<http://www.alisdn.com/wordpress/?cat=8> 阿里软件前端开发

最近，[YouMonitor.Us](http://youmonitor.us/)在做Web应用性能优化，在网上发现了文章[High Performance Web Sites: The Importance of Front-End Performance](http://developer.yahoo.com/performance/rules.html)，感觉其14条优化法则很实用，操作性很强。因此翻译出来，供大家参考。

**Web应用性能优化黄金法则：先优化前端程序(front-end)的性能，因为这是80%或以上的最终用户响应时间的花费所在。**

## 法则1. 减少HTTP请求次数

80%的最终用户响应时间花在前端程序上，而其大部分时间则花在各种页面元素，如图像、样式表、脚本和Flash等，的下载上。减少页面元素将会减少HTTP请求次数。这是快速显示页面的关键所在。

一种减少页面元素个数的方法是简化页面设计。但是否存在其他方式，能做到既有丰富内容，又能获得快速响应时间呢？以下是这样一些技术：

[Image maps](http://www.w3.org/TR/html401/struct/objects.html#h-13.6)组合多个图片到一张图片中。总文件大小变化不大，但减少了HTTP请求次数从而加快了页面显示速度。该方式只适合图片连续的情况；同时坐标的定义是烦人又容易出错的工作。

[CSS Sprites](http://alistapart.com/articles/sprites)是更好的方法。它可以组合页面中的图片到单个文件中，并使用CSS的background-image和background-position属性来现实所需的部分图片。

Inline images使用[data: URL scheme](http://tools.ietf.org/html/rfc2397)来在页面中内嵌图片。这将增大HTML文件的大小。组合inline images到你的（缓存）样式表是既能较少HTTP请求，又能避免加大HTML文件大小的方法。

Combined files通过组合多个脚本文件到单一文件来减少HTTP请求次数。样式表也可采用类似方法处理。这个方法虽然简单，但没有得到大规模的使用。10大美国网站每页平均有7个脚本文件和2个样式表。当页面之间脚本和样式表变化很大时，该方式将遇到很大的挑战，但如果做到的话，将能加快响应时间。

减少HTTP请求次数是性能优化的起点。这最提高首次访问的效率起到很重要的作用。据Tenni Theurer的文章[Browser Cache Usage - Exposed!](http://yuiblog.com/blog/2007/01/04/performance-research-part-2/)描述，40-60%的日常访问是首次访问，因此为首次访问者加快页面访问速度是用户体验的关键。

**我们的应用:**

外贸: 将首页的几十个小图标合并为一个，通过CSS控制它们的显示，减少了HTTP请求数。

## 法则2. 使用CDN(Content Delivery Network, 内容分发网络)

用户离web server的远近对响应时间也有很大影响。从用户角度看，把内容部署到多个地理位置分散的服务器上将有效提高页面装载速度。但是该从哪里开始呢？

作为实现内容地理分布的第一步，不要试图重构web应用以适应分布架构。改变架构将导致多个周期性任务，如同步session状态，在多个server之间复制数据库交易。这样缩短用户与内容距离的尝试可能被应用架构改版所延迟，或阻止。

我们还记得80-90%的最终用户响应时间花在下载页面中的各种元素上，如图像文件、样式表、脚本和Flash等。与其花在重构系统这个困难的任务上，还不如先分布静态内容。这不仅能大大减少响应时间，而且由于CDN的存在，分布静态内容非常容易实现。

CDN是地理上分布的web server的集合，用于更高效地发布内容。通常基于网络远近来选择给具体用户服务的web server。

一些大型网站拥有自己的CDN，但是使用如[Akamai Technologies](http://www.akamai.com/), [Mirror Image Internet](http://www.mirror-image.com/), 或 [Limelight Networks](http://www.limelightnetworks.com/)等CDN服务提供商的服务将是划算的。在Yahoo!把静态内容分布到CDN减少了用户影响时间20%或更多。切换到CDN的代码修改工作是很容易的，但能达到提高网站的速度。

**我们的应用：**

目前还没用到，不过据客户反映，广东山东等地网络情况比较差，如果可以将占据主要带宽的静态资源通过CDN发布，相信可以大大缓解目前网站访问速度的问题。

## 法则3. 增加Expires Header

网页内容正变得越来越丰富，这意味着更多的脚本文件、样式表、图像文件和Flash。首次访问者将不得不面临多次HTTP请求，但通过使用Expires header，您可以在客户端缓存这些元素。这在后续访问中避免了不必要的HTTP请求。Expires header最常用于图像文件，但是它也应该用于脚本文件、样式表和Flash。

浏览器（和代理）使用缓存来减少HTTP请求的次数和大小，使得网页加速装载。Web server通过Expires header告诉客户端一个元素可以缓存的时间长度。

如果服务器是Apache的话，您可以使用ExpiresDefault基于当期日期来设置过期日期，如：

ExpiresDefault “access plus 10 years” 设置过期时间为从请求时间开始计算的10年。

请记住，如果使用超长的过期时间，则当内容改变时，您必须修改文件名称。在Yahoo!我们经常把改名作为release的一个步骤：版本号内嵌在文件名中，如yahoo\_2.0.6.js。

**我们的应用：**

外贸：在Apache配置了JS,CSS,image的缓存，如果静态资源需要更新，则采用修改文件版本号的方案确保客户端取得最新版本；

E网打尽：E网打尽的探头规则（JS）是根据客户的设置生成的，但是在相当长的一段时间内基本上不会有变化，因此，在生成规则的同时附加一个expires响应头，尽量减少客户端的请求和探头规则生成的次数。

## 法则4. 压缩页面元素

通过压缩HTTP响应内容可减少页面响应时间。从HTTP/1.1开始，web客户端在HTTP请求中通过Accept-Encoding头来表明支持的压缩类型，如：

Accept-Encoding: gzip, deflate.

如果Web server检查到Accept-Encoding头，它会使用客户端支持的方法来压缩HTTP响应，会设置Content-Encoding头，如：Content-Encoding: gzip。

Gzip是目前最流行及有效的压缩方法。其他的方式如deflate，但它效果较差，也不够流行。通过Gzip，内容一般可减少70%。如果是Apache，在1.3版本下需使用[mod\_gzip](http://sourceforge.net/projects/mod-gzip/)模块，而在2.x版本下，则需使用[mod\_deflate](http://httpd.apache.org/docs/2.0/mod/mod_deflate.html)。

Web server根据文件类型来决定是否压缩。大部分网站对HTML文件进行压缩。但对脚本文件和样式表进行压缩也是值得的。实际上，对包括XML和JSON在内的任务文本信息进行压缩都是值得的。图像文件和PDF文件不应该被压缩，因为它们本来就是压缩格式保存的。对它们进行压缩，不但浪费CPU，而且还可能增加文件的大小。

因此，对尽量多的文件类型进行压缩是一种减少页面大小和提高用户体验的简便方法。

**我们的应用：**

外贸、E网打尽、K计划：600多K的ext2包，是人都会想到要去压缩它，压缩后的效果还不错，只有150多K。另外，JS、CSS、HTML也尽量压缩，要知道我们的很多客户还在使用1M的ADSL。

## 法则5. 把样式表放在头上

我们发现把样式表移到HEAD部分可以提高界面加载速度，因此这使得页面元素可以顺序显示。

在很多浏览器下，如IE，把样式表放在document的底部的问题在于它禁止了网页内容的顺序显示。浏览器阻止显示以免重画页面元素，那用户只能看到空白页了。Firefox不会阻止显示，但这意味着当样式表下载后，有些页面元素可能需要重画，这导致闪烁问题。

[HTML规范](http://www.w3.org/TR/html4/struct/links.html#h-12.3)明确要求样式表被定义在HEAD中，因此，为避免空白屏幕或闪烁问题，最好的办法是遵循HTML规范，把样式表放在HEAD中。

**我们的应用：**

目前还没有碰到把样式表放在文档后面的情况吧？

## 法则6. 把脚本文件放在底部

与样式文件一样，我们需要注意脚本文件的位置。我们需尽量把它们放在页面的底部，这样一方面能顺序显示，另方面可达到最大的并行下载。

浏览器会阻塞显示直到样式表下载完毕，因此我们需要把样式表放在HEAD部分。而对于脚本来说，脚本后面内容的顺序显示将被阻塞，因此把脚本尽量放在底部意味着更多内容能被快速显示。

脚本引起的第二个问题是它阻塞并行下载数量。[HTTP/1.1规范](http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.4)建议浏览器每个主机的并行下载数不超过2个。因此如果您把图像文件分布到多台机器的话，您可以达到超过2个的并行下载。但是当脚本文件下载时，浏览器不会启动其他的并行下载，甚至其他主机的下载也不启动。

在某些情况下，不是很容易就能把脚本移到底部的。如，脚本使用document.write方法来插入页面内容。同时可能还存在域的问题。不过在很多情况下，还是有一些方法的。

一个备选方法是使用延迟脚本（deferred script）。DEFER属性表明脚本未包含document.write，指示浏览器刻继续显示。不幸的是，Firefox不支持DEFER属性。在IE中，脚本可能被延迟执行，但不一定得到需要的长时间延迟。不过从另外角度来说，如果脚本能被延迟执行，那它就可以被放在底部了。

**我们的应用：**

这点之前大家可能都没有意识到，不过在我们的XCube XUI中我们已经实施了这条法则，相信可以进一步提升页面的访问性能。

**法则7. 避免CSS表达式**

CSS表达式是功能强大的(同时也是危险的)用于动态设置CSS属性的方式。IE，从版本5开始支持CSS表达式，如backgourd-color: expression((new Date()).getHours()%2?”#B8D4FF”:”#F08A00”)，即背景色每个小时切换一次。

CSS表达式的问题是其执行次数超过大部分人的期望。不仅页面显示和resize时计算表达式，而且当页面滚屏，甚至当鼠标在页面上移动时都会重新计算表达式。

一种减少CSS表达式执行次数的方法是一次性表达式，即当第一次执行时就以明确的数值代替表达式。如果必须动态设置的话，可使用事件处理函数代替。如果您必须使用CSS表达式的话，请记住它们可能被执行上千次，从而影响页面性能。

**我们的应用：**

目前CSS的维护工作主要由UI人员负责，他们已经尽量在避免这种情况了。

## 法则8. 把JavaScript和CSS放到外部文件中

上述很多性能优化法则都基于外部文件进行优化。现在，我们必须问一个问题：JavaScript和CSS应该包括在外部文件，还是在页面文件中？

在现实世界中，使用外部文件会加快页面显示速度，因为外部文件会被浏览器缓存。如果内置JavaScript和CSS在页面中虽然会减少HTTP请求次数，但增大了页面的大小。另外一方面，使用外部文件，会被浏览器缓存，则页面大小会减小，同时又不增加HTTP请求次数。

因此，一般来说，外部文件是更可行的方式。唯一的例外是内嵌方式对主页更有效，如[Yahoo!](http://www.yahoo.com/)和[My Yahoo!](http://my.yahoo.com/)都使用内嵌方式。一般来说，在一个session中，主页访问此时较少，因此内嵌方式可以取得更快的用户响应时间。

**我们的应用：**

外贸、E网打尽、K计划：ext2的代码作了很好的引导，目前前端开发人员都非常注意客户端模块的封装、重用，尽量以外部JS的方式提高代码的重用度，当然也要注意不要引入过多的外部资源，因为这违反了法则1。  
目前CSS的封装也不错，但是主要是针对IE系列的解决方案，可以考虑引入YAML、blueprint等CSS框架，轻松解决浏览器兼容性问题。

**法则9. 减少DNS查询次数**

DNS用于映射主机名和IP地址，一般一次解析需要20～120毫秒。为达到更高的性能，DNS解析通常被多级别地缓存，如由ISP或局域网维护的caching server，本地机器操作系统的缓存（如windows上的DNS Client Service），浏览器。IE的缺省DNS缓存时间为30分钟，Firefox的缺省缓冲时间是1分钟。

减少主机名可减少DNS查询的次数，但可能造成并行下载数的减少。避免DNS查询可减少响应时间，而减少并行下载数可能增加响应时间。一个可行的折中是把内容分布到至少2个，最多4个不同的主机名上。

**我们的应用：**

外贸：为了绕开浏览器对下载线程数的限制，我们对静态资源启用了多域名，但是这么做违反了该法则。不过，对windows IE来说，DNS的缓存可以缓解该问题。

## 法则10. 最小化JavaScript代码

最小化JavaScript代码指在JS代码中删除不必要的字符，从而降低下载时间。两个流行的工具是[JSMin](http://crockford.com/javascript/jsmin) 和[YUI Compressor](http://developer.yahoo.com/yui/compressor/)。

混淆是最小化于源码的备选方式。象最小化一样，它通过删除注释和空格来减少源码大小，同时它还可以对代码进行混淆处理。作为混淆的一部分，函数名和变量名被替换成短的字符串，这使得代码更紧凑，同时也更难读，使得难于被反向工程。Dojo Compressor ([ShrinkSafe](http://dojotoolkit.org/docs/shrinksafe))是最常见的混淆工具。

最小化是安全的、直白的过程，而混淆则更复杂，而且容易产生问题。从对美国10大网站的调查来看，通过最小化，文件可减少21%，而混淆则可减少25%。

除了最小化外部脚本文件外，内嵌的脚本代码也应该被最小化。即使脚本根据法则4被压缩后传输，最小化脚本刻减少文件大小5%或更高。

**我们的应用：**

我们没有直接使用JS压缩，但是我们用的许多组件例如ext2、jquery等，已经在为我们实践该法则。

## 法则11. 避免重定向

重定向功能是通过301和302这两个HTTP状态码完成的，如：

      HTTP/1.1 301 Moved Permanently

      Location: http://example.com/newuri

      Content-Type: text/html

浏览器自动重定向请求到Location指定的URL上，重定向的主要问题是降低了用户体验。

一种最耗费资源、经常发生而很容易被忽视的重定向是URL的最后缺少/，如访问<http://astrology.yahoo.com/astrology>将被重定向到<http://astrology.yahoo.com/astrology/>。在Apache下，可以通过Alias，mod\_rewrite或DirectorySlash等方式来解决该问题。

**我们的应用：**

经验丰富的SA已经为我们考虑了这个问题，有兴趣的同学可以看看线上环境的Apache配置文件：httpd.conf。

**法则12. 删除重复的脚本文件**

在一个页面中包含重复的JS脚本文件会影响性能，即它会建立不必要的HTTP请求和额外的JS执行。

不必要的HTTP请求发生在IE下，而Firefox不会产生多余的HTTP请求。额外的JS执行，不管在IE下，还是在Firefox下，都会发生。

一个避免重复的脚本文件的方式是使用模板系统来建立脚本管理模块。除了防止重复的脚本文件外，该模块还可以实现依赖性检查和增加版本号到脚本文件名中，从而实现超长的过期时间。

**我们的应用：**

旧版本的Xplatform中这个问题比较严重，不过相信新版的XCube不会重蹈覆辙。

## 法则13. 配置ETags

ETags是用于确定浏览器缓存中元素是否与Web server中的元素相匹配的机制，它是比last-modified date更灵活的元素验证机制。ETag是用于唯一表示元素版本的字符串，它需被包括在引号中。Web server首先在response中指定ETag：

      HTTP/1.1 200 OK

      Last-Modified: Tue, 12 Dec 2006 03:03:59 GMT

      ETag: "10c24bc-4ab-457e1c1f"

      Content-Length: 12195

后来，如果浏览器需要验证某元素，它使用If-None-Match头回传ETag给Web server，如果ETag匹配，则服务器返回304代码，从而节省了下载时间：

      GET /i/yahoo.gif HTTP/1.1

      Host: us.yimg.com

      If-Modified-Since: Tue, 12 Dec 2006 03:03:59 GMT

      If-None-Match: "10c24bc-4ab-457e1c1f"

      HTTP/1.1 304 Not Modified

ETags的问题在于它们是基于服务器唯一性的某些属性构造的，如Apache1.3和2.x，其格式是inode-size-timestamp，而在IIS5.0和6.0下，其格式是Filetimestamp:ChangeNumber。这样同一个元素在不同的web server上，其ETag是不一样的。这样在多Web server的环境下，浏览器先从server1请求某元素，后来向server2验证该元素，由于ETag不同，所以缓存失效，必须重新下载。

因此，如果您未用到ETags系统提供的灵活的验证机制，最好删除ETag。删除ETag会减少http response及后续请求的HTTP头的大小。[微软支持文章](http://support.microsoft.com/?id=922733)描述了如何删除ETags，而在Apache下，只要在配置文件中设置FileETag none即可。

**我们的应用：**

E网打尽：自定义ETag的生成策略，以尽量减少探头规则的生成次数。由于不是采用服务器默认的ETag，不存在该问题。

其他产品线：要注意了，这点大家都没有关注过吧，赶快检查一下Apache中的配置。

**法则14. 缓存Ajax**

性能优化法则同样适用于web 2.0应用。提高Ajax的性能最重要的方式是使得其response可缓存，就象“法则3增加Expires Header”讨论的那样。以下其他法则同样适用于Ajax，当然法则3是最有效的方式:

法则4. 压缩页面元素

法则9. 减少DNS查询次数

法则10. 最小化脚本文件

法则11. 避免重定向

法则13. 配置ETags.

**我们的应用：**

更多情况下，我们倒不希望Ajax请求被缓存，此时为每个Ajax请求的url附加一个时间戳就可以了。