tcp短连接TIME_WAIT问题解决方法大全（1）——高屋建瓴

2012年11月04日 21:45:54 阅读数：14820

tcp连接是网络编程中最基础的概念，基于不同的使用场景，我们一般区分为“长连接”和“短连接”，
长短连接的优点和缺点这里就不详细展开了，有心的同学直接去google查询，本文主要关注如何解决tcp短连接的TIME_WAIT问题。

短连接最大的优点是方便，特别是脚本语言，由于执行完毕后脚本语言的进程就结束了，基本上都是用短连接。
但短连接最大的缺点是将占用大量的系统资源，例如：本地端口、socket句柄。
导致这个问题的原因其实很简单：tcp协议层并没有长短连接的概念，因此不管长连接还是短连接，连接建立->数据传输->连接关闭的流程和处理都是一样的。

正常的TCP客户端连接在关闭后，会进入一个TIME_WAIT的状态，持续的时间一般在1~4分钟，对于连接数不高的场景，1~4分钟其实并不长，对系统也不会有什么影响，
但如果短时间内（例如1s内）进行大量的短连接，则可能出现这样一种情况：客户端所在的操作系统的socket端口和句柄被用尽，系统无法再发起新的连接！

举例来说：假设每秒建立了1000个短连接（Web场景下是很常见的，例如每个请求都去访问memcached），假设TIME_WAIT的时间是1分钟，则1分钟内需要建立6W个短连接，
由于TIME_WAIT时间是1分钟，这些短连接1分钟内都处于TIME_WAIT状态，都不会释放，而Linux默认的本地端口范围配置是：net.ipv4.ip_local_port_range = 32768 61000
不到3W，因此这种情况下新的请求由于没有本地端口就不能建立了。

可以通过如下方式来解决这个问题：
1）可以改为长连接，但代价较大，长连接太多会导致服务器性能问题，而且PHP等脚本语言，需要通过proxy之类的软件才能实现长连接；
2）修改ipv4.ip_local_port_range，增大可用端口范围，但只能缓解问题，不能根本解决问题；
3）客户端程序中设置socket的SO_LINGER选项；
4）客户端机器打开tcp_tw_recycle和tcp_timestamps选项；
5）客户端机器打开tcp_tw_reuse和tcp_timestamps选项；
6）客户端机器设置tcp_max_tw_buckets为一个很小的值；

在解决php连接Memcached的短连接问题过程中，我们主要验证了3）4）5）6）几种方法，采取的是基本功能验证和代码验证，并没有进行性能压力测试验证，
因此实际应用的时候需要注意观察业务运行情况，发现丢包、断连、无法连接等现象时，需要关注是否是因为这些选项导致的。

虽然这几种方法都可以通过google查询到相关信息，但这些信息大部分都是泛泛而谈，而且绝大部分都是人云亦云，没有很大参考价值。
我们在定位和处理这些问题过程中，遇到一些疑惑和困难，也花费了一些时间去定位和解决，以下就是相关的经验总结。