还记得我们前面一起实现的一个简单的登录注册网站吗？我们的服务端是打包成一个war包，发布在一个tomcat下。

随着用户的增加，越来越多的用户会来访问这个网站，单个tomcat已经捉襟见肘，我们要知道一个tomcat能支撑的并发也就几百个，当有几千个用户同时来访问这一个tomcat的时候，就无法满足此激增的并发访问了。很多用户反馈商城访问较慢，甚至服务器还宕机了。

那么这个时候如何解决呢？第一步就是集群。一台机器不够用，那么两台呢？三台呢？用户的请求分散到这些机器上，不就可以显著提高性能了吗？

并且，集群下某个机器宕机，不会影响整体服务，毕竟还有其他机器在工作，等我们修复好那台宕机的机器后再加入到集群中，显著提升了可用性！可以看到，集群可以同时提高性能和可用性，众人拾柴火焰高，这就是团队的力量。

那么这里就需要一个代理服务器，把用户的请求分发到我们的tomcat集群的机器上。对外，用户只需要向这个代理服务器发起请求即可，代理服务器将请求按照算法转发给集群下的某台主机去处理，并且将处理结果返回给用户。

那么显然，这个代理服务器就需要支持很高的并发，否则难堪此大任。苦苦寻找那位济世大侠，这个时候，Nginx款款而来！

一、Nginx、Apache、Tomcat的区别

目前常见的服务器有：

- `MS IIS` ：一般是用来发布 `asp.net` 的项目，了解即可。
- `Weblogic` 、 `Jboss` ：一般用于传统行业比如ERP/物流/电信/金融，一般是收费的，了解即可。
- `Tomcat` 、 `Jetty` ：一般要遵循 `J2EE` 规范进行开发应用程序，我们的 `JAVA` 项目一般是发布在 `tomcat` 上为主。
- `Apache` 、 `Nginx` ：静态服务、反向代理。
- `Netty` ：高性能服务器编程。

接触比较多的可能是 `Tomcat` 、 `Nginx` 和 `Netty` 。当然了还有一个 `Apache` ,实际上还有一个叫做 `Apache` 软件基金会， `Apache` 是这个基金会下的一个项目， `Tomcat` 是 `Apache` 基金会下的另一个项目。 `Apache` 图标是一根羽毛，我们以后很多的故事都是跟他有关的。

`Nginx` 同 `Apache` 一样都是一种 `WEB` 服务器。 `Apache` 出现的时间太长了，它兴起的年代，互联网产业远远比不上现在。所以它被设计为一个重量级的。它是不支持高并发的服务器。在 `Apache` 上运行数以万计的并发访问，会导致服务器消耗大量内存。操作系统对其进行进程或线程间的切换也消耗了大量的CPU资源，导致 `HTTP` 请求的平均响应速度降低。

这些都决定了 `Apache` 不可能成为高性能WEB服务器，轻量级高并发服务器 `Nginx` 就应运而生了。

俄罗斯的工程师 `Igor Sysoev` ，他在为 `Rambler Media` 工作期间，使用C语言开发了 `Nginx` 。 `Nginx` 作为 `WEB` 服务器一直为 `Rambler Media` 提供出色而又稳定的服务。

`Tomcat` 是 `JAVA` 应用服务器，处理动态请求的。前面说的 `Apache` 和 `Nginx` 是用来处理静态资源比如静态页面、图片、CSS等。实际在使用时， `Tomcat` 和 `Nginx` 相辅相成，配合使用：

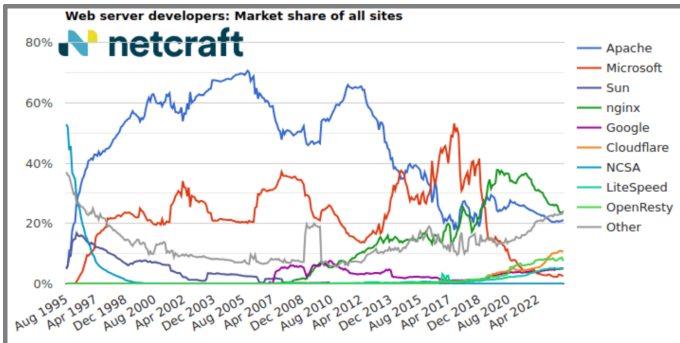
- 动静态资源分离—运用 `Nginx` 的反向代理功能分发请求：所有动态资源的请求交给 `Tomcat` ，而静态资源的请求（例如图片、视频、 `CSS` 、 `JavaScript` 文件等）则直接由 `Nginx` 返回到浏览器，这样能大大减轻 `Tomcat` 的压力。
- 负载均衡，当业务压力增大时，可能一个 `Tomcat` 的实例不足以处理，那么这时可以启动多个 `Tomcat` 实例进行水平扩展，而 `Nginx` 的负载均衡功能可以把请求通过算法分发到各个不同的实例进行处理

所以，如果你想整一个能提供接口的服务器，那么 `Tomcat` 将是你的最优选择。如果只是用于作为反向代理或者说是负载均衡的话， `Nginx` 是你的最优选择。而一个高性能的网站，往往需要两者的结合。

二、为什么选择Ngnix而不是其他代理服务器

NETCRAFT 公司官网每月公布的调研数据（Web Server Survey）已成为当今人们了解全球网站数量以及服务器市场分额情况的主要参考依据，时常被诸如华尔街杂志，英国 BBC，Slashdot 等媒体报道或引用。

根据2023年9月最新发布的报告显示（https://www.netcraft.com/blog/september-2023-web-server-survey/），其中所有网站中使用服务器类型的份额占比是：



Developer	August 2023	Percent	September 2023	Percent	Change
nginx	259,187,743	23.70%	254,196,148	23.43%	-0.27
Apache	228,121,898	20.86%	230,237,389	21.22%	0.36
Cloudflare	118,391,275	10.82%	115,640,533	10.66%	-0.17
OpenResty	95,236,092	8.71%	81,915,322	7.55%	-1.16

在性能方面，在达到百万并发后，`Apache` 的性能会逐渐下降，没有 `Nginx` 表现优异。

`Nginx` 是一款自由的、开源的、高性能的 `HTTP` 服务器和反向代理服务器，特点是占有内存少，并发能力强；同时也是一个 `IMAP` 、 `POP3` 、 `SMTP` 代理服务器；`Nginx` 可以作为一个 `HTTP` 服务器进行网站的发布处理，另外 `Nginx` 可以作为反向代理进行负载均衡的实现。

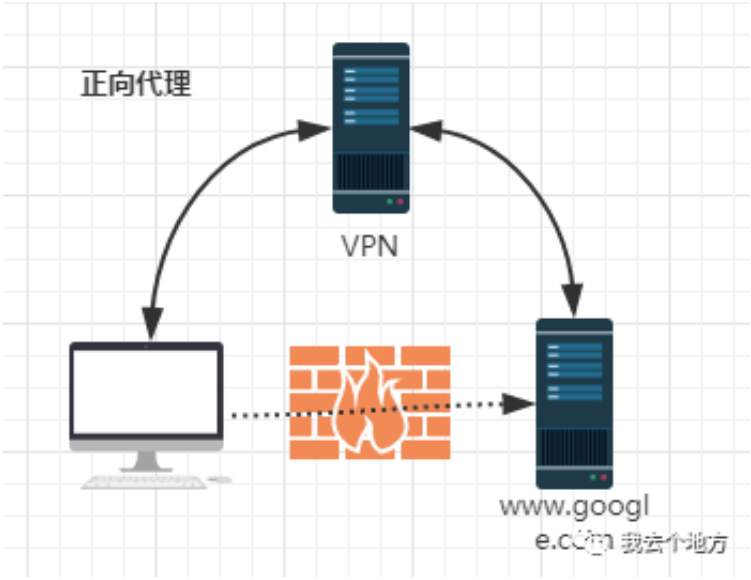
目前使用 `Nginx` 的公司有很多，比如百度、京东、新浪、网易、腾讯、淘宝等，足见它的优秀。

三、正向代理和反向代理

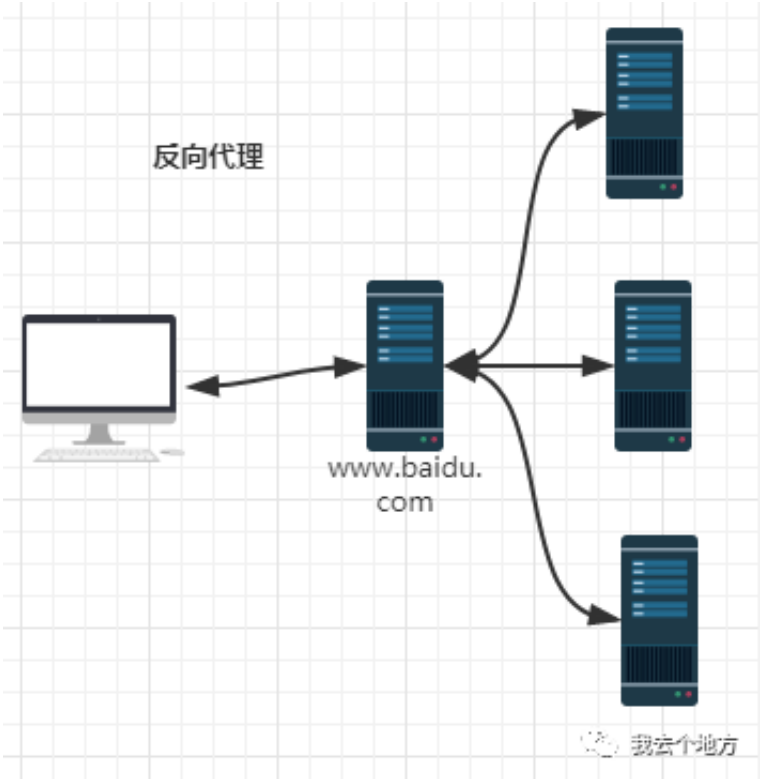
上面提到了反向代理这个名词。需要做下解释。

代理这个名词很容易理解，生活中到处都是代理。比如我们要买车票，本来是需要到车站买的，很麻烦，那么附近可能就有一家店可以卖一样的车票给你，那么这家店就是车站的代理。

那么正向代理和反向代理我们可能就会很迷惑了。



正向代理的工作原理就像一个跳板，比如：我访问不了`google.com`，但是我能访问一个代理服务器A，A能访问`google.com`，于是我先连上代理服务器A，告诉他我需要`google.com`的内容，A就去取回来，然后返回给我。从网站的角度，只在代理服务器来取内容的时候有一次记录，有时候并不知道是用户的请求，也隐藏了用户的资料，这取决于代理告不告诉网站。

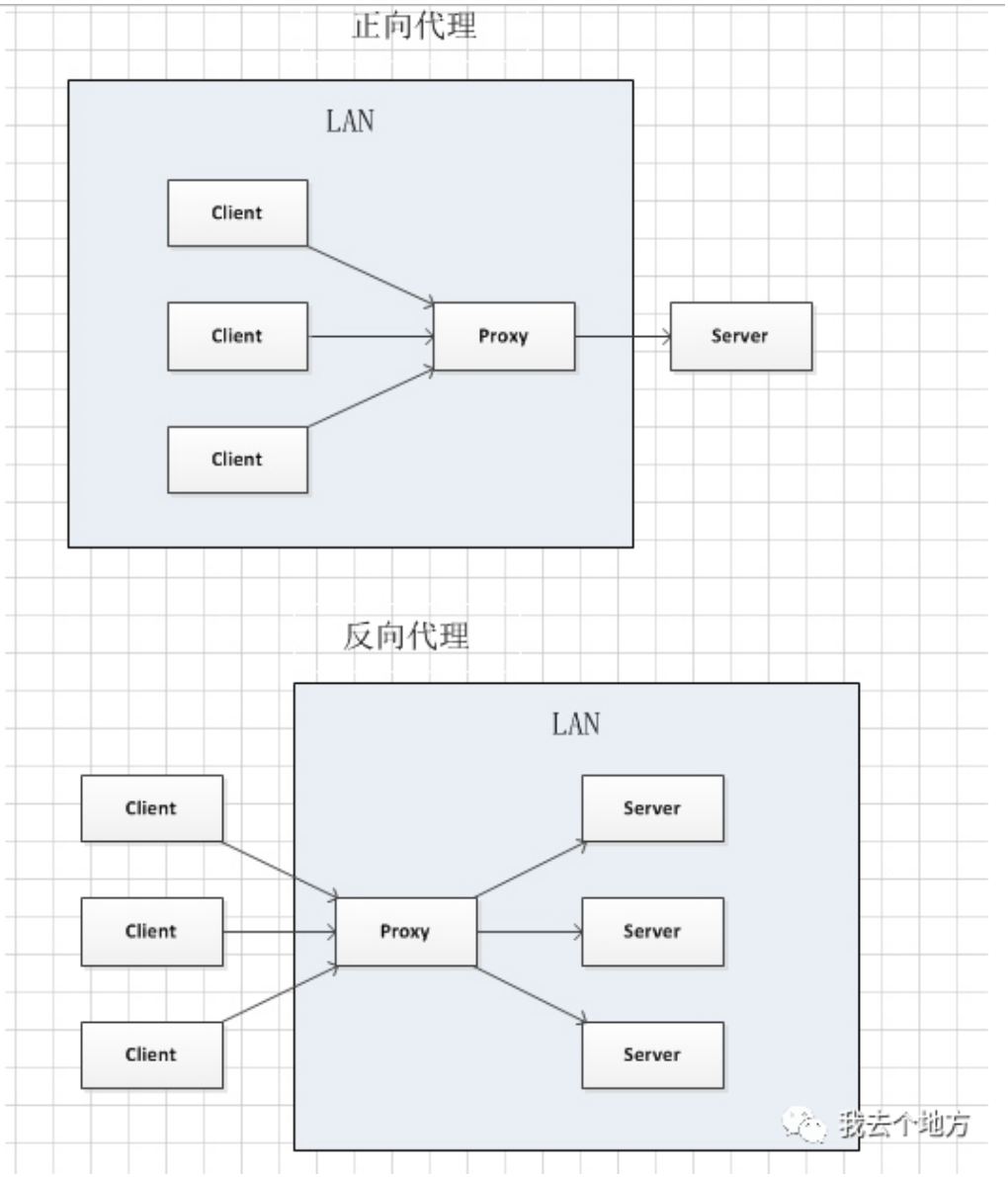


反向代理（`Reverse Proxy`）方式是指以代理服务器来接受`internet`上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给`internet`上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

简单来说：

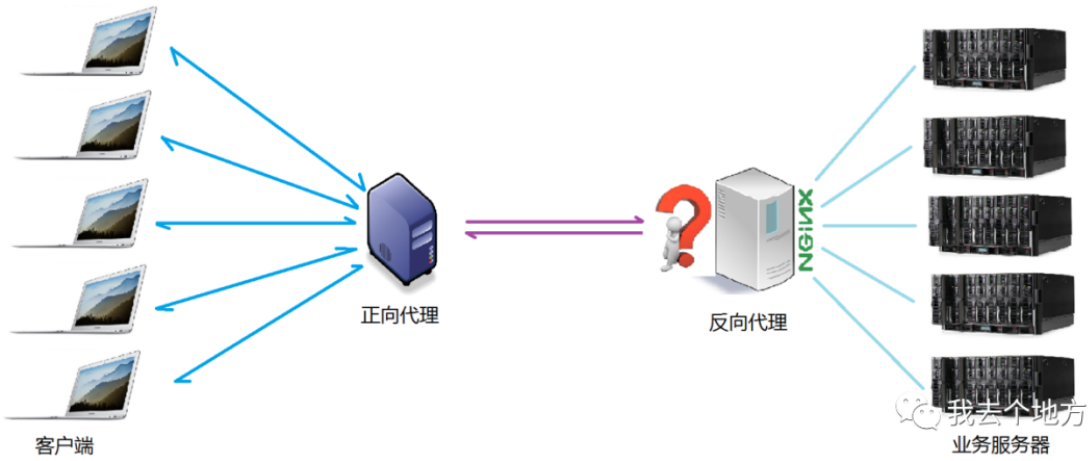
- **正向代理是不知道客户端是谁**，代理是一个跳板，所有客户端通过这个跳板来访问到对应的内容。
- **反向代理是不知道服务端是谁**，用户的请求被转发到内部的某台服务器去处理。

再用一张图来显示下两者的区别：



四、负载均衡算法

反向代理的模式里面，就会涉及上面说的一个名词叫做：负载均衡。它如何来选择服务器来响应呢？又或者说 `Nginx` 是以什么样的规则进行请求分发呢？分发的规则是否可以控制或改变呢？



> 负载量：客户端发送的、Nginx反向代理服务器接收到的请求数量。
> 负载均衡：将接收到的请求按照规则分发的过程。如何分发，就是算法实现的啦！

说的很高大上，其实就是 `Nginx` 作为反向代理的时候，到底把请求分发给谁的问题。最简单的情况就是轮询，各个服务器平均承担请求。那么十个负载量分发过来，背后有5个节点的话，那么每个节点处理两个。

不患寡而患不均，现在大家都一样多好！但是，有福同享，有难就不一定能同担，有些服务器性能较差，有些服务器性能较强，是不是能者多劳，多劳多得？就像工作，多劳多得才能激发个人潜能，将工作做的更好！

当然了，上面说的只是其中两种算法，下面我们简单列举说明下几种典型的策略：

> 1、轮询（默认）：每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。
> 2、weight：指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。
> 3、ip_hash：每个请求按访问ip的hash结果分配，这样每个访客固定访问同一个后端服务器，可以解决session的问题。但是不能解决宕机问题。前三种是nginx自带的，直接在配置文件中配置即可使用。
> 4、fair（第三方）：按后端服务器的响应时间来分配请求，响应时间短的优先分配。
> 5、url_hash（第三方）：按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为缓存时比较有效。

后续我们通过实战来具体看一看这几种负载均衡算法的表现情况。