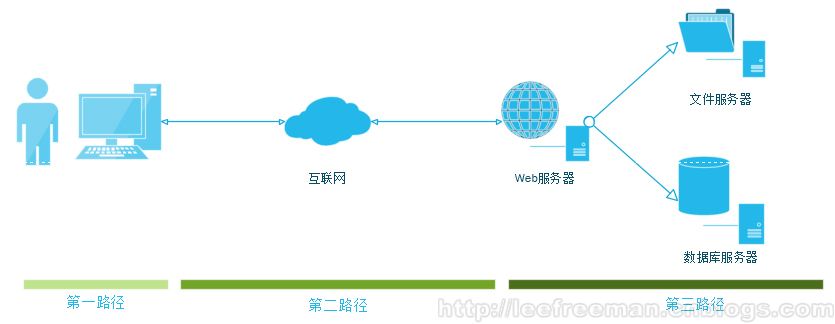
## 大型网站的灵魂——性能

1. **什么是性能**

有人说性能就是访问速度快慢，这是最直观的说法，也是用户的真实体验。一个用户从输入网址到按下回车键，看到网页的快慢，这就是性能。对于我们来说，需要去挖掘这个过程，因为这决定我们怎么去做性能优化。



用户访问网站的整个流程：用户输入网站域名，通过DNS解析，找到目标服务器IP，请求数据经互联网达到目标服务器，目标服务器收到请求数据，进行处理（执行程序、访问数据库、文件服务器等）。处理完成，将响应数据又经互联网返回给用户浏览器，浏览器得到结果进行计算渲染显示给用户。

我们把整个过程，分为三段路径：

1、第一段在用户和浏览器端，主要负责发出用户请求，以及接受响应数据进行计算渲染显示给用户；

2、第二段在网络上，负责对请求数据、响应数据的传输；

3、第三段在网站服务器端，负责对请求数据进行处理（执行程序、访问数据库、文件等），并将结果返回。

1. **第一路径**

第一路径花费的时间包括输入域名发起请求的时间和浏览器收到响应后计算渲染的时间。

输入域名发起请求，实质过程是：

1、用户在浏览器输入要访问的网站域名；

2、本地DNS请求网站授权的DNS服务器对域名进行解析，并得到解析结果即IP地址（并将IP地址缓存起来）。

3、向目标IP地址发出请求。

从这个过程我们可以看到，优化的地方主要是减少DNS解析次数，而如果用户浏览器设置了缓存，则再第二次访问相同域名的时候就不会去请求DNS服务器，直接用缓存中的IP地址发出请求。因此这个过程主要取决于浏览器的设置。现在主流的浏览器默认设置了DNS的预取功能（DNS Prefetch），当然你也可以主动告知浏览器我的网站需要做DNS预取：<meta http-equiv="x-dns-prefetch-control" content="on" />

浏览器将数据进行计算渲染的过程（这里css和js的下载根据引入文件的位置等决定）：

1、浏览器解析响应数据；

2、浏览器创建DOM树；

3、浏览器下载CSS样式，并应用到DOM树，进行渲染；

4、浏览器下载JS文件，开始解析执行；

5、显示给用户。

从这个过程，我们可以找出不少可以优化的地方。首先我们可以尽量控制页面大小，使得浏览器解析的时间更短；并且将多个CSS文件、JS文件文件合并压缩减少文件下载的次数和大小；另外注意将CSS放在页面前面，JS访问页面后面，这样便于页面首先能渲染出来，再执行js脚本，对于用户来说有更好的体验。最后我还可以设置浏览器缓存，下次访问时从缓存读取内容，减少http请求。

<meta http-equiv="Cache-Control" content="max-age=5" />：该代码说明了浏览器启用了缓存并在5秒内不会再次访问服务器。注意缓存的设置需要结合你的业务特性来适当配置。

还有一些简单的优化，如：

css样式放在html前面，并且进行了合并。

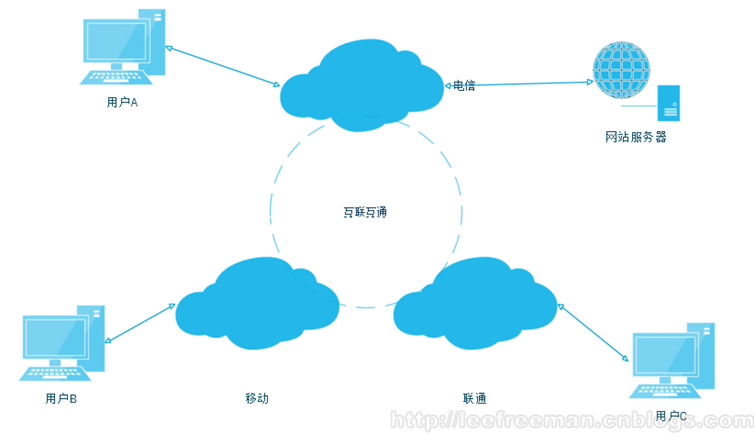
JS文件放在页尾及其合并。

1. **第二路径**

第二路径在网络上，花费的时间同样包括请求数据的传输时间和响应数据的传输时间，这个两个时间取决于数据传输的速度，这里我们要讲一个名词“带宽”。什么是带宽，我们经常说带宽10M，20M是什么意思？我的带宽20M，这意味着什么？我们知道带宽速度分为上行、下行速度，也就是上传和下载的速度。带宽20M对于用户来说则是下载速度20M（20×1024×1024比特率），换算成字节20M/8=2.5M。也就是说20M的带宽下载速度理论可达2.5M/s，而对于家庭用户而言上传速度一般比下载速度小的多，大约是不到十分之一。而对于网站服务器（企业用户）来说，则不然，一般上行速度等于下载速度。这也是运营商根据实际需求分配的，毕竟用户的主要需求是下载数据，而不是上传数据。

整个流程从传输方式看就是：用户发送请求数据（上传），网站服务器接受请求数据（下载），网站服务器返回响应数据（上传），用户接受响应数据（下载）。对于用户来说，上传数据是很小的（Url参数），而下载数据是较大的（响应数据）；对于服务器来说，下载数据是很小的（url参数），上传数据是较大（响应数据）。理解了这个，我们可以解释为什么有时用户反映为什么自己的带宽足够，但打开某些网站仍然很慢，就是因为尽管用户的下载速度很快，但网站服务器的上传速度很慢，这就像一个抽水管和一个出水管，不管抽水管再大，但出水管很小，同样抽到的水量是有限的。了解了这个原理我们来看怎么提高数据传输的速度，首先用户的上传、下载速度我们是无法决定的，我们能决定的是网站服务器的上传、下载速度，所以我们可以做的是适当的增加服务器带宽（带宽是很贵的，盲目的增加只会增加不必要成本）。购买合适的带宽需要根据网站业务特性、规模以及结合运维人员的经验来选择。通常可以考虑的算法，即根据一次响应数据的大小，乘以PV数，除以对应的高峰时间段，从而大致估算出网站带宽的需求。

下面我们继续进一步研究第二路径：



上图表示用户访问网站服务器时网络的大致情况，从图上可以看出假设网站服务器从电信网络接入，而用户A作为电信的宽带用户，则可以通过电信骨干网快速的访问到网站服务器。用户B,用户C作为移动和联通用户需要通过运营商的互联互通经过较长路径才能访问到服务器。

针对这种情况，我们可以采取以下方法来优化：

1、在各运营商发达的地区的IDC（互联网数据中心，可以理解成机房）部署网站服务器，各运营商的用户即可通过各自的骨干网访问服务器。

2、购买代理服务，也就是原来联通用户需要通过联通骨干网——>联通互联互通路由器——>电信骨干网——>网站服务器的过程。通过代理服务，代理服务器直连到电信骨干网，访问网站服务器。

3、在主要地区城市购买CDN服务，缓存对应的数据，用户可先从最近的CDN运营商获取请求数据。

1. **第三路径**

第三路径主要是网站服务器内部处理的过程，当中包括执行程序、访问文件、数据库等资源。

这是对于我们来说最可以发挥的地方：

1、使用缓存，根据需要使用本地缓存或分布式缓存；

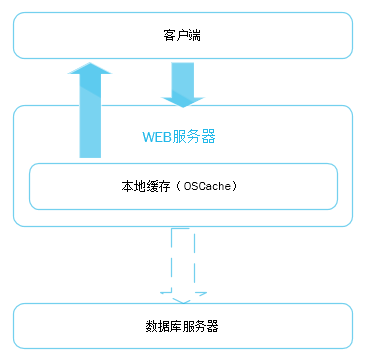
2、使用异步操作，这种方式不仅可以提高性能，也提高了系统的扩展性；

3、代码优化；

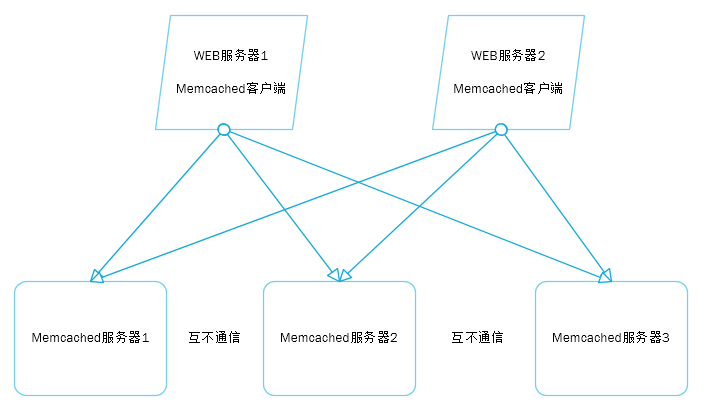
4、存储优化；

* 1. **缓存**

如果缓存数据较少，可以利用OSCache实现本地缓存：

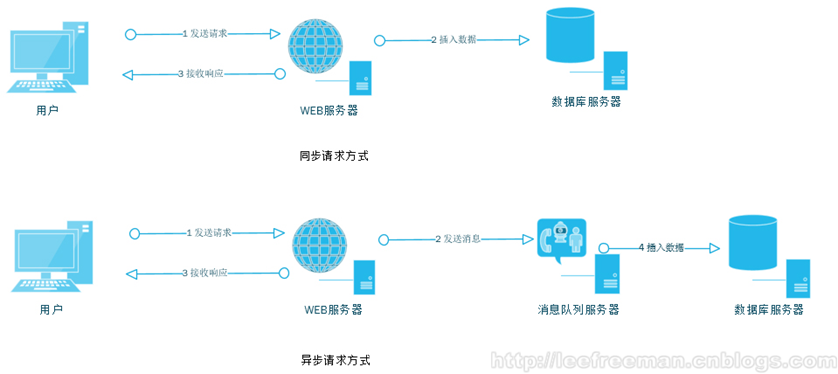


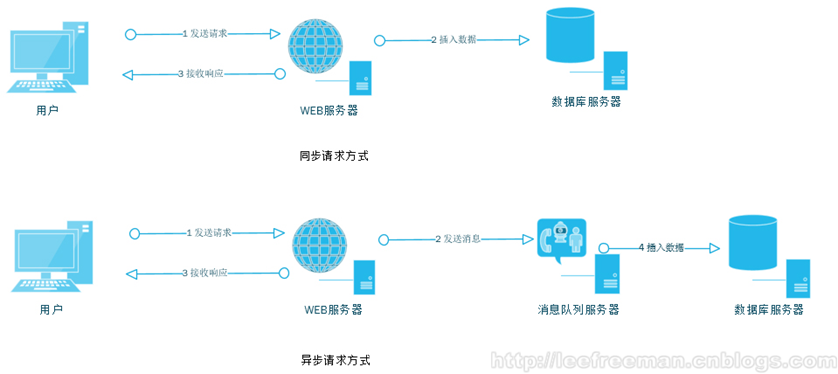
当缓存数据过多时，利用Memcached、redis等实现分布式缓存：



Memcached实现分布式缓存，缓存服务器之间是互不通信的，也就是我们可以方便的通过增加Memcached服务器对系统进行扩展。

* 1. **异步操作**





使用同步请求的方式，在高并发的情况下，会对数据库造成很大的压力，也会让用户感觉响应时间过长。异步请求方式，则可以快速的对用户做出响应，而具体的数据库操作请求，则通过消息队列服务器发送给数据库服务器，做具体的插入操作。插入操作的结果则已其他方式通知客户端。例如一般在订票系统当中，出票行为就是异步完成，最终的出票结果会以邮件或其他方式告知用户。

* 1. **代码优化**

这里就不在详细描述，参考：

http://www.cnblogs.com/leefreeman/p/3585032.html

* 1. **存储优化**

大型网站中海量的数据读写对磁盘造成很大压力，系统最大的瓶颈还是在磁盘的读写。可以考虑使用磁盘阵列、分布式储存来改善存储的性能。

1. **性能的指标和测试**

上面通过解析用户访问网站的过程来思考怎么提高用户感知的性能，对于用户来言性能就是快和慢。但对于我们来说，不能这样简单描述，我们需要去量化他，用一些数据指标去衡量它。这里讲到几个名词：响应时间、并发量、吞吐量。

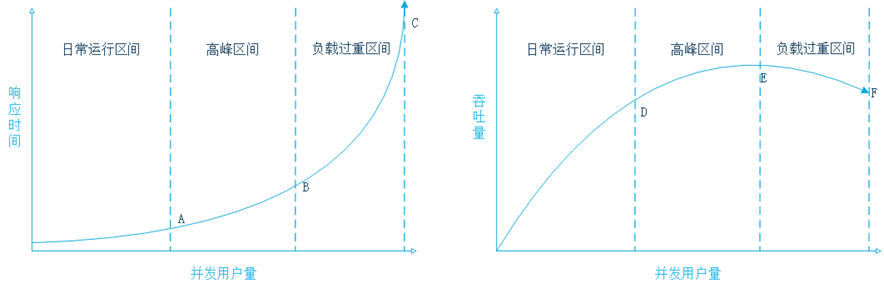
响应时间：就是用户发出请求到收到响应数据的时间；

并发量：就是系统同时能处理多少用户请求；

吞吐量：就是单位时间内系统处理的请求数量；

为了通俗的了解这三个概念，我们以高速公路的收费站为例子：响应时间是指一辆车经过收费站的时间，也就是车辆从进入收费站、付钱、开闸、离开收费站的时间；并发量是指这个收费站同时能通行多少辆车，可以理解为收费站的出口数量。吞吐量是指：在一段时间内，这个收费站通往了多少了车。这个例子不晓得恰不恰当。

对于性能测试来说，基本也是围绕这些方面来测试，下图说明了性能测试的过程：



左图表示响应时间和并发用户量的二维坐标图，从图上可以看出，并发用户量在一定量增加时，响应时间很短，并且没有太大的起伏，这表示系统目前处于日常运行期，可以很快处理用户请求（A点之前）；随着并发量的增加，系统处于请求高峰期，但仍然可以有序的处理用户请求，响应时间较日常有所增加（A、B之间）；当并发量增加到一定数量时，超过了系统的负载能力，系统处于濒临崩溃的边缘（B、C之间），响应时间严重过长，直到系统崩溃。

右图表示吞吐量与并发用户量的二维坐标图，可以看出，随着并发用户量的增加，吞吐量逐渐增加；在并发量到达一定量时，由于系统处理能力达到最大，吞吐量增加放缓；当并发量超过系统负载时（E点），系统处理能力开始下降，不能再请求增加的用户请求，吞吐量反而降低。

1. **小结**

本文通过用户访问网站的过程，分析了三个路径过程中提高性能的想法和手段，最后介绍了描述性能的指标，并对性能测试做了简要说明。