Research and Implementation of HTTP Caching Freshness Algorithm

Tao Zhao

YunYang Teachers' College, Shiyan 442000, China

doitor@sina.com

Keywords: HTTP, cache, freshness, algorithm.

Abstract. Data caching technology can effectively reduce network congestion, alleviate server load and accelerate information access. Common mechanisms to keep cache documents fresh are document expiration and server revalidation. By calculating two values, age and freshness lifetime of the cached copy, you can determine whether the cached document is fresh enough. Based on the above mechanisms and methods, this paper has studied and implemented the HTTP cache freshness algorithm.

Introduction

Cache [1] is HTTP equipment which can automatically save a copy of common document. HTTP cache server is located between the client and the source server. When Web requests to arrive at the cache server, if there is "cached" copy at local, this document can be extracted from the local device. Using cache has the following advantages:

1. Reducing redundant data transmission.
2. Alleviating network bottleneck problem [2], able to load pages faster without more bandwidth.
3. Reducing the requirements on the source server, the server can respond quickly.

Treatment Scheme of Cache

Basic caching treating process [3] of a HTTP GET message contains seven steps.

1. Reception. Cache server reads the arrived request message from the network.
2. Parsing. Cache server parses the message, extracting the URL and each header.
3. Query. Cache server queries if there is available local copy, if not, it will get a copy from the source server and save it locally.
4. Freshness detection. Cache server checks if the cached copy is fresh enough, and if not, it will query the source server for updates.
5. Creating a response. Cache server uses the new header and the cached body to build a response message.
6. Sending. Cache server sends a response to the client.
7. Log. Cache server creates a log file to describe the transaction.

Mechanisms to Keep the Copy Fresh

HTTP has some mechanisms which can keep the cached data consistent with the data of source server without requiring the server to remember which cache has copy of document. These mechanisms are called document expiration and server revalidation.

Document Expiration. Server can use response header of Expires of HTTP / 1.0 + or Cache-Control: max-age [4] of HTTP / (Figure 1) to specify expiration date of the file.

Expires: specify an absolute expiration date.

Cache-control: max-age: define maximum period of use of the document, from when the document is generated the first time to when the document is no longer fresh until the document can't be used anymore, the maximum legal lifetime (with s as the unit).

```
HTTP/1.1 200 OK
Server: Tengine
Content-Type: image/jpeg
Content-Length: 26985
Connection: keep-alive
Date: Thu, 21 Aug 2014 04:03:50 GMT
Last-Modified: Thu, 21 Aug 2014 04:00:59 GMT
Expires: Sun, 18 Aug 2024 04:03:50 GMT
Cache-Control: max-age=315360000
Access-Control-Allow-Origin: *
Via: http/1.1 l2cn6 (ATS [cMsSfW]), cache1.cn109
Age: 1223132
X-Cache: HIT TCP_MEM_HIT dirn:1:1070920511
```

Fig. 1 Document expiration and server revalidation

Server Revalidation. If a cached document has expired, it does not mean there is a practical difference between it and other documents currently in the active state of the source server, so then the cache server needs to inquire whether the document of source server has changed. This condition is called "server revalidation".

Condition method of HTTP [5] can efficiently achieve revalidation. HTTP allows cache to send a "conditional GET" to the source server, requesting the server to send the object subject only when existing copies in the document and the cache are not the same.

HTTP defines five headers of condition request, and the most useful two headers for cache validation are If-Modified-Since and If-None-Match.

If-Modified-Since fields: It means the method that if the document is modified after the specified date, the request will be executed. It can be used cooperatively with Last-modified server response header, and it will get the content only when the modified content is different from the copy in cache.

If-None-Match fields: It means the method that server can provide document with special tag (ETag), and if the tag is not the same with tag of the server, the request will be executed.

If the server response comprises an ETag and also a Last-Mofidied value, two validation mechanisms will be used when the client is sending the request, and only when both the validation mechanisms are met, 304 Not Modified will be returned.

Tentative expiration. If there is no Expires header or Cache-control: max-age header in the source server response, the cache can calculate a tentative maximum trial period.

LM-Factor algorithm is a very commonly used algorithm for tentative expiration, which can be used if the document contains the last modification time. Figure 2 shows the calculated fresh cycle when the LM-Factor value is set as 0.1.

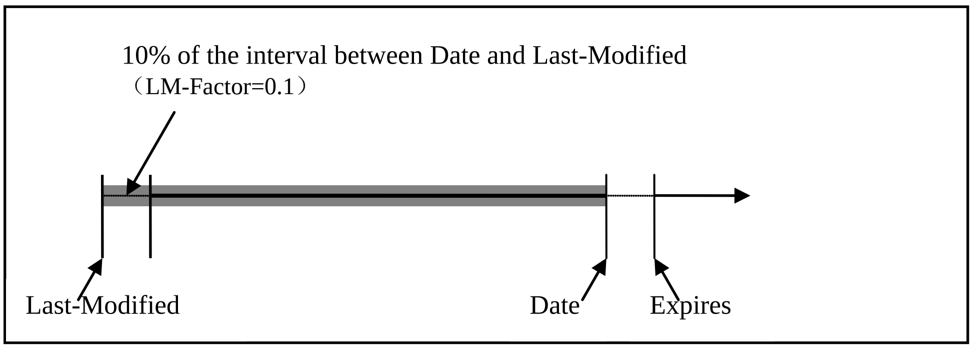

Fig. 2 Calculate fresh cycle with LM-Factor algorithm

LM-Factor algorithm can calculate the difference between the dialogue time of cache and the server (Date) and the last modification time of the document declared by the server, and take a part of the modified value as freshness duration of the cache. Here is PHP code of LM-Factor algorithm:

```
$ freshness_lifetime = (int) ($ factor * max (0, $ Last_Modified_value - $ Date_value));
```

Server Freshness Algorithm [6]

In order to identify whether the cached document is fresh enough, the cache only needs to calculate two values: age of the cached copy and freshness lifetime of the cached copy.

HTTP Caching Age Algorithm. Age of response is the total time passed after the server releases the response (or revalidation through validation model). Age includes the transmission time through the Internet, caching time in the intermediate nodes, and the standing time in the local cache.

```
apparent_age = max (0, response_time - date_value); // response age when cache receives the response
corrected_received_age = max (apparent_age, age_value); // tolerate error of Age header
response_delay = response_time-request_time; // deal with network delay
corrected_initial_age = corrected_received_age + response_delay;
resident_time = now-response_time; // local standing time, which is the time from reception of response to now
current_age = corrected_initial_age + resident_time;
```

Therefore, the complete age calculation algorithm is checking Date header and Age header to determine the time used by response, and then recording its standing time in the local cache which is the total age. In addition, HTTP protocol compensates clock skew and network delay. Particularly its compensation for network delay may repeatedly calculate the used time, so as to make the entire algorithm produce conservative results, make the document look older than the actual age and trigger revalidation [7].

HTTP Cache Freshness Algorithm. By using the age of cached document t[8], according to restrictions of the server and the client, the cache freshness lifetime can be calculated, which is shown as follows with PHP code:

Header fields related to this include (in descending priority): "max-age" control instruction value of Cache-Control fields, Expires, Last-Modified, default minimum lifetime.

```
function server_freshness_limit () {
global $ Max_Age_value_set, $ Max_Age_value;
global $ Expires_value_set, $ Expires_value;
global $ Date_value, $ default_cache_min_lifetime, $ default_cache_max_lifetime;
$ factor = 0.1; // typically set to 10%
$ heuristic = 0; //
if ($ Max_Age_value_set) {
$ freshness_lifetime = $ Max_Age_value;
} elseif ($ Expires_value_set) {
$ freshness_lifetime = $ Expires_value- $ Date_value;
} elseif ($ Last_Modified_value_set) {
$ freshness_lifetime = (int) ($ factor * max (0, $ Last_Modified_value- $ Date_value));
$ heuristic = 1;
} else {
$ freshness_lifetime = $ default_cache_min_lifetime;
$ heuristic = 1;
}
if ($ heuristic) {
$ freshness_lifetime = $ freshness_lifetime> $ default_cache_max_lifetime $ default_ cache_ max_ lifetime: $ freshness_lifetime;?
$ freshness_lifetime = $ freshness_lifetime <$ default_cache_min_lifetime $ default_ cache_ min_lifetime: $ freshness_lifetime;?
}
return $ freshness_lifetime;
}
```

In order to determine whether a response is fresh or stale, just compare freshness lifetime with age, shown as follows with PHP code:

```
response_is_fresh = (server_freshness_limit ()> current_age)
```

Conclusion

There is a set of rules to help all caches decide when to use copy in the cache to provide services. Among these rules, some are defined in protocol (such as HTTP / 1.0 + and HTTP / 1.1), while others are set by the administrator of cache (such as DBA, browser user, proxy server administrator or developer). For browser cache, these rules are defined in the message header of HTTP and Meta tags on HTML page. Based on freshness and check value, it can be determined whether the browser can directly use the copy in cache, or need get updated content from the source server. Studying freshness algorithm can help us better understand the mechanism of HTTP caching.

References

[1] Q. W. Shen. Technical analysis based on Internet caching. Journal (Natural Science Edition) of Hefei University of Technology, Vol. 03 (2002), p.451-454.

[2] Z. Zhou. Research and Implementation of WWW caching technology. Dalian Maritime University, 2004.

[3] X. S. Liu, Y. J. Qiu . Research and Implementation of HTTP caching. Small and Micro Computer System. 2000 (04).

[4] S. L. Tang. Research and Implementation of Web service caching technology based on Squid. Huazhong University of Science and Technology, 2004.

[5] Q. M. Zhang. Design and implementation of HTTP caching system. Southwest Jiaotong University, 2013.

[6] Y. W. Zhao. Applied research of caching mechanism in WWW. Wuhan University of Technology, 2006.

[7] G. Di. Study on HTTP implementing proxy server and cache replacement algorithm. Jilin University, 2010.

[8] Z. W. Tang. A cache replacement algorithm based on byte hit rate. Jinan University, 2012.