第一步：获取IP地址

（1）在浏览器输入地址之后，注意，咱们这个**baidu是域名**，不是IP地址；

（2）这个时候浏览器会去找这个域名对应的是哪个IP地址，一般来说，会先找我们电脑**本地硬盘的 hosts** 文件，看一下有没有相关规则；

（3）如果本地hosts文件没有找到IP地址，那这个时候就去会去**DNS服务器**找,并返回IP地址；

第二步：发请求，建立TCP连接（三次握手）

（1）这里的细节我不过多说，我们只需要知道，上一步获取到IP地址之后，客户端开始向这个IP地址发送请求前，会先进行**TCP连接**；

三次握手是指这个连接的过程中：

1）客户端先发送一次报文给服务器；

2）服务端收到，再发一个给客户端；

3）客户端接受到，再发一次报文，同时也把这次的HTTP请求一起发过去

（2）经过握手之后，TCP连接成功，服务端这时也接受到了请求（**在最后一次握手**），接着处理之后把响应发回给客户端的同时，服务端会先**单方面关闭TCP连接**；

（3）客户端接受到请求数据，也把**TCP连接关闭**，这时才算**完成一次请求**；

小结论：从这里我们就可以知道，为什么要说前端减少HTTP请求会对性能有优化作用，就是因为你每一个请求它都要跑上面的过程，握手耗时

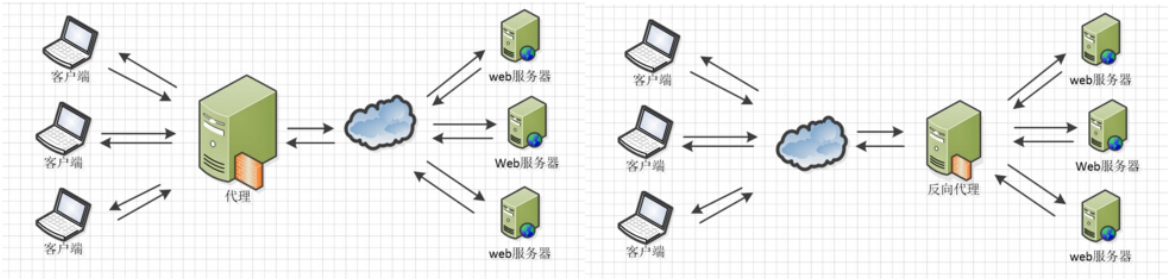
第三步：浏览器拿到html文件，开始渲染，并且发送请求获取嵌入在HTML 中的资源（如CSS、JS、图片、音频、视频等）

（这里后面再补充）

==========

Nginx是Igor Sysoev为俄罗斯访问量第二的rambler.ru站点设计开发的。Nginx的稳定性、功能集、示例配置文件和低系统资源的消耗让他后来居上

1、Http代理，反向代理：作为web服务器最常用的功能之一，尤其是反向代理。



Nginx在做反向代理时，提供性能稳定，并且能够提供配置灵活的转发功能。Nginx可以根据不同的正则匹配，采取不同的转发策略，比如图片文件结尾的走文件服务器，动态页面走web服务器，只要你正则写的没问题，又有相对应的服务器解决方案，你就可以随心所欲的玩。并且Nginx对返回结果进行错误页跳转，异常判断等。如果被分发的服务器存在异常，他可以将请求重新转发给另外一台服务器，然后自动去除异常服务器。

（1）正向代理

说明：

正向代理(forward)是一个位于客户端【用户A】和原始服务器(origin server)【服务器B】之间的服务器【代理服务器Z】，为了从原始服务器取得内容，用户A向代理服务器Z发送一个请求并指定目标(服务器B)，然后代理服务器Z向服务器B转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

通俗大白话：

正向代理**需要在客户端设置**，在这个过程中，**真正的服务器B**并不知道到底是哪个客户端发起的请求，因为它所有的请求都是来自**代理服务器Z**，所以说，在正向代理中，我们会说此时客户端是透明的。

（2）反向代理

说明：

反向代理正好与正向代理相反，对于客户端而言代理服务器就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的命名空间(name-space)中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端。

通俗大白话：

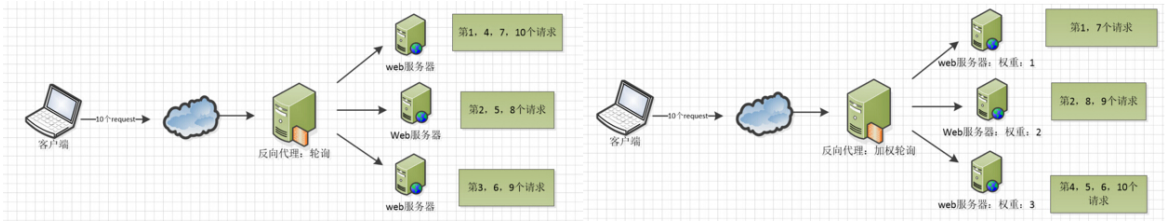
反向代理**不需要在客户端设置**，在这个过程，客户端只需要请求这个代理服务器，这个代理服务器会自动根据相关设置去请求对应的真正的服务器，所以说，在反向代理中，我们会说此时服务端是透明的。

反向代理用途：

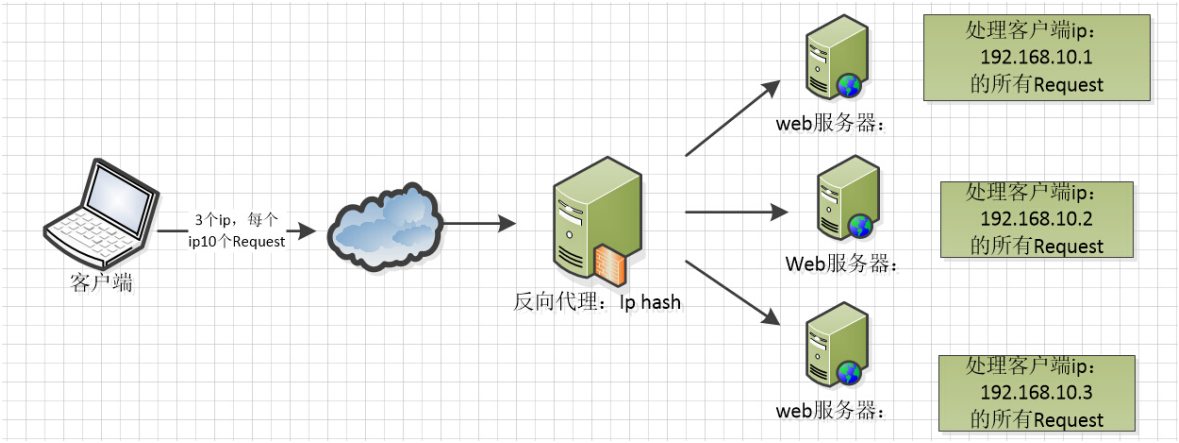
1）负载均衡，减轻服务器压力（目前只知道这个）；

2、负载均衡

Nginx提供的负载均衡策略有2种：内置策略和扩展策略。内置策略为轮询，加权轮询，Ip hash。扩展策略，就天马行空，只有你想不到的没有他做不到的啦，你可以参照所有的负载均衡算法，给他一一找出来做下实现。



Ip hash算法，对客户端请求的ip进行hash操作，然后根据hash结果将同一个客户端ip的请求分发给同一台服务器进行处理，可以解决session不共享的问题。



3、web缓存

Nginx可以对不同的文件做不同的缓存处理，配置灵活，并且支持FastCGI_Cache，主要用于对FastCGI的动态程序进行缓存。配合着第三方的ngx_cache_purge，对制定的URL缓存内容可以的进行增删管理。

4、Nginx相关地址

源码：https://trac.nginx.org/nginx/browser

官网：http://www.nginx.org/

==========

1、全局块：配置影响nginx全局的指令。一般有运行nginx服务器的用户组，nginx进程pid存放路径，日志存放路径，配置文件引入，允许生成worker process数等。
2、events块：配置影响nginx服务器或与用户的网络连接。有每个进程的最大连接数，选取哪种事件驱动模型处理连接请求，是否允许同时接受多个网路连接，开启多个网络连接序列化等。
3、http块：可以嵌套多个server，配置代理，缓存，日志定义等绝大多数功能和第三方模块的配置。如文件引入，mime-type定义，日志自定义，是否使用sendfile传输文件，连接超时时间，单连接请求数等。
4、server块：配置虚拟主机的相关参数，一个http中可以有多个server。
5、location块：配置请求的路由，以及各种页面的处理情况。
下面给大家上一个配置文件，作为理解，同时也配入我搭建的一台测试机中，给大家示例。

```
########### 每个指令必须有分号结束。#################
#user administrator administrators;  #配置用户或者组，默认为nobody nobody。
#worker_processes 2;  #允许生成的进程数，默认为1
#pid /nginx/pid/nginx.pid;   #指定nginx进程运行文件存放地址
error_log log/error.log debug;  #制定日志路径，级别。这个设置可以放入全局块，http块，server块，级别以此为：debug|info|notice|warn|error|crit|alert|emerg
events {
    accept_mutex on;   #设置网路连接序列化，防止惊群现象发生，默认为on
    multi_accept on;  #设置一个进程是否同时接受多个网络连接，默认为off
    #use epoll;      #事件驱动模型，select|poll|kqueue|epoll|resig|/dev/poll|eventport
    worker_connections  1024;    #最大连接数，默认为512
}
http {
    include       mime.types;   #文件扩展名与文件类型映射表
    default_type  application/octet-stream; #默认文件类
```

```
型，默认为text/plain
    #access_log off; #取消服务日志
    log_format myFormat '$remote_addr-$remote_user
[$time_local] $request $status $body_bytes_sent
$http_referer $http_user_agent
$http_x_forwarded_for'; #自定义格式
    access_log log/access.log myFormat;  #combined为日
志格式的默认值
    sendfile on;   #允许sendfile方式传输文件，默认为off，
可以在http块，server块，location块。
    sendfile_max_chunk 100k;  #每个进程每次调用传输数量不
能大于设定的值，默认为0，即不设上限。
    keepalive_timeout 65;  #连接超时时间，默认为75s，可以
在http，server，location块。

    upstream mysvr {
      server 127.0.0.1:7878;
      server 192.168.10.121:3333 backup;  #热备
    }
    error_page 404 https://www.baidu.com; #错误页
    server {
        keepalive_requests 120; #单连接请求上限次数。
        listen       4545;   #监听端口
        server_name  127.0.0.1;   #监听地址
        location  ~*^.+$ {       #请求的url过滤，正则匹
配，~为区分大小写，~*为不区分大小写。
           #root path;  #根目录
           #index vv.txt;  #设置默认页
           proxy_pass  http://mysvr;  #请求转向mysvr 定
义的服务器列表
           deny 127.0.0.1;  #拒绝的ip
           allow 172.18.5.54; #允许的ip
        }
    }
}
```

```
      location  /hstravel-access-mgt/ {
           proxy_set_header Host "$http_host";
```

```
            proxy_pass http://192.168.228.181:8080/
hstravel-access-mgt/;
            proxy_set_header Header-Url "${scheme}://
$http_host$request_uri";
            proxy_set_header X-Forwarded-For
$request_uri;
            proxy_redirect default;
      }
```

上面是nginx的基本配置，需要注意的有以下几点：
1、1.$remote_addr 与$http_x_forwarded_for 用以记录客户端的ip地址； 2.$remote_user： 用来记录客户端用户名称； 3.$time_local： 用来记录访问时间与时区；4.$request： 用来记录请求的url与http协议；
5.$status： 用来记录请求状态；成功是200， 6.$body_bytes_s ent： 记录发送给客户端文件主体内容大小；7.$http_referer： 用来记录从那个页面链接访问过来的； 8.$http_user_agent： 记录客户端浏览器的相关信息；
2、惊群现象：一个网路连接到来，多个睡眠的进程被同时叫醒，但只有一个进程能获得链接，这样会影响系统性能。
3、每个指令必须有分号结束
==========

Nginx代理服务的配置说明

1、上一篇中我们在http模块中有下面的配置，当代理遇到状态码为404时，我们把404页面导向百度。

```
error_page 404 https://www.baidu.com; #错误页
```

然而这个配置，细心的朋友可以发现他并没有起作用。
如果我们想让他起作用，我们必须配合着下面的配置一起使用

```
proxy_intercept_errors on;      #如果被代理服务器返回的状态码为400或者大于400，设置的error_page配置起作用。默认为off。
```

2、如果我们的代理只允许接受get，post请求方法的一种

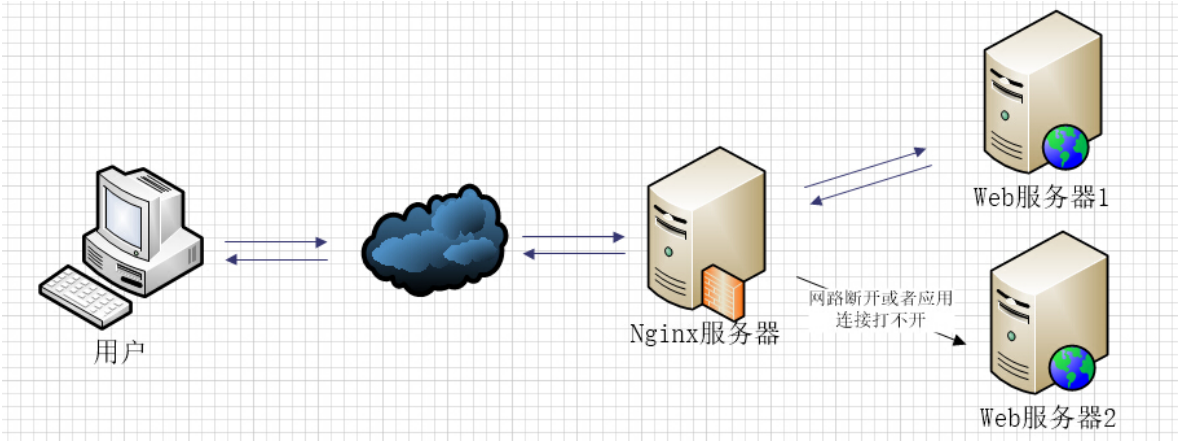
```
proxy_method get;      #支持客户端的请求方法。post/get;
```

3、设置支持的http协议版本

```
proxy_http_version 1.0 ; #Nginx服务器提供代理服务的http协议版本1.0，1.1，默认设置为1.0版本
```

4、如果你的nginx服务器给2台web服务器做代理，负载均衡算法采用轮询，那么当你的一台机器web程序iis关闭，也就是说web不能访问，那么

nginx服务器分发请求还是会给这台不能访问的web服务器，如果这里的响应连接时间过长，就会导致客户端的页面一直在等待响应，对用户来说体验就打打折扣，这里我们怎么避免这样的情况发生呢。这里我配张图来说明下问题。



5、如果使用upstream指令配置啦一组服务器作为被代理服务器，服务器中的访问算法遵循配置的负载均衡规则，同时可以使用该指令配置在发生哪些异常情况时，将请求顺次交由下一组服务器处理。

```
proxy_next_upstream timeout;  #反向代理upstream中设置的服务器组，出现故障时，被代理服务器返回的状态值。error|timeout|invalid_header|http_500|http_502|http_503|http_504|http_404|off
```

error：建立连接或向被代理的服务器发送请求或读取响应信息时服务器发生错误。

timeout：建立连接，想被代理服务器发送请求或读取响应信息时服务器发生超时。

invalid_header:被代理服务器返回的响应头异常。

off:无法将请求分发给被代理的服务器。

http_400，....:被代理服务器返回的状态码为400，500，502，等。

6、如果你想通过http获取客户的真是ip而不是获取代理服务器的ip地址，那么要做如下的设置。

```
proxy_set_header Host $host; #只要用户在浏览器中访问的域名绑定了 VIP VIP 下面有RS；则就用$host ；host是访问URL中的域名和端口  www.taobao.com:80
proxy_set_header X-Real-IP $remote_addr;  #把源IP【$remote_addr,建立HTTP连接header里面的信息】赋值给X-Real-
```

IP;这样在代码中 $X-Real-IP来获取 源IP
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;#在nginx 作为代理服务器时，设置的IP列表，会把经过的机器ip，代理机器ip都记录下来，用 【，】隔开；代码中用 echo $x-forwarded-for |awk -F, '{print $1}' 来作为源IP

关于X-Forwarded-For与X-Real-IP的一些相关文章我推荐一位博友的：HTTP 请求头中的 X-Forwarded-For ，这位博友对http协议有一系列的文章阐述，推荐大家去关注下。

7、下面是我的一个关于代理配置的配置文件部分，仅供参考。

```
    include       mime.types;    #文件扩展名与文件类型映射表
    default_type  application/octet-stream; #默认文件类型，默认为text/plain
    #access_log off; #取消服务日志
    log_format myFormat ' $remote_addr-$remote_user [$time_local] $request $status $body_bytes_sent $http_referer $http_user_agent $http_x_forwarded_for'; #自定义格式
    access_log log/access.log myFormat;  #combined为日志格式的默认值
    sendfile on;   #允许sendfile方式传输文件，默认为off，可以在http块，server块，location块。
    sendfile_max_chunk 100k;  #每个进程每次调用传输数量不能大于设定的值，默认为0，即不设上限。
    keepalive_timeout 65;  #连接超时时间，默认为75s，可以在http，server，location块。
    proxy_connect_timeout 1;   #nginx服务器与被代理的服务器建立连接的超时时间，默认60秒
    proxy_read_timeout 1; #nginx服务器想被代理服务器组发出read请求后，等待响应的超时间，默认为60秒。
    proxy_send_timeout 1; #nginx服务器想被代理服务器组发出write请求后，等待响应的超时间，默认为60秒。
    proxy_http_version 1.0 ; #Nginx服务器提供代理服务的http协议版本1.0，1.1，默认设置为1.0版本。
```

```
    #proxy_method get;     #支持客户端的请求方法。post/get;
    proxy_ignore_client_abort on;  #客户端断网时，nginx服务器是否终端对被代理服务器的请求。默认为off。
    proxy_ignore_headers "Expires" "Set-Cookie";  #Nginx服务器不处理设置的http相应投中的头域，这里空格隔开可以设置多个。
    proxy_intercept_errors on;     #如果被代理服务器返回的状态码为400或者大于400，设置的error_page配置起作用。默认为off。
    proxy_headers_hash_max_size 1024; #存放http报文头的哈希表容量上限，默认为512个字符。
    proxy_headers_hash_bucket_size 128; #nginx服务器申请存放http报文头的哈希表容量大小。默认为64个字符。
    proxy_next_upstream timeout;  #反向代理upstream中设置的服务器组，出现故障时，被代理服务器返回的状态值。error|timeout|invalid_header|http_500|http_502|http_503|http_504|http_404|off
    #proxy_ssl_session_reuse on; 默认为on，如果我们在错误日志中发现“SSL3_GET_FINSHED:digest check failed”的情况时，可以将该指令设置为off。
```

```
==================================================
==================================================
==================
```

（1）概念

跨域是由浏览器同源策略引起的，是指页面请求的接口地址，必须与页面url地址处于同域上（即域名，端口，协议相同）。这是为了防止某域名下的接口被其他域名下的网页非法调用，是浏览器对JavaScript施加的安全限制。

注意：通过后台调接口是不会有跨域问题的，而是会出现这个接口设置了不让其他服务器调用的其他问题

（2）怎么解决呢

虽然浏览器有限制，但是HTML中有两个属性是不受限制的，那就是**src**

和href属性

```
<img src="http://www.lizi.com/xxx.jpeg"></script>
<link rel="stylesheet" href="http://www.lizi.com/css/reset.css">
<script src="https://cdn.bootcss.com/jquery/3.4.0/jquery.min.js"></script>
```

2.1）JSONP

先说结论：这个方法是**前后端要互相配合**，比如约定那个函数的名字叫啥之类

2.2）CORS

先说结论：这个只需要**后端设置**就行了，前端不用
这个方法具体操作是设置接口的响应头，服务器设置HTTP响应头中Access-Control-Allow-Origin值，解除跨域限制。
不细说，到后面我再贴相关的参考链接（因为我也不懂）

但是这两个跨域方案都存在一个致命的缺陷，严重依赖后端的协助。
开发中遇到的每一个接口都需要提前找后端进行特殊处理。而且即使后端愿意帮忙，某些接口也不是随便能开放的（譬如已经在线上正式环境的接口）。 所以依赖后端协助的跨域解决方式都会在一定程度上限制前端的开发进度。
那么有没有前端独立就能解决的跨域方案呢？有的，我们可以利用「代理」或「反向代理」技术来解决开发中的跨域问题。

比较

- 从用途上来讲：
  - 正向代理的典型用途是为在防火墙内的局域网客户端提供访问Internet的途径。正向代理还可以使用缓冲特性减少网络使用率。
  - 反向代理的典型用途是为后端的多台服务器提供负载平衡，或为后端较慢的服务器提供缓冲服务。
- 从安全性来讲：
  - 正向代理允许客户端通过它访问任意网站并且隐藏客户端自身，因此你必须采取安全措施以确保仅为经过授权的客户端提供服务。
  - 反向代理对外都是透明的，访问者并不知道自己访问的是一个代理。
- 从使用方来看：

- 正向代理是浏览器端进行配置的，与服务器端无关，甚至可以对服务端隐藏。
- 反向代理是服务器端配置的，对浏览器端是透明的。

利用正向代理实现跨域

2.3）代理

在一般的开发过程中，上面两种方法在使用的时候都是需要后端配合的，而下面说的代理，是可以实现只需要在前端配置即可。

Nginx 配置中nginx和alias的区别分析

1： root和alias都可以定义在location模块中，都是用来指定请求资源的真实路径，比如：

```
location /i/ {
  root /data/w3;
}
```

请求 http://foofish.net/i/top.gif 这个地址时，那么在服务器里面对应的真正的资源是 /data/w3/i/top.gif文件

注意：真实的路径是root指定的值加上location指定的值 。

而 alias 正如其名，alias指定的路径是location的别名，不管location的值怎么写，资源的 真实路径都是 alias 指定的路径 ，比如：

```
location /i/ {
  alias /data/w3/;
}
```

同样请求 http://foofish.net/i/top.gif 时，在服务器查找的资源路径是： /data/w3/top.gif

其他区别：

1、 alias 只能作用在location中，而root可以存在server、http和location中。

2、alias 后面必须要用“/”结束，否则会找不到文件，而 root 则对 ”/” 可有可无。