一个网页打开的全过程

1、概要。。。

2、分析

众所周知，打开一个网页的过程中，浏览器会因页面上的 css/js/image 等静态资源会多次发起连接请求，所以我们暂且把这个网页加载过程分成两部分：

1. html(jsp/php/aspx) 页面加载(假设存在简单的 Nginx 负载均衡)
2. css/js/image 等 网页静态资源加载(假设使用 CDN)

2.1 页面加载

先上一张图，直观明了地让大家了解下基本流程，然后我们再逐一分析。

2.1.1 DNS 解析

什么是 DNS 解析？当用户输入一个网址并按下回车键的时候，浏览器得到了一个域名。而在实际通信过程中，我们需要的是一个 IP 地址。因此我们需要先把域名转换成相应的 IP 地址，这个过程称作 DNS 解析。

1）浏览器首先搜索浏览器自身缓存的 DNS 记录。

或许很多人不知道，浏览器自身也带有一层 DNS 缓存。Chrome 缓存 1000 条 DNS 解析结果，缓存时间大概在一分钟左右。

（Chrome 浏览器通过输入：chrome://net-internals/#dns 打开 DNS 缓存页面）

2）如果浏览器缓存中没有找到需要的记录或记录已经过期，则搜索 hosts 文件和操作系统缓存。

在 Windows 操作系统中，可以通过 ipconfig /displaydns 命令查看本机当前的缓存。

通过 hosts 文件，你可以手动指定一个域名和其对应的 IP 解析结果，并且该结果一旦被使用，同样会被缓存到操作系统缓存中。

Windows 系统的 hosts 文件在%systemroot%\system32\drivers\etc 下，linux 系统的 hosts 文件在/etc/hosts 下。

3）如果在 hosts 文件和操作系统缓存中没有找到需要的记录或记录已经过期，则向域名解析服务器发送解析请求。

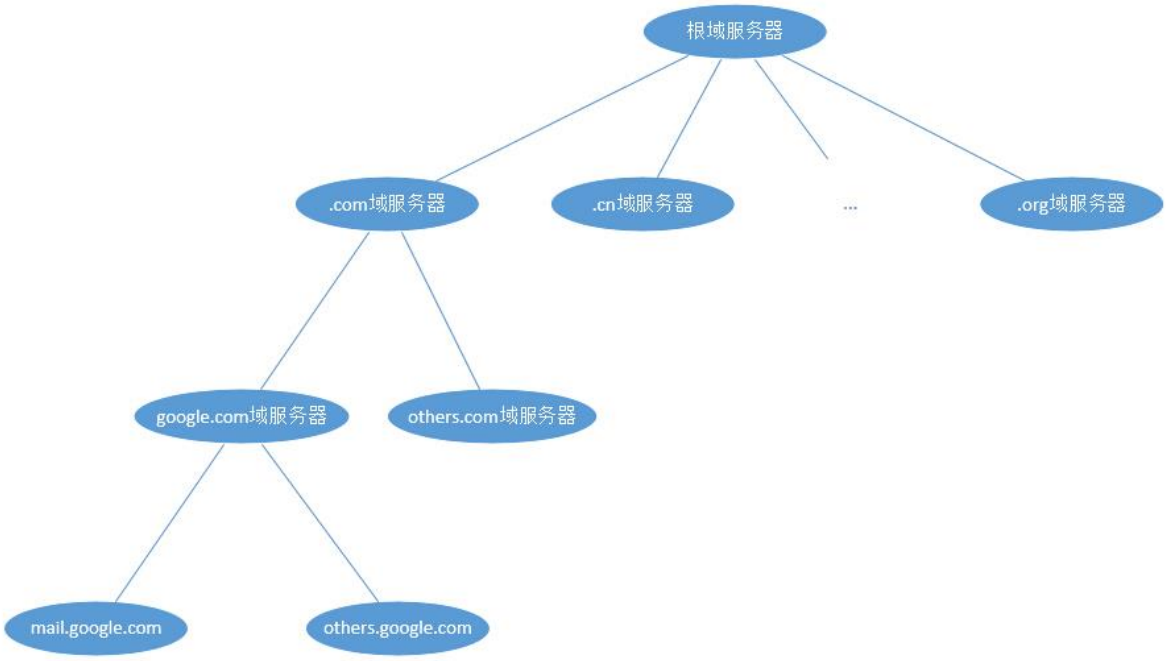
其实第一台被访问的域名解析服务器就是我们平时在设置中填写的 DNS 服务器一项，当操作系统缓存中也没有命中的时候，系统会向 DNS 服务器正式发出解析请求。这里是真正意义上开始解析一个未知的域名。

一般一台域名解析服务器会被地理位置临近的大量用户使用（特别是 ISP 的 DNS），一般常见的网站域名解析都能在这里命中。

4）如果域名解析服务器也没有该域名的记录，则开始递归+迭代解析。

这里我们举个例子，如果我们要解析的是 mail.google.com。

首先我们的域名解析服务器会向根域服务器（全球只有 13 台）发出请求。显然，仅凭 13 台服务器不可能把全球所有 IP 都记录下来。所以根域服务器记录的是 com 域服务器的 IP、cn 域服务器的 IP、org 域服务器的 IP……。如果我们要查找.com 结尾的域名，那么我们可以到 com 域服务器去进一步解析。所以其实这部分的域名解析过程是一个树形的搜索过程。



根域服务器告诉我们 com 域服务器的 IP。

接着我们的域名解析服务器会向 com 域服务器发出请求。根域服务器并没有 mail.google.com 的 IP，但是却有 google.com 域服务器的 IP。

接着我们的域名解析服务器会向 google.com 域服务器发出请求。...

如此重复，直到获得 mail.google.com 的 IP 地址。

为什么是递归：问题由一开始的本机要解析 mail.google.com 变成域名解析服务器要解析 mail.google.com，这是递归。

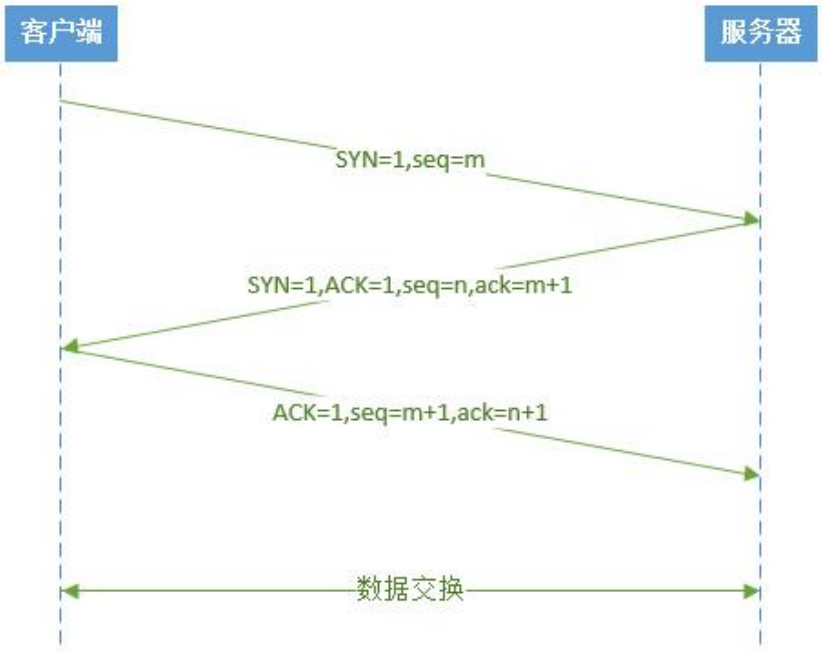
为什么是迭代：问题由向根域服务器发出请求变成向 com 域服务器发出请求再变成向 google.com 域发出请求，这是迭代。

5）获取域名对应的 IP 后，一步步向上返回，直到返回给浏览器。

2.1.2 发起 TCP 请求

浏览器会选择一个大于 1024 的本机端口向目标 IP 地址的 80 端口发起 TCP 连接请求。经过标准的 TCP 握手流程，建立 TCP 连接。

关于 TCP 协议的细节，这里就不再阐述。这里只是简单地用一张图说明一下 TCP 的握手过程。如果不了解 TCP，可以选择跳过此段，不影响本文其他部分的浏览。



2.1.3 发起 HTTP 请求

其本质是在建立起的 TCP 连接中，按照 HTTP 协议标准发送一个索要网页的请求。

2.1.4 负载均衡

什么是负载均衡？当一台服务器无法支持大量的用户访问时，将用户分摊到两个或多个服务器上的方法叫负载均衡。

什么是 Nginx？Nginx 是一款面向性能设计的 HTTP 服务器，相较于 Apache、lighttpd 具有占有内存少，稳定性高等优势。

负载均衡的方法很多，Nginx 负载均衡、LVS-NAT、LVS-DR 等。这里，我们以简单的 Nginx 负载均衡为例。关于负载均衡的多种方法详情大家可以 Google 一下。

Nginx 有 4 种类型的模块：core、handlers、filters、load-balancers。

我们这里讨论其中的 2 种，分别是负责负载均衡的模块 load-balancers 和负责执行一系列过滤操作的 filters 模块。

1）一般，如果我们的平台配备了负载均衡的话，前一步 DNS 解析获得的 IP 地址应该是我们 Nginx 负载均衡服务器的 IP 地址。所以，我们的浏览器将我们的网页请求发送到了 Nginx 负载均衡服务器上。

2）Nginx 根据我们设定的分配算法和规则，选择一台后端的真实 Web 服务器，与之建立 TCP 连接、并转发我们浏览器发出去的网页请求。

Nginx 默认支持 RR 轮转法 和 ip_hash 法 这 2 种分配算法。

前者会从头到尾一个个轮询所有 Web 服务器，而后者则对源 IP 使用 hash 函数确定应该转发到哪个 Web 服务器上，也能保证同一个 IP 的请求能发送到同一个 Web 服务器上实现会话粘连。

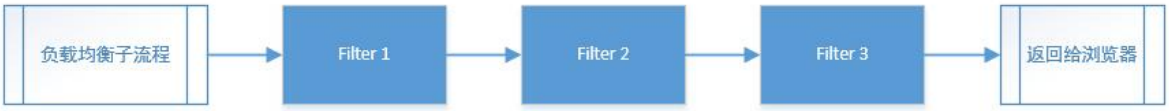
也有其他扩展分配算法，如：

fair：这种算法会选择相应时间最短的 Web 服务器

url_hash：这种算法会使得相同的 url 发送到同一个 Web 服务器

3）Web 服务器收到请求，产生响应，并将网页发送给 Nginx 负载均衡服务器。

4）Nginx 负载均衡服务器将网页传递给 filters 链处理，之后发回给我们的浏览器。



而 Filter 的功能可以理解成先把前一步生成的结果处理一遍，再返回给浏览器。比如可以将前面没有压缩的网页用 gzip 压缩后再返回给浏览器。

2.1.5 浏览器渲染

1）浏览器根据页面内容，生成 DOM Tree。根据 CSS 内容，生成 CSS Rule Tree(规则树)。调用 JS 执行引擎执行 JS 代码。

2）根据 DOM Tree 和 CSS Rule Tree 生成 Render Tree(呈现树)

3）根据 Render Tree 渲染网页

但是在浏览器解析页面内容的时候，会发现页面引用了其他未加载的 image、css 文件、js 文件等静态内容，因此开始了第二部分。

2.2 网页静态资源加载

以阿里巴巴的淘宝网首页的 logo 为例，其 url 地址为 img.alicdn.com/tps/i2/TB1bNE7LFXXXXaOXFXXwFSA1XXX-292-116.png_145x145.jpg

我们清楚地看到了 url 中有 cdn 字样。

什么是 CDN？如果我在广州访问杭州的淘宝网，跨省的通信必然造成延迟。如果淘宝网能在广东建立一个服务器，静态资源我可以直接从就近的广东服务器获取，必然能提高整个网站的打开速度，这就是 CDN。CDN 叫内容分发网络，是依靠部署在各地的边缘服务器，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度。

接下来的流程就是浏览器根据 url 加载该 url 下的图片内容。本质上是浏览器重新开始第一部分的流程，所以这里不再重复阐述。区别只是负责均衡服务器后端的服务器不再是应用服务器，而是提供静态资源的服务器。