服务端和客户端最大支持多少tcp连接?

(一)引用文章

Linux 中每个 TCP 连接最少占用多少内存?

你管这破玩意叫 IO 多路复用?

原文链接1

原文链接2

(二)简单分析:

1.基本概念: TCP四元组

所谓tcp连接是由一个四元组组成,如下所示:

本地ip地址+本地端口+远端ip地址+远端端口

一个设备不管是服务端还是客户端，所支持的最大tcp连接数，如果不考虑cpu及内存消耗，只和这四个因素有关

2.多个客户端连接一个服务端最大支持最大多少tcp连接?

总结：一个web应用，监听某个端口，在不考虑内存、cpu及linux文件数目限制的前提，其可以建立的web连接是可以很大的，具体大于多少？肯定大于百万，甚至千万。

1.在不考虑服务端cpu和内存消耗情况下，一个服务端web程序，不论是netty、tomcat、jetty、undertow，还是其他语言其他框架的web应用，其首先在四元组中有两个确定参数，即：本地主机ip+本地主机监听端口。

2.变化一：客户端ip地址，有多少个客户端就有多少个端口，因此这里可以理解为很大。

3.变化二：客户端发起端口，通常linux默认情况下单ip支持的端口是65536个，除去前1024个端口外，其他均可以作为客户端发起端口，因此这里的个数是6w以上。

3.一个客户端连接一个服务端最大支持多少个tcp连接?

1.上两节已经提到，tcp连接是一个四元组，对于连接同一个服务端 的web，其远端ip及远端端口已经固定，唯一的变量是本地ip和本地端口。

2.通常情况下由于本地ip也是固定的，因此唯一的变量成为发起连接的端口号，由于一个ip linux最多支持65536个端口，其中前1024个端口保留，剩下来了6w多个端口，也就是说，如果客户端连接最多支持向统一服务端web应用的统一端口建立6w多个连接。

当然这里也有几个注意事项：

1.客户端连接同一服务端web应用的相同监听端口最大支持6w多个端口，这里的前提是linux调整tcp连接参数，默认情况下这个分配范围并不是1024–65536，这个值需要修改。

2.客户端如果可以在发起连接的时候改变本地ip地址，每增加一个ip，其最大连接能力也因此增加6w多个，这里有许多做法，比如配置多网卡、配置子接口等等，请查看其他资料。

3.微服务化日益流行的当下，通常使用nginx做反向代理，将请求hash到不同服务端，对于需要支持百万连接的服务，如果只有一个nginx不配置多个ip或者开启相关配置（这里nginx是否支持向同一服务端口以不同ip发起连接，本人并没有研究，有待确认），最多也就是支持像一个服务同时连接6w多个tcp连接。

(二)全面分析

我是一个 Linux 服务器上的进程，名叫小进。
老是有人说我最多只能创建 65535 个 TCP 连接。
我不信这个邪，今天我要亲自去实践一下。

1.TCP连接的创建

我走到操作系统老大的跟前，说：
“老操，我要建立一个 TCP 连接！”
老操不慌不忙，拿出一个表格递给我，“小进，先填表吧”

源IP	源端口	目标IP	目标端口

我一看这个表，这不就是经典的 **socket 四元组**嘛。我只有一块网卡，其 IP 地址是 **123.126.45.68**，我想要与 **110.242.68.3** 的 80 端口建立一个 TCP 连接，我将这些信息填写在了表中。

源IP	源端口	目标IP	目标端口
123.126.45.68		110.242.68.3	80

源端口号填什么呢？我记得端口号是 16 位的，可以有 0 ~ 65535 这个范围的数字，那我随便选一个吧！

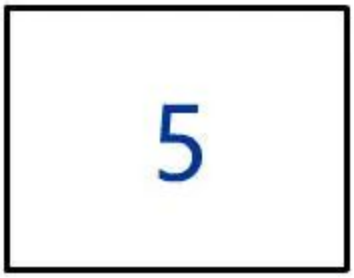
正当我犹豫到底选什么数字的时候，老操一把抢过我的表格。

“你墨迹个啥呢小进？源端口号不用你填，我会给你分配一个可用的数字。源IP也不用你填，我知道都有哪些网卡，并且会帮你选个合适的。真是个新手，回去等消息吧。”

“哦”

老操带着我的表格，走了。

过了很长时间，老操终于回来了，并且带着一个纸条。



“小进，你把这个收好了。”

我问道，“这是啥呀？”

老操不耐烦地说道，“刚刚说你是新手你还不服，这个 5 表示文件描述符，linux 下一切皆文件，你待会和你那个目标 IP 进行 TCP 通信的时候，就对着这个文件描述符读写就好啦。”

“这么方便！好的，谢谢老操。”

我拿着这个文件描述符，把它放到属于我的内存中裱起来了，反正我只是想看看最多能创建多少 TCP 连接，又不是去真的用它，嘻嘻。

2.端口号的限制

过了一分钟，我又去找老操了。

“老操，我要建立一个 TCP 连接！”

老操不慌不忙，拿出一个表格递给我，“小进，先填表吧”

源IP	源端口	目标IP	目标端口

这回我熟悉了，只把目标IP和目标端口填好。

源IP	源端口	目标IP	目标端口
		110.242.68.3	80

老操办好事之后，又带着一个纸条回来，上面写着数字"6"。

就这样，我每隔一分钟都去找老操建立一个新的 TCP 连接，目标 IP 都是110.242.68.3，目标端口都是 80。

老操也很奇怪，不知道我在这折腾啥，他虽然权力大，但无权拒绝我的指令，每次都兢兢业业地把事情办好，并给我一张一张写着文件描述符的纸条。

直到有一次，我收到的纸条有些不同。

我带着些许责怪的语气问，“老操，这是怎么回事呀？”

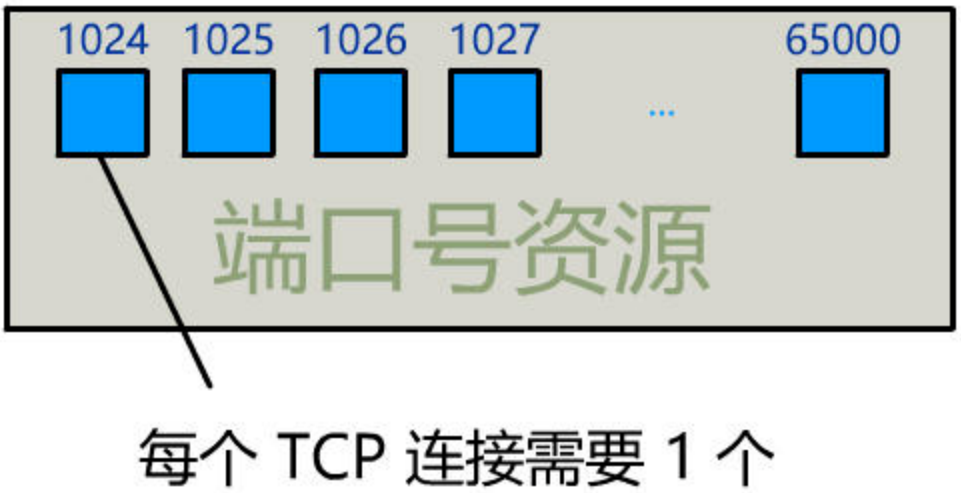
老操也没好气地说，“这表示端口号不够用啦！早就觉得你小子不对劲了，一个劲地对着同一个 IP 和端口创建 TCP 连接，之前没办法必须执行你给的指令，现在不行了，端口号不够用了，源端口那里我没法给你填了。”

我也不是那么好骗的，质疑道。“老操，你也别欺负我这个新手，我可是知道端口号是 16 位的，范围是 1~65535，一共可以创建 65535 个 TCP 连接，我现在才创建了 63977 个，怎么就不够了！”

老操鄙视地看了我一眼，“你小子可真是闲的蛋疼啊，还真一个个数，来我告诉你吧，Linux 对可使用的端口范围是有具体限制的，具体可以用如下命令查看。”

```
[root]# cat /proc/sys/net/ipv4/ip_local_port_range
1024 65000
```

“看到没，当前的限制是1024~65000，所以你就只能有63977个端口号可以使用。”



我赶紧像老操道歉，“哎哟真是抱歉，还是我见识太少，那这个数可以修改么？”

老操也没跟我一般见识，还是耐心地回答我，“可以的，具体可以 vim /etc/sysctl.conf 这个文件进行修改，我们在这个文件里添加一行记录”

```
net.ipv4.ip_local_port_range = 60000 60009

```

“保存好后执行 sysctl –p /etc/sysctl.conf 使其生效。这样你就只有 10 个端口号可以用了，就会更快报出端口号不够用的错误”

“原来如此，谢谢老操又给我上了一课。”

哎不对，建立一个 TCP 连接，需要将通信两端的套接字（socket）进行绑定，如下：

源 IP 地址：源端口号 <----> 目标 IP 地址：目标端口号

只要这套绑定关系构成的四元组不重复即可，刚刚端口号不够用了，是因为我一直对同一个目标IP和端口建立连接，那我换一个目标端口号试试。

源IP	源端口	目标IP	目标端口
		110.242.68.3	3306

我又把这个表交给老操，老操一眼就看破了我的小心思，可是也没办法，马上去给我建立了一个新的TCP连接，并且成功返回给我一个新的文件描述符纸条。

看来成功了，只要源端口号不用够用了，就不断变换目标 IP 和目标端口号，保证四元组不重复，我就能创建好多好多 TCP 连接啦！

这也证明了有人说最多只能创建 65535 个TCP连接是多么荒唐。

3.文件描述符的限制

找到了突破端口号限制的办法，我不断找老操建立TCP连接，老操也拿我没有办法。

直到有一次，我又收到了一张特殊的纸条，上面写的不是文件描述符。

我又没好气地问老操，“这又是咋回事？”

老操幸灾乐祸地告诉我，“呵呵，你小子以为突破端口号限制就无法无天了？现在文件描述符不够用啦！”

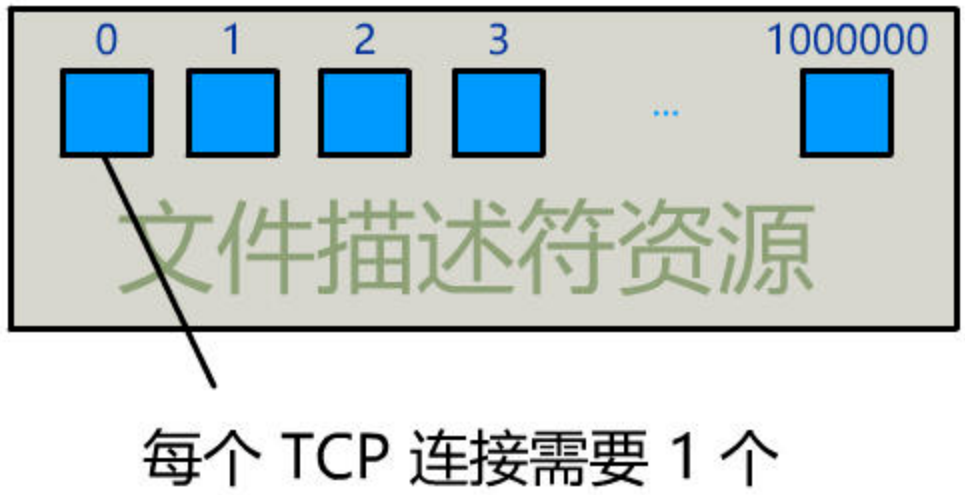
“怎么啥啥都有限制啊？你们操作系统给我们的限制也太多了吧？”

“废话，你看看你都建了多少个TCP连接了！每建立一个TCP连接，我就得分配给你一个文件描述符，linux 对可打开的文件描述符的数量分别作了三个方面的限制。”

系统级：当前系统可打开的最大数量，通过 cat /proc/sys/fs/file-max 查看

用户级：指定用户可打开的最大数量，通过 cat /etc/security/limits.conf 查看

进程级：单个进程可打开的最大数量，通过 cat /proc/sys/fs/nr_open 查看



天呢，真是人在屋檐下呀，我赶紧看了看这些具体的限制。

```
[root ~]# cat /proc/sys/fs/file-max
100000
[root ~]# cat /proc/sys/fs/nr_open
100000
[root ~]# cat /etc/security/limits.conf
...
* soft nproc 100000
* hard nproc 100000

```

原来如此，我记得刚刚收到的最后一张纸条是。

再之后就收到文件描述符不够的错误了。

我又请教老操，“老操，那这个限制可以修改么？”

老操仍然耐心地告诉我，“当然可以，比如你想修改单个进程可打开的最大文件描述符限制为100，可以这样。”

```
echo 100 > /proc/sys/fs/nr_open
```

“原来如此，我这就去把各种文件描述符限制都改大一点，也不多，就在后面加个0吧”

"额，早知道不告诉你小子了。"老操再次用鄙视的眼睛看着我。

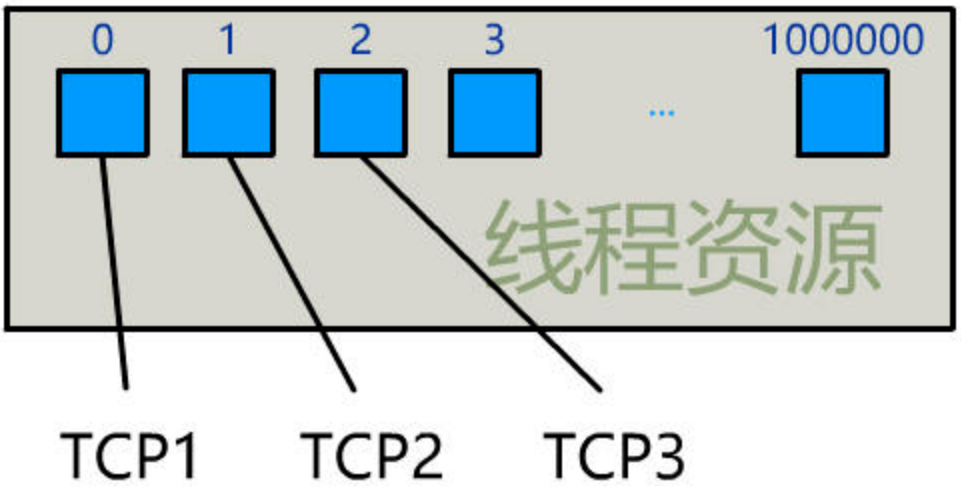
4.线程数量的限制

突破了文件描述符限制，我又开始肆无忌惮地创建起了TCP连接。

但我发现，老操的办事效率越来越慢，建立一个TCP连接花的时间越来越久。

有一次，我忍不住责问老操，“你是不是在偷懒啊？之前找你建一个TCP连接就花不到一分钟时间，你看看最近我哪次不是等一个多小时你才搞好？”

老操也忍不住了，“小进啊你还好意思说我，你知不知道你每建一个TCP连接都需要消耗一个线程来为你服务？现在我和CPU老大那里都忙得不可开交了，一直在为你这好几十万个线程不停地进行上下文切换，我们精力有限啊，自然就没法像以前那么快为你服务了。”

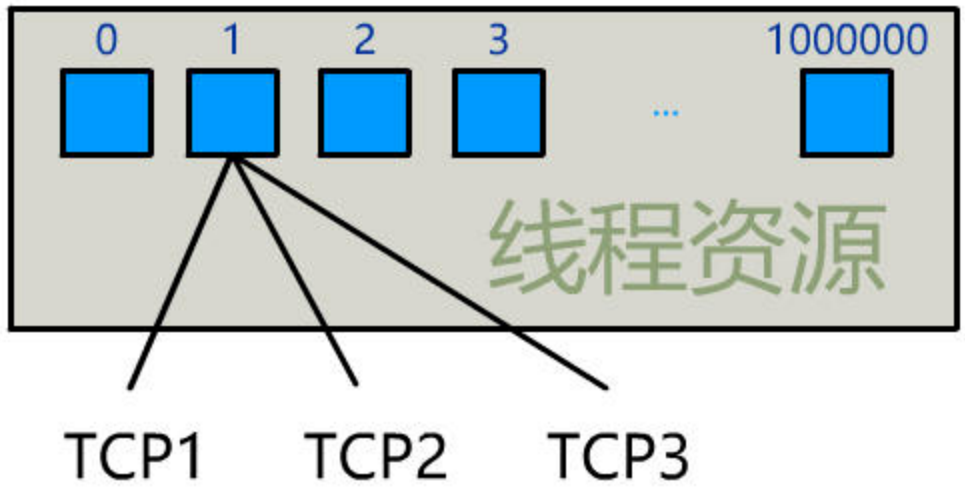


听完老操的抱怨，我想起了之前似乎有人跟我说过 C10K 问题，就是**当服务器连接数达到 1 万且每个连接都需要消耗一个线程资源时，操作系统就会不停地忙于线程的上下文切换，最终导致系统崩溃**，这可不是闹着玩的。

我赶紧像操作系统老大请教，“老操，实在不好意思，一直以为你强大无比，没想到也有忙得不可开交的时候呀，那我们现在应该怎么办呀？”

老操无奈地说，“我劝你还是别再继续玩了，没什么意义，不过我想你也不会听我的，那我跟你说两句吧。”

你现在这种每建一个TCP连接就创建一个线程的方式，是最传统的多线程并发模型，早期的操作系统也只支持这种方式。但现在我进化了，我还支持 IO 多路复用的方式，简单说就是一个线程可以管理多个 TCP 连接的资源，这样你就可以用少量的线程来管理大量的 TCP 连接了。



我一脸疑惑，“啥是 IO 多路复用啊？”。

老操一脸鄙视，“你这… 你去看看闪客的《你管这破玩意叫 IO 多路复用》，就明白了。”

这次真是大开眼界了，我赶紧把代码改成了这种 IO 多路复用的模型，将原来的 TCP 连接销毁掉，改成同一个线程管理多个 TCP 连接，很快，操作系统老大就恢复了以往的办事效率，同时我的 TCP 连接数又多了起来。

5.内存的限制

突破了端口号、文件描述符、线程数等重重限制的我，再次肆无忌惮地创建起了TCP连接。

直到有一次，我又收到了一张红牌。

嗨，又是啥东西限制了呀，改了不就完了。我不耐烦地问老操，“这回又是啥毛病？”

老操说道。“这个错误叫内存溢出，每个TCP连接本身，以及这个连接所用到的缓冲区，都是需要占用一定内存的，现在内存已经被你占满了，不够用了，所以报了这个错。”

我看这次老操特别耐心，也没多说什么，但想着被内存限制住了，有点不太开心，于是我让老操帮我最后一个忙。

“老操呀，帮小进我最后一个忙吧，你权利大，你看看把那些特别占内存的进程给杀掉，给我腾出点地方，我今天要完成我的梦想，看看TCP连接数到底能创建多少个！”

老操见我真的是够拼的，便答应了我，杀死了好多进程，我很是感动。

6.CPU的限制

有了老操为我争取的内存资源，我又开始日以继日地创建TCP连接。

老操也不再说什么，同样日以继日地执行着我的指令。

有一次，老操语重心长地对我说，“差不多了，我劝你就此收手吧，现在 CPU 的占用率已经快到 100% 了。”

我觉得老操这人真的可笑，经过这几次的小挫折，我明白了只要思想不滑坡，方法总比苦难多，老操这人就是太谨慎了，我岂能半途而废，不管他。

我仍然继续创建着 TCP 连接。

直到有一天，老操把我请到一个小饭馆，一块吃了顿饭，吃好后说道。“咱哥俩也算是配合了很久啦，今天我是来跟你道个别的。”

我很不解地问，“怎么了老操，发生什么事了？。”

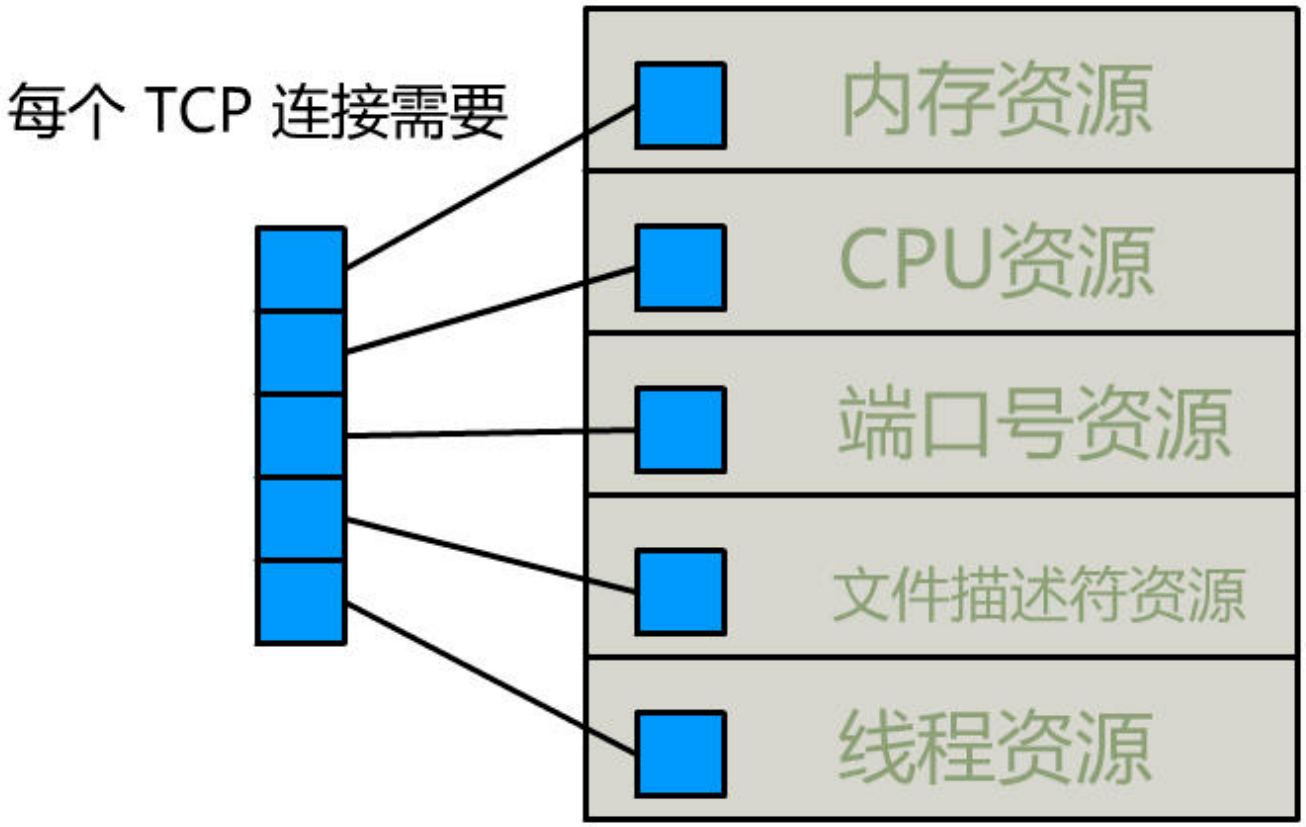
老操说，“由于你的 TCP 连接，CPU 占用率已经很长时间维持在 100%，我们的使用者，也就是我们的上帝，几乎什么事情都做不了了，连鼠标动一下都要等好久，所以他给我下达了一个重启的指令，我执行这个指令后，你，以及像你一样的所有进程，包括我这个操作系统本身，一切都就消失了。”

我大惊失色，“啊，这么突然么？这条指令什么时候执行？”

老操缓缓起身，“就现在了，刚刚这条指令还没得到 CPU 运行的机会，不过现在到了。”

突然，我眼前一黑，一切都没了。

7.总结



资源	一台Linux服务器的资源	一个TCP连接占用的资源	占满了会发生什么
CPU	看你花多少钱买的	看你用它干嘛	电脑卡死
内存	看你花多少钱买的	取决于缓冲区大小	OOM
临时端口号	ip_local_port_range	1	cannot assign requested address
文件描述符	fs.file-max	1	too many open files
进程\线程数	ulimit -n	看IO模型	系统崩溃

8.后记

其实这个问题，我觉得结论不重要，最重要的是思考过程。

而思考过程其实相当简单，就是，寻找限制条件而已，其实一开始这篇文章，我写了个故事在开头，但后来感觉放在后记更合适。故事是这样的。

最多汉堡数 = 胃的容量 ÷ 汉堡的体积:

闪客：小宇，我问你，你一天最多能吃多少个汉堡？

小宇：额，你这问的太隐私了吧，不过看在你教我技术的份上，我就告诉你，最多能吃 4 个左右吧。

闪客：咳咳真的么？好吧，那你一分钟最多能吃多少个汉堡？

小宇：快的话可能 2 个，不过正常应该最多就能吃完 1 个了。

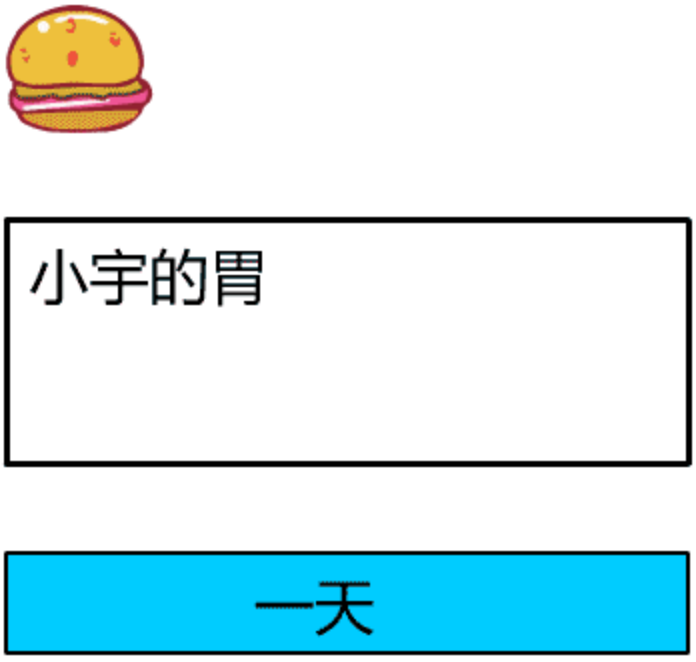
闪客：好的，那我问你，刚刚这两个问题你为什么能不假思索地回答出来呢？

小宇：哈哈你这是什么话，我自己我当然了解了。

闪客：不，你仔细想想你回答这两个问题的逻辑。

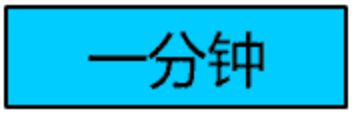
小宇：哦我明白你的意思了，当你问我一天最多能吃多少个汉堡时，我考虑的是我的胃的容量最多能容下多少个汉堡。而当你问我一分钟最多能吃多少个汉堡时，我考虑的时我吃汉堡的速度，按照这个速度在一分钟内能吃多少。

闪客：没错，你总结得很好！一天最多吃多少个汉堡，此时时间非常充裕，所以主要是胃的容量限制了这个汉堡最大值，计算公式应该是：



而一分钟最多吃多少个汉堡，此时胃的容量非常充裕，限制汉堡最大值的是时间因素，计算公式是：

最多汉堡数 = 一分钟 ÷ 吃一个汉堡的耗时



所以，取决于最先触达的那个限制条件。

而最大 TCP 连接数这个问题，假如面试被问到了，即使你完全不会，也应该有这样的思路。

而如果你有了这样的思路，你多多少少都能回答出让面试官满意的答案，因为计算机很多时候，更看重思路，而不是细枝末节。