**概述**

  Web缓存可以自动保存常见文档副本的HTTP设备。当Web请求抵达缓存时，如果本地有“已缓存的”副本，就从本地存储设备而不是原始服务器中提取这个文档。

使用缓存的优点

1. 减少了冗余的数据传输
2. 缓解了网络的瓶颈的问题（不需更多的带宽就能更快加载）
3. 降低了对原始服务器的请求
4. 降低了距离时延

**一、冗余的数据传输**

  1.**多名用户访问原始服务器页面时，服务器会多次传输同一份文档，相同的字节会在网络中一遍遍地传输**。有了缓存，就可以保留第一条服务器响应的副本，后继请求就可以由缓存的副本来应对了。

  2.因为缓存放在了离计算机更近的机房，可以让客户端更快得到一份副本。

**“命中和未命中”**

  缓存命中（cache hit）：可以用已有的副本为某些到达缓存的请求提供服务。

  缓存未命中（cache miss）：没有副本可用，而被转发到原始服务器。

**二、再验证**

  原始服务器的内容可能会发生变化，缓存要不时的对其进行检测，看看保存的副本是否为最新的。“新鲜度检测”被称为HTTP再验证（revalidation）。**为了有效的验证，HTTP定义了一些特殊的请求，不要用从服务器获取整个对象，就可以快速检测出内容是否为最新。**

  由于缓存中通常会包含数百万文档，而且网络宽带比较珍贵，所以大部分缓存只有客户端发起请求，并且副本旧的足以需要检测时，才会再验证。

  缓存对缓存的副本进行再验证时，会向服务器发送一个小的再验证请求。若缓存副本仍然有效，副本就会被标志成是新鲜的，并将副本提供给客户端，这称为再验证命中（revalidate hit）。通常会用**If-Modified-Since**首部来验证，把它添加到GET请求中去，就可以告诉服务器。

下面列出响应报文中可能出现的三种情况（**内容未修改、内容被修改、内容被删除**）：

* 再验证命中

如果服务器对象未被修改，服务器会向客户端发送一个小的HTTP 304 Not Modified响应。

* 再验证未命中

如果服务器对象与已缓存副本不同，服务器向客户端发送一条普通的，带有完整内容的HTTP 200 OK响应。

* 对象被删除

如果服务器对象已经被删除了，服务器就会送一个 404 Not Found响应，缓存也会将其副本删除。

**三、区分命中和未命中的情况**

  HTTP没有为用户提供一种手段来区分响应是缓存命中的，还是访问原始服务器得到的。这两种情况返回的都是200 OK

  客户端有一种方法能判断响应是否来自缓存，就是使用Date首部。将响应中Date首部的值与当前时间进行比较，如果响应中的日期值比较早，客户端通常就可以认为是来自缓存的；客户端也可以通过Age首部来检测缓存的响应，通过这个首部可以分辨出这条响应的使用期。否则，则认为是来自原始服务器中的。

**两种缓存**

**私有缓存**

Web浏览器中有内建的私有缓存，大多数浏览器会将常用文档缓存在你个人电脑的磁盘和内存中，并且允许用户去配置缓存的大小和各种设置。还可以看浏览器的缓存中有什么内容。

     私有缓存不需要很大的动力或存储空间。

**公有代理缓存**

公有代理缓存是特殊的共享代理服务器，被称为缓存代理服务器（caching proxy server），或称为代理缓存（proxy cache）。代理缓存会从本地缓存中提取文档，或者代表用户与服务器进行联系。公有缓存会接受来自多个用户的访问，所以通过他可以更好地减少冗余流量。

**缓存的处理步骤**

对一条HTTP GET报文的基本缓存处理过程包括7个步骤

1. 接受——缓存从网络中读取抵达的请求报文。
2. 解析——缓存对报文进行解析，提取出URL和各种首部。
3. 查询——缓存查看是否有本地副本可用，如果没有，就获取一份副本并保存在本地。
4. 新鲜度检测——缓存查看已缓存副本是否足够新鲜，如果不是，就询问服务器是否有任何更新。
5. 创建响应——缓存会用新的首部和已缓存的主体来构建一条响应报文。
6. 发送——缓存通过网络将响应发回给客户端。
7. 日志——缓存可选择创建一个日志文件条目来描述这个事务。

**四、保持副本的新鲜**

**文档过期**

通过特殊的HTTP Cache-Control和Expires首部，HTTP让原始服务器向每个文档附加一个“过期日期”。

**[plain]** [view plaincopy](http://blog.csdn.net/sunorry/article/details/8499291)

1. Expires：Fri, 05, Jan 2012, 01:31:39 GMT
2. Cache-Control: max-age=439293

**Expires与Cache-Control**

**Expries：**指定一个绝对过期日期。如果过期日期已经过了，就说明文档不再新鲜了。

Expires首部和Cache-Control:max-age首部所做的事情本质上是一样的，但是Cache-Control首部使用的是相对时间而不是绝对日期（Expires使用绝对的过期日期），所以**我们更倾向于使用比较新的Cache-Control首部，绝对日期依赖于计算机时钟的正确设置。**

**Cache-Control：**max-age：max-age值定义了文档的最大使用期——从第一次生成文档到文档不再新鲜，无法使用为止，最大的合法生存时间（以秒为单位）。

**服务器再验证**

仅仅是已缓存文档过期了并不意味着它和原始服务器上目前处理活跃状态的文档有实际区别，这只是意味着到了要进行审核的时间了。这种情况被称为“服务器再验证”，说明缓存需要询问服务器文档是否发生了变化。

1）如果内容发生变化，缓存会获取一份新的文档副本，并将其存储在旧文档的位置上，然后将文档发送给客户端。

**2）如果内容没有变化，缓存只需要获取新的首部，包括一个新的过期日期，并对缓存中的首部进行更新就行了。**

**用条件方法进行再验证**

对于缓存再验证来说最有用的2个首部是：

**1、(If-Modified-Since、If-Unmodified-Since）[Last-Modified]**

**2、(If-None-Match、If-Match）[ETag]**

**If-Modified-Since：**如果从指定日期之后文档被修改过了，就执行请求的方法。可以与Last-Modified服务器响应首部配合使用，只有在内容被修改后与已缓存版本有所不同的时候才去获取内容。

**If-None-Match**：服务器可以为文档提供特殊的标签（**Etag**，版本标识符），而不是将其与最近修改日期相匹配，这些标签就像序列号一样。如果已缓存标签与服务器文档中标签有所不同，If-None-Match首部就会执行所请求的方法。

**服务端与客户端控制缓存的能力**

可以通过HTTP定义的几种方式来指定文档的过期策略

Cache-Control:no-store（客户端不要存储缓存数据，当服务器不希望客户端浏览器缓存到临时文件时可以使用）

Cache-Control:no-cache（客户端不要存储缓存数据，但客户端仍要自己存储也是OK的，服务器是否有更新客户端看着办）

Cache-Control:max-age（缓存的内容将在 xxx 秒后失效）

Cache-Control是关于浏览器缓存的最重要的设置，因为它覆盖其他设置，比如 Expires 和 Last-Modified。