链滴

Http 缓存 1：基本工作原理

作者：Lord-X

原文链接：https://ld246.com/article/1587902353875

来源网站：链滴

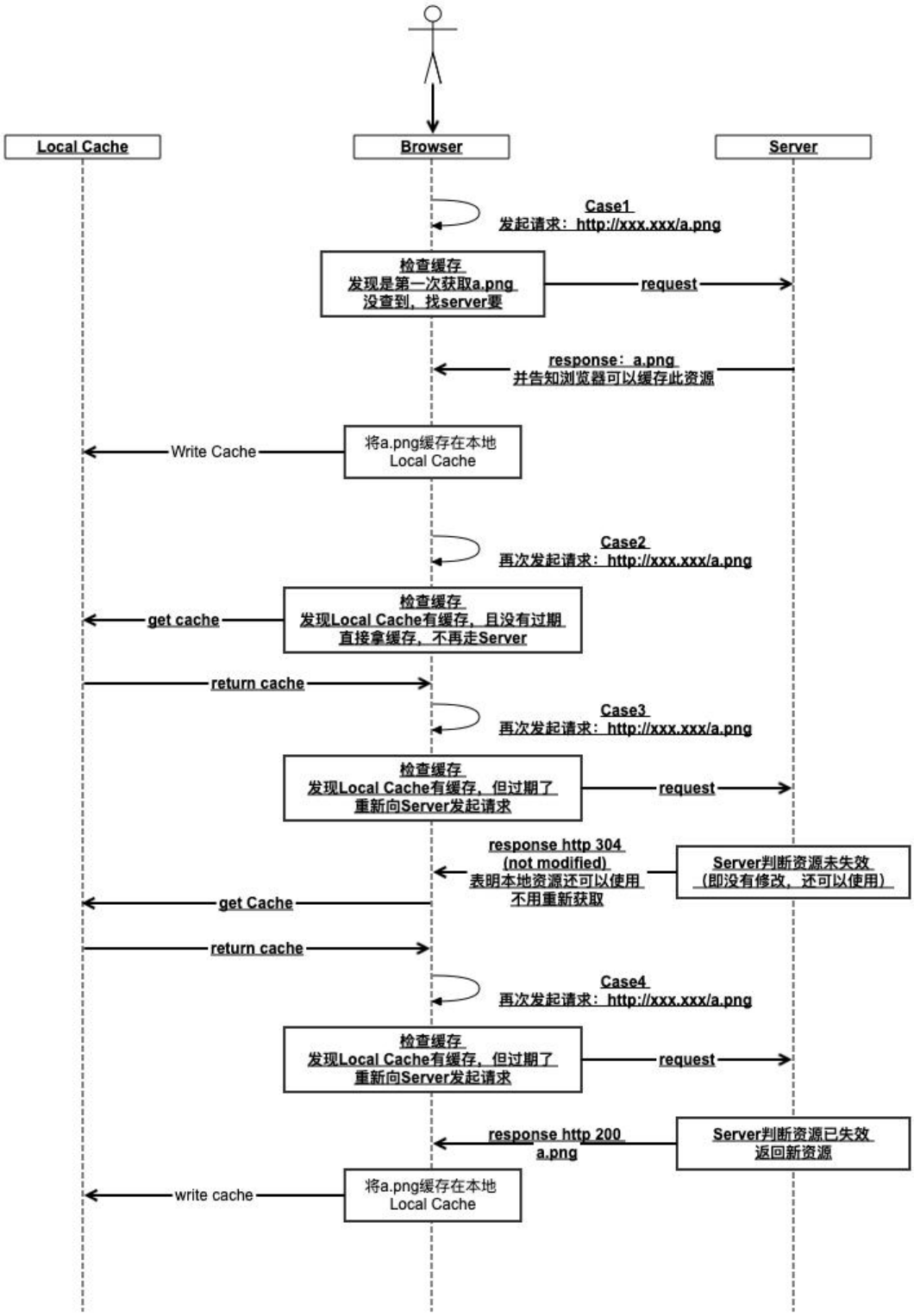
Http缓存：基本工作原理

Http缓存是解决http1.1协议性能问题的一个主要手段。缓存即有可能存在于浏览器中，也可能存在服务器中。

Http缓存的主要原理是为当前请求复用之前请求的响应，其主要目标是降低时延、降低带宽的消耗。

Http缓存的基本交互流程

下图以4个Case说明Browser向Server获取a.png这个资源的几种Cache交互场景。



- Case1

第一次访问，没有本地缓存，向Server发起请求，Server返回200并将a.png返回给Browser，Browse将a.png缓存在本地。

- Case2

第二次访问，本地已有a.png的缓存，不再向Server发起请求，直接使用本地缓存。

原文链接：Http 缓存 1：基本工作原理

● Case3

第三次访问，本地有a.png缓存，但本地已过期，向Server发起请求，同时会带上本地a.png的指纹此时Server会检查Server端的a.png的指纹是否和Browser传过来的相同，如果相同，说明Server端有更改，Server端会返回http304，但并不会返回a.png给Browser，让Browser继续使用本地的a.pn即可。

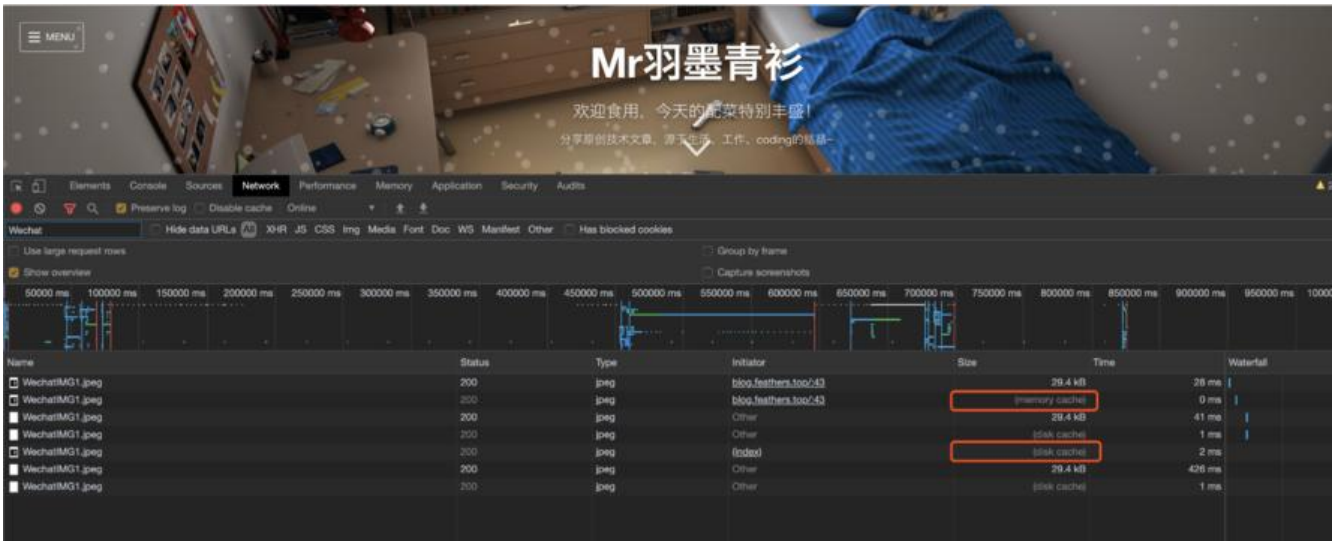
● Case4

第四次访问，本地有a.png缓存，但本地已过期，向Server发起请求，同时带上本地a.png的指纹。时Server检查Server端的a.png指纹是否与Browser传过来的相同，如果不同，说明Server端有更新，erver端会返回http200，并返回新的a.png给Browser，Browser使用新的a.png并将本地cache更新。

举个栗子

例如我们模拟向我的blog站点访问 http://blog.feathers.top/ ，并观察 http://image.feathers.top/mage/WechatIMG1.jpeg 这个资源，为了保证第一次不使用本地缓存，用 command + shift + R 制从Server获取。第一次访问如下所示：

这里看到Size这个字段是这个资源实际的大小，表示没有使用缓存，接下来刷新页面，使用缓存，注这个字段的变化：



这里显示了使用缓存的情况，有两种情况，Memory Cache和Disk Cache。

如果间隔时间较短刷新页面，再次请求时，因为此时内存中的缓存还没失效，会使用Memory Cache；

如果内存中的Cache存在时间过长，Browser会将Cache缓存在Disk中，此时再访问，会使用Disk Cac

e。

然后我们模拟一下Browser缓存失效的场景。默认情况下，缓存失效时间过长，为了模拟，我们将 WechatIMG1.jpeg 的请求cURL复制出来：

```
curl 'http://image.feathers.top/image/WechatIMG1.jpeg' \
  -H 'Referer: http://blog.feathers.top/' \
  -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.122 Safari/537.36' \
  --compressed \
  --insecure -I
```

在结尾加上 -I，使cURL只显示响应头。其响应结果如下：

```
HTTP/1.1 200 OK
Server: Tengine
Content-Type: image/jpeg
Content-Length: 28340
Connection: keep-alive
Date: Fri, 03 Apr 2020 22:16:19 GMT
Cache-Control: public, max-age=31536000
Etag: "FvMIFM00fGiTlUMzlREabTaxjkVS"
X-M-Log: QNM:xs450;QNM3/304
X-M-Reqid: MFUAAO5R_b0abwIW
X-Qnm-Cache: Hit
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: X-Log, X-Reqid
Access-Control-Max-Age: 2592000
Content-Disposition: inline; filename="WechatIMG1.jpeg"; filename*=utf-8''WechatIMG1.jpeg
Content-Transfer-Encoding: binary
Last-Modified: Thu, 19 Sep 2019 10:17:01 GMT
X-Log: X-Log
X-Qiniu-Zone: 0
X-Reqid: oM8AAADUIuzZT_IV
X-Svr: IO
Ali-Swift-Global-Savetime: 1583266120
Via: cache10.l2cn2179[0,200-0,H], cache20.l2cn2179[1,0], vcache15.cn1005[0,200-0,H], vcache.cn1005[2,0]
Age: 1943115
X-Cache: HIT TCP_HIT dirn:10:343300469
X-Swift-SaveTime: Thu, 23 Apr 2020 03:14:44 GMT
X-Swift-CacheTime: 2592000
Timing-Allow-Origin: *
EagleId: b7cb451c15878952940375540e
```

可以看到，max-age 是个很大的值，那如何模拟Browser端缓存失效的场景呢？其实，Browser在认缓存失效时，会在请求头加上一个 If-None-Match 头，他的值就是对应资源的标签（或者说指纹，上面的Etag字段），这样Server就可以拿这个值和自己的比对，如果相同，说明没有更改，就直接响一个304，告诉客户端继续用自己的就行了。下面我们加上这个Header模拟一下。

```
curl 'http://image.feathers.top/image/WechatIMG1.jpeg' \
  -H 'Referer: http://blog.feathers.top/' \
  -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.122 Safari/537.36' \
```

```
  -H 'If-None-Match: "FvMIFM00fGiTlUMzlREabTaxjkVS"' \
  --compressed \
  --insecure -I
```

结果如下：

```
HTTP/1.1 304 Not Modified
Server: Tengine
Content-Type: image/jpeg
Connection: keep-alive
Date: Thu, 02 Apr 2020 05:58:08 GMT
Cache-Control: max-age=4152551
Expires: Wed, 20 May 2020 07:27:19 GMT
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: X-Log, X-Reqid
Access-Control-Max-Age: 2592000
Content-Disposition: inline; filename="WechatIMG1.jpeg"; filename*=utf-8''WechatIMG1.jpeg
Content-Md5: aLibpZRSi2qRbdP2VjAb3g==
Content-Transfer-Encoding: binary
Etag: "FvMIFM00fGiTlUMzlREabTaxjkVS"
Last-Modified: Thu, 19 Sep 2019 10:17:01 GMT
X-Log: X-Log
X-M-Log: QNM:xs450;QNM3
X-M-Reqid: MFUAAIEPfg8l6wEW
X-Qiniu-Zone: 0
X-Qnm-Cache: Hit
X-Reqid: dA0AAACHK96wuAEW
X-Svr: IO
Ali-Swift-Global-Savetime: 1585807088
Via: cache77.l2cn2270[0,200-0,H], cache67.l2cn2270[80,0], vcache29.cn1015[0,304-0,H], vcach
10.cn1015[20,0]
Age: 2093585
X-Cache: HIT TCP_IMS_HIT dirn:0:506046938
Timing-Allow-Origin: *
EagleId: 6f06b41e15879006739354086e
```

第一行可以发现，Server直接返回了304，如果去掉 -I 参数，会发现它没有返回图片文件。

参考

- 极客时间 - Web协议详解与抓包实战
- RFC7234
- Cacheable
- Http Caching