  Sent 664 bytes 12 pkt (dropped 6, overlimits 0 requeues 0)
  backlog 0b 0p_requeues 0
```

- `tc -s qdisc show dev eth0`
- 从 `tc` 的输出中可以看到，`eth0` 上面配置了一个网络模拟排队规则(`qdisc netem`)，并且配置了丢包率为 30%(`loss 30%`)。
- 發送了 12 個包，但是丟了 6 個。

- 看来, 应该就是这里, 导致 Nginx 回复的响应包, 被 netem 模块给丢了。
- 直接删掉 netem 模块。
 - `tc qdisc del dev eth0 root netem loss 30%`

```
[root@demo-2:~# hping3 -c 10 -S -p 80 172.16.99.129
HPING 172.16.99.129 (ens33 172.16.99.129): S set, 40 headers + 0 data bytes
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=1 win=65535 rtt=6.8 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=2 win=65535 rtt=6.1 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=3 win=65535 rtt=5.4 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=4 win=65535 rtt=4.4 ms
DUP! len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=3 win=65535 rtt=1037.1 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=5 win=65535 rtt=7.9 ms
len=46 ip=172.16.99.129 ttl=63 DF id=0 sport=80 flags=SA seq=7 win=65535 rtt=2.8 ms

--- 172.16.99.129 hping statistic ---
10 packets transmitted, 7 packets received, 30% packet loss
round-trip min/avg/max = 2.8/152.9/1037.1 ms
```

hping3 -c 10 -S -p 80 172.16.99.129

網路層和傳輸層

- 在网络层和传输层中，引发丢包的因素非常多。不过，其实想确认是否丢包，是非常简单的事，因为 Linux 已经为我们提供了各个协议的收发汇总情况。
- `netstat -s`

```
root@nginx:/# netstat -s
Ip:
  Forwarding: 1
  36 total packets received
  0 forwarded
  0 incoming packets discarded
  26 incoming packets delivered
  20 requests sent out
Icmp:
  0 ICMP messages received
  0 input ICMP message failed
  ICMP input histogram:
  0 ICMP messages sent
  0 ICMP messages failed
  ICMP output histogram:
Tcp:
  0 active connection openings
  0 passive connection openings
  10 failed connection attempts
  0 connection resets received
  0 connections established
  26 segments received
  28 segments sent out
  10 segments retransmitted
  0 bad segments received
  0 resets sent
Udp:
  0 packets received
  0 packets to unknown port received
  0 packet receive errors
  0 packets sent
  0 receive buffer errors
  0 send buffer errors
UdpLite:
TcpExt:
  10 resets received for embryonic SYN_RECV sockets
  0 packet headers predicted
  TCPTimeouts: 14
  TCPSynRetrans: 10
IpExt:
  InOctets: 1440
  OutOctets: 880
  InNoECTPkts: 36
```

- netstat 汇总了 IP、ICMP、TCP、UDP 等各种协议的收发统计信息。不过，我们的目的是排查丢包问题，所以这里主要观察的是错误数、丢包数以及重传数。

10 次连接失败重试 (10 failed connection attempts)

10 次重传 (10 segments retransmitted)

10 次半连接重置 (10 resets received for embryonic SYN_RECV sockets)

14 次超时 (TCPTimeouts)

10 次 SYN 重传 (TCPSynRetrans)

小結

- 网络丢包，通常会带来严重的性能下降，特别是对 TCP 来说，丢包通常意味着网络拥塞和重传，进一步还会导致网络延迟增大、吞吐降低。
- 我们学会了如何从链路层、网络层和传输层等入手，分析网络丢包的问题。不过，案例最后，我们还没有找出最终的性能瓶颈，下一节，**強哥**将继续为你讲解。

思考

今天我们只分析了链路层、网络层以及传输层等。而根据 TCP/IP 协议栈和 Linux 网络收发原理, 还有很多我们没分析到的地方。那么, 接下来, 我们又该如何分析, 才能破获这个案例, 找出“真凶”呢?