

原 线上问题分析：The target server failed to respond (目标服务器返回失败)

2015-09-07 17:57 5310人阅读 评论(0) 收藏 举报

目录(?)

[-]

1. 背景
2. 问题分析
3. 结论
4. 总结

背景

后台服务通过REST接口调用第三方服务，调用偶尔会发生失败。查看后台日志发现，抛出The target server failed to respond。

该问题只是在特定操作下才会出现，暂时没有任何规律可言。

问题分析

- 1、到第三方服务器上查看后台日志，发现在相同时间点没有日志输出。查看tomcat的localhost_access访问日志，也没有发现任何日志信息。初步怀疑请求没有达到第三方服务器。
- 2、怀疑是gateway的问题，但是调用服务是通过ip直连的方式，没有通过gateway中转。排除此种可能。
- 3、为了验证http请求是否发出，以及是否到达了第三方服务器，使用tcpdump工具，对网络包进行了监控。只监控这两台服务器，脚本为：

```
01. tcpdump host A and B -w test.cap
```

注意，此处生成的是cap文件，方便在windows上用wireshark工具进行分析。

用wireshark分析的结果如下：

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000			HTTP	324	GET /...%7B%22priceType%22%3A%220%22%2C%22skuSn%22%3A%22...
2	0.000189			TCP	54	47926->10072 [FIN, ACK] Seq=271 Ack=1 Win=123 Len=0
3	0.000484			TCP	54	10072->47926 [RST] Seq=1 Win=0 Len=0

从分析结果可以看出，已经与第三方服务器建立了连接，但是第三方服务器没有返回HTTP应答，只有一个TCP响应报文。由此可以排除步骤一的疑点。

但是按顺序分析这几个报文，一开始就建立一个HTTP连接，不太符合TCP连接的规范。理论上，应该是走TCP的三次握手。有一种可能是，使用了连接池的机制，重用了TCP的连接。但是为什么第三方服务器会发送一个重置的TCP报文呢？

- 4、在我们的服务器上查看连接情况，使用netstat -nltpa，发现有个CLOSE_WAIT状态的连接：

```
tcp        1      0  ::ffff:10.77.144.186:52040  ::ffff:10.77.145.107:10072  CLOSE_WAIT  20127/java
```

根据TCP的四次挥手理论，此种情况是因为第三方服务器端主动发起了关闭请求，而咱们的服务器还没有对这个请求做处理。出现CLOSE_WAIT状态连接的原因：

咱们的HTTP协议使用的是1.1版本，因此会默认为Keep-Alive，在请求头中，Connection确实设置为了Keep-Alive。排除客户端主动关闭的可能。

```
⊟ Hypertext Transfer Protocol
  + GET /pricecenter/v1/price/mcondit
    Host: 10.77.145.107:10072\r\n
    Connection: Keep-Alive\r\n
```

客户端出现CLOSE_WAIT状态的TCP连接

查看第三方服务器的tomcat配置,发现超时时间是20s。因为在请求处理完后的20s,服务器端就会主动关闭TCP连接。

有了这个线索,我们就掌握了重现的规律:发送一个请求后20s,待客户端连接状态变为CLOSE_WAIT状态,再发送一次请求就出现错误了。

结论

客户端肯定会出现CLOSE_WAIT状态的TCP连接

1、客户端使用了HttpClient的连接池机制,因此TCP连接会被重复使用,并且由于默认使用的是HTTP 1.1协议,Keep-Alive会被置上,因此不会主动关闭连接。但是由于第三方服务器配置了20s的超时时间,TCP连接会被关闭,但是此时客户端的TCP连接会被回收到池子里,不会理睬服务器发送过来的关闭请求,因此客户端的连接状态会变成CLOSE_WAIT,而服务器经过一段时间等待后会关闭自身的TCP连接。如果客户端下一次请求正好使用了这个TCP连接,就会出现服务器返回空应答的情况,从而抛出该问题。

2、HTTP协议是基于TCP连接的,如果是重用已有的TCP连接,则TCP三次握手不会发生。抛出该问题,是HttpClient在解析HTTP头时,发现没有数据。根本原因是因为没有返回HTTP数据包,而是返回了TCP数据包。

问题根源:

总结

1、可以使用HttpClient重试机制,调用第一次失败时,第二次可能就正常了。HTTP 1.1默认使用了持久连接机制,像这种情况不可避免。如果访问频繁的话,该问题不会发生,而且性能会比较高。

2、开启一个线程,定期关闭无效的TCP连接。

HttpClientConnectionManager.closeExpiredConnections(),

HttpClientConnectionManager.closeIdleConnections()。这个方法没试过。(此方法为官方文档的推荐方法)

3、在请求头里设置"Connection":"close"。弊端是,没有用上连接池的性能。

4、设置HttpClient的连接重用策略。目前的重用策略,对于HTTP 1.1,都是默认为可以重用。

服务端

如果服务器使用长连接配置,那不做调整应该也不会碰到该问题。

但是服务器如果设置了keep-alive时间,客户端如果也同样设置,感觉不太灵活。

有没有更好的方法,还需要去研究下HttpClient的实现。