Web 缓存与测试

1. **什么是Web缓存**

Web缓存是指一个Web资源（如html页面，图片，js，数据等）存在于Web服务器和客户端（浏览器）之间的副本。缓存会根据进来的请求保存输出内容的副本；当下一个请求来到的时候，如果是相同的URL，缓存会根据缓存机制决定是直接使用副本响应访问请求，还是向源服务器再次发送请求。比较常见的就是浏览器会缓存访问过网站的网页，当再次访问这个URL地址的时候，如果网页没有更新，就不会再次下载网页，而是直接使用本地缓存的网页。只有当网站明确标识资源已经更新，浏览器才会再次下载网页。

1. **缓存作用**
2. 减少网络带宽消耗，降低网络传输：当Web缓存副本被使用时，只会产生极小的网络流量，可有效降低运营成本
3. 降低服务器压力：给网络资源设定有效期后，用户可重复使用本地缓存，减少对源服务器的请求，间接降低服务器的压力。同时，搜索引擎的爬虫机器人也能根据过期机制降低爬取的频率，也能有效降低服务器的压力
4. 减少网络延迟、加快页面打开速度：缓存离客户端更近，从缓存请求内容比从源服务器所用时间更少，呈现速度更快，网站就显得更灵敏
5. **缓存类型**
6. 数据库数据缓存

Web应用中，特别是SNS类型的应用，往往关系比较复杂，数据库表繁多，如果频繁进行数据库查询，很容易导致数据库不堪重荷。为了提供查询的性能，会将查询后的数据放到内存中进行缓存，下次查询时，直接从内存缓存直接返回，提供响应效率。比如常用的缓存方案有memcached等。

1. 浏览器缓存

在任何浏览器上（如IE、FireFox、Chrome）打开清除隐私数据的对话框，基本上都有“缓存”设置项，浏览器会在你的硬盘上专门开辟一个空间专门存储资源副本。

浏览器缓存根据一套与服务器约定的规则进行工作，在同一个会话过程中会检查一次并确定缓存的副本足够新。如果在浏览过程中，比如前进或后退，访问到同一个图片，这些图片可以从浏览器缓存中调出而即时显现。

1. 服务器端缓存：代理服务器缓存

代理服务器是浏览器和源服务器之间的中间服务器，浏览器先向这个中间服务器发起Web请求，经过处理后（比如权限验证，缓存匹配等），再将请求转发到源服务器。代理服务器缓存的运作原理跟浏览器的运作原理差不多，只是规模更大。可以把它理解为一个共享缓存，不只为一个用户服务，一般为大量用户提供服务，因此在减少相应时间和带宽使用方面很有效，同一个副本会被重用多次。常见代理服务器缓存解决方案有Squid等

由于代理服务器缓存并非客户端或者源服务器的一部分，而是处于网络中，请求需要以某种方式路由到它们。一种方法是手动设置，告诉浏览器的你常用的代理服务器(//zxx: 翻墙的时候常用的)，另外就是使用拦截。拦截代理(Interception proxies)把Web请求根据自己的底层网络重定向，因此，客户端无需配置，甚至都不需要知道它们。//zxx: 维基百科上提供的几种检测拦截代理服务器存在的方法

代理缓存属于一种共享缓存；往往有大量的用户使用，因此，其在降低延时和网络流量上很有用，毕竟每个副本都被大量重用。

1. CDN缓存

CDN（Content delivery networks）缓存，也叫网关缓存、反向代理缓存、替代缓存。CDN缓存一般是由网站管理员自己部署，为了让他们的网站更容易扩展并获得更好的性能。浏览器先向CDN网关发起Web请求，网关服务器后面对应着一台或多台负载均衡源服务器，会根据它们的负载请求，动态将请求转发到合适的源服务器上。虽然这种架构负载均衡源服务器之间的缓存没法共享，但却拥有更好的处扩展性。从浏览器角度来看，整个CDN就是一个源服务器，从这个层面来说，本文讨论浏览器和服务器之间的缓存机制，在这种架构下同样适用。

①负载均衡器：是一种采用各种分配算法把网络请求分散到一个服务器集群中的可用服务器上去，通过管理进入的Web数据流量和增加有效的网络带宽，从而使网络访问者获得尽可能最佳的联网体验的硬件设备。

②内容分发网络：即CDN, 基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容，解决 Internet网络拥挤的状况，提高用户访问网站的响应速度。

1. Web应用层缓存

应用层缓存指的是从代码层面上，通过代码逻辑和缓存策略，实现对数据，页面，图片等资源的缓存，可以根据实际情况选择将数据存在文件系统或者内存中，减少数据库查询或者读写瓶颈，提高响应效率。

1. **Web缓存工作机制**

部分规则设定在协议中（HTTP1.0以及1.1），一般而言有如下常用规则：

1. 响应头明确说明，不想被缓存的，则不会被缓存
2. 如果请求信息是需要认真或安全加密得（如，HTTPS），相应内容也不会被缓存
3. 缓存如果存在以下表现，则认为是新鲜（fresh）的（无需检查源服务器，直接发送给客户端），则内容缓存直取，绕过源服务器
4. 含有完整的过期的时间和寿命控制头信息，并内容仍在保险期内，或者缓存最近已展现，并且在不久前修改
5. 若内容陈旧，则会要求源服务器做验证validate，或告诉缓存其拷贝的副本是否是OK的
6. 特定情况下，如断网了，之前有过的响应缓存直取而不检查源服务器

响应如果没有类似*ETag*或*Last-Modified*头这样的校验器，也没有明确的更新信息，通常（并不绝对）认为是不可缓存的。

总而言之，**新鲜度***freshness*和**校验***validation*是确定缓存内容是否可用的最重要途径。如果要展示的足够新，直接缓存取；如果检测发现展示内容并未变化，则不会再来一次完整的传输。

1. **缓存测试**