7.9.5 试探性过期

如果响应中没有 Cache-Control：max-age 首部，也没有 Expires 首部，缓存可以计算出一个试探性最大使用期。可以使用任意算法，但如果得到的最大使用期大于 24 小时，就应该向响应首部添加一个 Heuristic Expiration Warning（试探性过期警告，警告 13）首部。据我们所知，很少有浏览器会为用户提供这种警告信息。

LM-Factor 算法是一种很常用的试探性过期算法，如果文档中包含了最后修改日期，就可以使用这种算法。LM-Factor 算法将最后修改日期作为依据，来估计文档有多么易变。算法的逻辑如下所示。

- 如果已缓存文档最后一次修改发生在很久以前，它可能会是一份稳定的文档，不太会突然发生变化，因此将其继续保存在缓存中会比较安全。
- 如果已缓存文挡最近被修改过，就说明它很可能会频繁地发生变化，因此在与服务器进行再验证之前，只应该将其缓存很短一段时间。

实际的 LM-Factor 算法会计算缓存与服务器对话的时间跟服务器声明文档最后被修改的时间之间的差值，取这个间隔时间的一部分，将其作为缓存中的新鲜度持续时间。下面是 LM-factor 算法的 Perl 伪代码：

```
$time_since_modify = max(0, $server_Date - $server_Last_Modified);
$server_freshness_limit = int($time_since_modify * $lm_factor);
```

图 7-16 以图形方式给出了 LM-factor 的新鲜周期。图中用交叉线画出的阴影表示的是将 LM-factor 设置为 0.2 计算出的的新鲜周期。

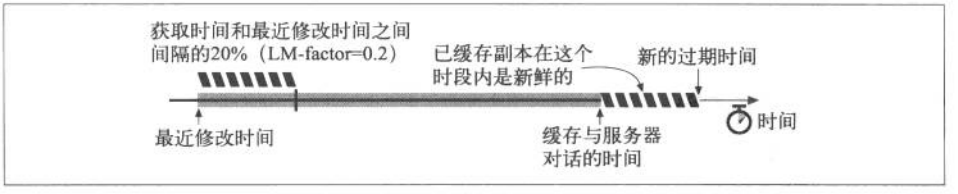

图 7-16 用 LM-factor 算法计算新鲜周期

通常人们会为试探性新鲜周期设置上限，这样它们就不会变得太大了。尽管比较保守的站点会将这个值设置为一天，但通常站点会将其设置为一周。

如果最后修改日期也没有的话，缓存就没什么信息可利用了。缓存通常会为没有任何新鲜周期线索的文档分配一个默认的新鲜周期（通常是一个小时或一天）。有时，比较保守的缓存会将这种试探性新鲜生存期设置为 0，强制缓存在每次将其提供给客户端之前，都去验证一下这些数据仍然是新鲜的。

184