**使用缓存的优点**

* 缓存减少了冗余的数据传输，因为毕竟每次http事务请求的东西都是一样的时候，多次发送同样的数据是不必要和冗余的！
* 缓存缓解了网络瓶颈的问题，不需要更多的带宽就能够更快地加载页面！
* 缓存降低了对原始服务器的要求，因为想象一下，从一个性能很差劲的原始服务器和从一个性能和牛逼的缓存服务器请求事务，肯定会弥补服务器的缺点的，同时也会减少服务器过载情况，因为大部分请求都由缓存代劳处理了！
* 缓存降低了距离时延，因为从较远的地方加载页面会更慢一些！

**冗余的数据传输**

* 每次都从原始服务器拿数据，那么带来的后果就是：多次发送重复的数据浪费流量、耗费昂贵的网络带宽从而降低传输速率、加大服务器的负载。而有了缓存之后这些问题都可以迎刃而解！

**带宽瓶颈**

* 带宽瓶颈说明的问题：很多网络为本地客户端配置的带宽要比远程服务器配置的带宽要宽，如果在这种状况下客户端去请求远程服务器，那么客户端将会以一种的较低的速度去请求服务端，从而没有发挥出客户端带宽宽的长处！如果在客户端方向配置一个高速缓存服务器，那么就可以很快的得到响应，由此也看出带宽对报文传输速率的影响！

**瞬间拥塞**

* 瞬间拥塞描述的是这样一种情况：一个爆炸性的新闻和热点事件，如果再没有配置缓存的情况下，那么在短时间之内，服务器将会收到突变的请求增长，负荷会爆炸性增长，肯定会吃不消的！但是有了缓存，可以大大分担服务器的负载数量！

**距离时延**

* 距离时延说明的一个问题就是传输数据过程这个过程是需要时间的，而且路程越长，那么需要的时间也会越多，即时延越长。所以在距离客户端较近的地方部署缓存服务器，减小了传输路程，那么就减小了传输时延！

**命中与未命中的**

* \*缓存命中与缓存未命中：\*一次http事务请求如果是从得到的响应是从缓存得到的原始响应副本，那么这样的过程就称之为缓存命中。反之，如果缓存没有响应的副本，而要去请求原始服务，那么就把这个过程称之为缓存未命中！
* http再验证:原始响应内容是在变化的，所以缓存应该在文档“过期时间”之后去验证缓存的副本是不是新鲜的，这个过程就叫做http再验证！如果再验证之后得知缓存副本是新鲜的，那么原始服务器返回304 not modified。此时，称之为再验证命中或缓存慢命中！如果得知缓存不是新鲜的，那么服务器返回200 ok。此时，称之为再验证未命中！如果原始对象已经被删除了，返回404 not found响应。相应地缓存副本要删除。（注：虽然再验证命中需要跟原始服务器沟通一次，但是它与直接请求服务器相比，还是要快一点，因为再验证命中只是返回了一些新的过期时间有关的新首部而已，并没有发送主体对象。）
* *命中率*指由缓存返回副本事务在全部事务中所占的比例，称为缓存命中率。这个数据实际中意义不是很大！而*字节命中率*从资源大小总量的角度说明缓存命中所占的比例。因为他从数据流量的角度出发，所以实际中这个数据的意义挺大的！
* *区分命中和未命中*， http没有相应的机制来告知客户端响应是从缓存得到的还是从原始服务器得到的！但我们可以从http响应报文首部中的date字段得知这一情况：如果这个字段的时间比当前时间更早得多，说明这是从缓存得到的，因为date描述的服务器第一次响应的时间，而缓存是不会对这个字段进行修改的！

**缓存的拓扑结构**

* 缓存分为私有缓存（只为一个客户端服务，比如我们给浏览器配置的代理）和共有缓存（为多个客户端服务，现实中是以代理缓存服务器的形式踹出现）。
* 代理缓存的层级结构：此种结构描述的以父、子层级出现的层次结构，同时离客户端越近的子缓存的命中率较低（较廉价），他们可以把请求上升到父缓存（较昂贵），从而在父缓存那里实现事务处理！
* *代理缓存的网状结构*描述的缓存结构并不是很明显呈现父子关系的结构，而是呈无规则的网状！这种结构的思想就是子缓存可以动态选择上一级缓存，从而实现更灵活的缓存控制！

**缓存的处理步骤**

* 简单概括，即：

接收————解析————查询————新鲜度检测————创建响应————发送————记入日志

* 第一步接收：读取网络连接http请求报文
* 第二部解析：把报文解析为片段，并把首部放入到缓存易操作的数据结构中
* 第三步查询：查找存下来缓存副本
* 第四步新鲜度检测：说白了检查缓存副本是不是还有效的
* 第五步创建响应：缓存服务器用原始服务器的缓存副本实现响应的起点，同时再在此基础上做一些修改，比如协议转换等
* 第六步发送：发送报文
* 第七步日志：事务完成之后，在日志文件插入一个条目，用以记录缓存处理情况，以及记录一些与缓存命中率的数据