任喜军的博客

java tcp mysql performance network docker Linux

通过案例来理解MSS、MTU等相关TCP概念

2017-07-18 | tcp | 34 次

通过案例来理解MSS、MTU等相关TCP概念

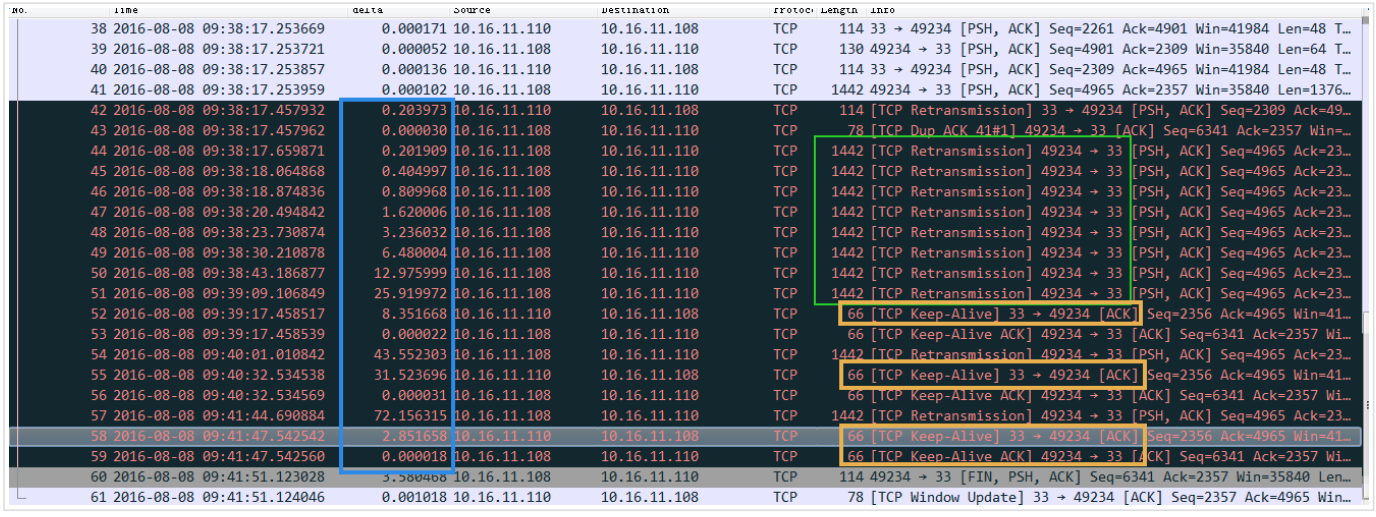
问题的描述

- 最近要通过Docker的方式把产品部署到客户机房， 过程中需要部署一个hbase集群，hbase总是部署失败（在我们自己的环境没有问题）
- 发现hbase卡在同步文件，人工登上hbase 所在的容器中看到在hbase节点之间scp同步一些文件的时候，同样总是失败（稳定重现）
- 手工尝试scp那些文件，发现总是在传送某个文件的时候scp卡死了
- 尝试单独scp这个文件依然卡死
- 在这个容器上scp其它文件没问题
- 换一个容器scp这个文件没问题

分析过程

> 实在很难理解为什么单单这个文件在这个容器上scp就卡死了，既然scp网络传输卡死，那么就同时在两个容器上tcpdump抓包，想看看为什么传不动了

在客户端抓包如下：（33端口是服务端的sshd端口，10.16.11.108是客户端ip）



No.	Time	delta	Source	Destination	Protoc	Length	Info
38	2016-08-08 09:38:17.253669	0.000171	10.16.11.110	10.16.11.108	TCP	114	33 → 49234 [PSH, ACK] Seq=2261 Ack=4901 Win=41984 Len=48 T…
39	2016-08-08 09:38:17.253721	0.000052	10.16.11.108	10.16.11.110	TCP	130	49234 → 33 [PSH, ACK] Seq=4901 Ack=2309 Win=35840 Len=64 T…
40	2016-08-08 09:38:17.253857	0.000136	10.16.11.110	10.16.11.108	TCP	114	33 → 49234 [PSH, ACK] Seq=2309 Ack=4965 Win=41984 Len=48 T…
41	2016-08-08 09:38:17.253959	0.000102	10.16.11.108	10.16.11.110	TCP	1442	49234 → 33 [PSH, ACK] Seq=4965 Ack=2357 Win=35840 Len=1376…
42	2016-08-08 09:38:17.457932	0.203973	10.16.11.110	10.16.11.108	TCP	114	[TCP Retransmission] 33 → 49234 [PSH, ACK] Seq=2309 Ack=49…
43	2016-08-08 09:38:17.457962	0.000030	10.16.11.108	10.16.11.110	TCP	78	[TCP Dup ACK 41#1] 49234 → 33 [ACK] Seq=6341 Ack=2357 Win=…
44	2016-08-08 09:38:17.659871	0.201909	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
45	2016-08-08 09:38:18.064868	0.404997	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
46	2016-08-08 09:38:18.874836	0.809968	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
47	2016-08-08 09:38:20.494842	1.620006	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
48	2016-08-08 09:38:23.730874	3.236032	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
49	2016-08-08 09:38:30.210878	6.480004	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
50	2016-08-08 09:38:43.186877	12.975999	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
51	2016-08-08 09:39:09.106849	25.919972	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
52	2016-08-08 09:39:17.458517	8.351668	10.16.11.110	10.16.11.108	TCP	66	[TCP Keep-Alive] 33 → 49234 [ACK] Seq=2356 Ack=4965 Win=41…
53	2016-08-08 09:39:17.458539	0.000022	10.16.11.108	10.16.11.110	TCP	66	[TCP Keep-Alive ACK] 49234 → 33 [ACK] Seq=6341 Ack=2357 Wi…
54	2016-08-08 09:40:01.010842	43.552303	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
55	2016-08-08 09:40:32.534538	31.523696	10.16.11.110	10.16.11.108	TCP	66	[TCP Keep-Alive] 33 → 49234 [ACK] Seq=2356 Ack=4965 Win=41…
56	2016-08-08 09:40:32.534569	0.000031	10.16.11.108	10.16.11.110	TCP	66	[TCP Keep-Alive ACK] 49234 → 33 [ACK] Seq=6341 Ack=2357 Wi…
57	2016-08-08 09:41:44.690884	72.156315	10.16.11.108	10.16.11.110	TCP	1442	[TCP Retransmission] 49234 → 33 [PSH, ACK] Seq=4965 Ack=23…
58	2016-08-08 09:41:47.542542	2.851658	10.16.11.110	10.16.11.108	TCP	66	[TCP Keep-Alive] 33 → 49234 [ACK] Seq=2356 Ack=4965 Win=41…
59	2016-08-08 09:41:47.542560	0.000018	10.16.11.108	10.16.11.110	TCP	66	[TCP Keep-Alive ACK] 49234 → 33 [ACK] Seq=6341 Ack=2357 Wi…
60	2016-08-08 09:41:51.123028	3.580468	10.16.11.108	10.16.11.110	TCP	114	49234 → 33 [FIN, PSH, ACK] Seq=6341 Ack=2357 Win=35840 Len…
61	2016-08-08 09:41:51.124046	0.001018	10.16.11.110	10.16.11.108	TCP	78	[TCP Window Update] 33 → 49234 [ACK] Seq=2357 Ack=4965 Win…

从抓包中可以得到这样一些结论：

- 从抓包中可以明显知道scp之所以卡死是因为丢包了，客户端一直在重传，图中绿框
- 图中篮框显示时间间隔，时间都是花在在丢包重传等待的过程
- 奇怪的问题是图中橙色框中看到的，网络这时候是联通的，客户端跟服务端在这个会话中依然有些包能顺利到达（Keep-Alive包）
- 同时注意到重传的包长是1442，包比较大了，看了一下tcp建立连接的时候MSS是1500，应该没有问题
- 查看了scp的两个容器的网卡mtu都是1500，正常

```
1  基本上看到这里，能想到是因为丢包导致的scp卡死，因为两个容器mtu都正常，包也小于mss，那只能是网络路由上某个环节mtu太小导致这个1
```

接下来分析网络传输链路

scp传输的时候实际路由大概是这样的

```
1  容器A---> 宿主机1 ---> ……中间的路由设备 …… ---> 宿主机2 ---> 容器B
```

- 前面提过其它容器scp同一个文件到容器B没问题，所以我认为中间的路由设备没问题，问题出在两台宿主机上
- 在宿主机1上抓包发现抓不到丢失的那个长度为 1442 的包，也就是问题出在了 容器A—> 宿主机1 上

查看宿主机1的dmesg看到了这样一些信息

```
2016-08-08T08:15:27.125951+00:00 server kernel: openvswitch: ens2f0.627: dropped over-mtu packet: 1428 > 1400
2016-08-08T08:15:27.536517+00:00 server kernel: openvswitch: ens2f0.627: dropped over-mtu packet: 1428 > 1400
```

结论

- 到这里问题已经很明确了 openvswitch 收到了 一个1428大小的包因为比mtu1400要大，所以扔掉了，接着查看宿主机1的网卡mtu设置果然是1400，悲催，马上修改mtu到1500，问题解决。

最后的总结

- 因为这是客户给的同一批宿主机默认想当然的认为他们的配置到一样，尤其是mtu这种值，只要不是故意捣乱就不应该乱修改才对，我只检查了两个容器的mtu，没看宿主机的mtu，导致诊断中走了一些弯路
- 通过这个案例对mtu/mss等有了进一步的了解
- 从这个案例也理解了vlan模式下容器、宿主机、交换机之间的网络传输链路
- 其实抓包还发现了比1500大得多的包顺利通过，反而更小的包无法通过，这是因为网卡基本都有拆包的功能了

常见问题

Q: 传输的包超过MTU后表现出来的症状？

A：卡死，比如scp的时候不动了，或者其他更复杂操作的时候不动了，卡死的状态。

Q： 为什么我的MTU是1500，但是抓包看到有个包2700，没有卡死？

A： 有些网卡有拆包的能力，具体可以Google：LSO、TSO，这样可以减轻CPU拆包的压力，节省CPU资源。

tcp # MTU # MSS

‹ 10+倍性能提升全过程