**大话keepalive**

<http://www.cnblogs.com/yjf512/p/5354055.html>

我们说到keepalive的时候，需要先明确一点，这个keepalive说的是tcp的还是http的。

tcp的keepalive是侧重在保持客户端和服务端的连接，一方会不定期发送心跳包给另一方，当一方端掉的时候，没有断掉的定时发送几次心跳包，如果间隔发送几次，对方都返回的是RST，而不是ACK，那么就释放当前链接。设想一下，如果tcp层没有keepalive的机制，一旦一方断开连接却没有发送FIN给另外一方的话，那么另外一方会一直以为这个连接还是存活的，几天，几月。那么这对服务器资源的影响是很大的。

http的keep-alive一般我们都会带上中间的横杠，普通的http连接是客户端连接上服务端，然后结束请求后，由客户端或者服务端进行http连接的关闭。下次再发送请求的时候，客户端再发起一个连接，传送数据，关闭连接。这么个流程反复。但是一旦客户端发送connection:keep-alive头给服务端，且服务端也接受这个keep-alive的话，两边对上暗号，这个连接就可以复用了，一个http处理完之后，另外一个http数据直接从这个连接走了。

**tcp层的keepalive**

tcp的keepalive就是为了检测链接的可用性。主要调节的参数有三个：

* tcp\_keepalive\_time // 距离上次传送数据多少时间未收到判断为开始检测
* tcp\_keepalive\_intvl // 检测开始每多少时间发送心跳包
* tcp\_keepalive\_probes // 发送几次心跳包对方未响应则close连接

基本上的流程：

在客户端和服务端进行完三次握手之后，客户端和服务端都处在ESTABLISH状态，这个时候进行正常的PSH和ACK交互，但是一旦一方服务中断了，另一方在距离上次PSH时间tcp\_keepalive\_time发现对方未发送数据，则开始心跳检测。心跳检测实际就是发送一个PSH的空心跳包，这里说的空心跳包就是包的数据为空，但是TCP包的头部的数据和标识和正常包一样。如果这个包获取到的是RST返回的话，下面就会继续每隔tcp\_keepalive\_intval的时长发送一个空心跳包，如果tcp\_keepalive\_probes次心跳包对方都是返回RST而不是ACK，则心跳发起方就判断这个连接已经失效，主动CLOST这个连接。

这三个参数可以每个TCP连接都不同，使用tcp设置变量的函数可以设置当前tcp连接的这三个对应的值。

`(}%GU1FEMZMGPR}[67[}W5

tcp层的keepalive会在两个场景下比较有用：

* 检测连接的一方是否断了

这里说的连接的一方是否断了包含几种情况：

* 连接一方服务中止
* 网络不好导致的服务长时间无响应
* 连接一方服务重启中

结合这三种方式就很好理解为什么会有 tcp\_keepalive\_time, tcp\_keepalive\_intval, tcp\_keepalive\_probes三种的设置了。如果是对方服务器进行重启的时候，我们不能根据一次的tcp返回重置信号就判定这个连接失效。相反的，重启之后，这个心跳包一旦正常，这个连接仍然可以继续使用。

* 防止因为长时间不用链接导致连接失效

这个往往在代理或者内网状况下会使用到。一般NAT网络为了资源，会和外网保持一定的资源连接数，而且采用的是淘汰机制，淘汰掉旧的，不用的连接，创建和使用新的连接。如果我们没有心跳检测机制，那么我们的连接在一段时间没有使用的时候，NAT对外的机制会判断对应的对外网络是无用的，淘汰掉旧的，即使这个时候客户端和服务端都还正常服务着，只是长时间未联络了而已。keepalive的机制由于有定时心跳包，自然就能解决这个问题了。

**http层的keep-alive**

http层有个keep-alive, 它主要是用于客户端告诉服务端，这个连接我还会继续使用，在使用完之后不要关闭。

这个设置会影响web服务的哪几个方面呢？

* 性能

这个设置首先会在性能上对客户端和服务器端性能上有一定的提升。很好理解的是少了TCP的三次握手和四次挥手，第二次传递数据就可以通过前一个连接直接进行数据交互了。当然会提升服务性能了。

* 服务器TIME\_WAIT的时间

由于HTTP服务的发起方一般都是浏览器，即客户端。但是先执行完逻辑，传输完数据的一定是服务端。那么一旦没有keep-alive机制，服务端在传送完数据之后会率先发起连接断开的操作。由于TCP的四次挥手机制，先发起连接断开的一方会在连接断开之后进入到TIME\_WAIT的状态达到2MSL之久。设想，如果没有开启HTTP的keep-alive，那么这个TIME\_WAIT就会留在服务端，由于服务端资源是非常有限的，我们当然倾向于服务端不会同一时间hold住过多的连接，这种TIME\_WAIT的状态应该尽量在客户端保持。那么这个http的keep-alive机制就起到非常重要的作用了。

所以

基本上基于这两个原因，现在的浏览器发起web请求的时候，都会带上connection:keep-alive的头了。

**总结**

TCP的keepalive机制和HTTP的keep-alive机制是说的完全不同的两个东西，tcp的keepalive是在ESTABLISH状态的时候，双方如何检测连接的可用行。而http的keep-alive说的是如何避免进行重复的TCP三次握手和四次挥手的环节。