HTTP 断点续传

2018.11.13 19:28

[**HTTP断点续传（分块传输）（HTTP头格式非常清楚）**](https://www.cnblogs.com/findumars/p/5745345.html)

**简述**

断点续传：指的是在上传/下载时，将任务（一个文件或压缩包）人为的划分为几个部分，每一个部分采用一个线程进行上传/下载，如果碰到网络故障，可以从已经上传/下载的部分开始继续上传/下载未完成的部分，而没有必要从头开始上传/下载。可以节省时间，提高速度。

* [简述](http://blog.csdn.net/liang19890820/article/details/53215087#%E7%AE%80%E8%BF%B0)
* [断点续传的用途](http://blog.csdn.net/liang19890820/article/details/53215087#%E6%96%AD%E7%82%B9%E7%BB%AD%E4%BC%A0%E7%9A%84%E7%94%A8%E9%80%94)
* [Range Content-Range](http://blog.csdn.net/liang19890820/article/details/53215087#range-content-range)
* [增强校验](http://blog.csdn.net/liang19890820/article/details/53215087#%E5%A2%9E%E5%BC%BA%E6%A0%A1%E9%AA%8C)
  + [Last-Modified](http://blog.csdn.net/liang19890820/article/details/53215087#last-modified)
  + [Etag](http://blog.csdn.net/liang19890820/article/details/53215087#etag)
  + [If-Range](http://blog.csdn.net/liang19890820/article/details/53215087#if-range)
  + [工作原理](http://blog.csdn.net/liang19890820/article/details/53215087#%E5%B7%A5%E4%BD%9C%E5%8E%9F%E7%90%86)
* [检测服务器是否支持断点续传](http://blog.csdn.net/liang19890820/article/details/53215087#%E6%A3%80%E6%B5%8B%E6%9C%8D%E5%8A%A1%E5%99%A8%E6%98%AF%E5%90%A6%E6%94%AF%E6%8C%81%E6%96%AD%E7%82%B9%E7%BB%AD%E4%BC%A0)

**断点续传的用途**

有时用户上传/下载文件需要历时数小时，万一线路中断，不具备断点续传的 HTTP/FTP 服务器或下载软件就只能从头重传，比较好的 HTTP/FTP 服务器或下载软件具有断点续传能力，允许用户从上传/下载断线的地方继续传送，这样大大减少了用户的烦恼。

常见的支持断点续传的上传/下载软件：QQ 旋风、迅雷、快车、电驴、酷6、土豆、优酷、百度视频、新浪视频、腾讯视频、百度云等。

在 Linux/Unix 系统下，常用支持断点续传的 FTP 客户端软件是 lftp。

**Range & Content-Range**

HTTP1.1 协议（RFC2616）开始支持获取文件的部分内容，这为并行下载以及断点续传提供了技术支持。它通过在 Header 里两个参数实现的，客户端发请求时对应的是 Range ，服务器端响应时对应的是 Content-Range。

**Range**

用于请求头中，指定第一个字节的位置和最后一个字节的位置，一般格式：

Range:(unit=first byte pos)-[last byte pos]

Range 头部的格式有以下几种情况：

Range: bytes=0-499 表示第 0-499 字节范围的内容   
Range: bytes=500-999 表示第 500-999 字节范围的内容   
Range: bytes=-500 表示最后 500 字节的内容   
Range: bytes=500- 表示从第 500 字节开始到文件结束部分的内容   
Range: bytes=0-0,-1 表示第一个和最后一个字节   
Range: bytes=500-600,601-999 同时指定几个范围

**Content-Range**

用于响应头中，在发出带 Range 的请求后，服务器会在 Content-Range 头部返回当前接受的范围和文件总大小。一般格式：

Content-Range: bytes (unit first byte pos) - [last byte pos]/[entity legth]

例如：

Content-Range: bytes 0-499/22400

0－499 是指当前发送的数据的范围，而 22400 则是文件的总大小。

而在响应完成后，返回的响应头内容也不同：

HTTP/1.1 200 Ok（不使用断点续传方式）   
HTTP/1.1 206 Partial Content（使用断点续传方式）

**增强校验**

在实际场景中，会出现一种情况，即在终端发起续传请求时，URL 对应的文件内容在服务器端已经发生变化，此时续传的数据肯定是错误的。如何解决这个问题了？显然此时需要有一个标识文件唯一性的方法。

在 RFC2616 中也有相应的定义，比如实现 Last-Modified 来标识文件的最后修改时间，这样即可判断出续传文件时是否已经发生过改动。同时 FC2616 中还定义有一个 ETag 的头，可以使用 ETag 头来放置文件的唯一标识。

**Last-Modified**

If-Modified-Since，和 Last-Modified 一样都是用于记录页面最后修改时间的 HTTP 头信息，只是 Last-Modified 是由服务器往客户端发送的 HTTP 头，而 If-Modified-Since 则是由客户端往服务器发送的头，可以看到，再次请求本地存在的 cache 页面时，客户端会通过 If-Modified-Since 头将先前服务器端发过来的 Last-Modified 最后修改时间戳发送回去，这是为了让服务器端进行验证，通过这个时间戳判断客户端的页面是否是最新的，如果不是最新的，则返回新的内容，如果是最新的，则返回 304 告诉客户端其本地 cache 的页面是最新的，于是客户端就可以直接从本地加载页面了，这样在网络上传输的数据就会大大减少，同时也减轻了服务器的负担。

**Etag**

Etag（Entity Tags）主要为了解决 Last-Modified 无法解决的一些问题。

1. 一些文件也许会周期性的更改，但是内容并不改变（仅改变修改时间），这时候我们并不希望客户端认为这个文件被修改了，而重新 GET。
2. 某些文件修改非常频繁，例如：在秒以下的时间内进行修改（1s 内修改了 N 次），If-Modified-Since 能检查到的粒度是 s 级的，这种修改无法判断（或者说 UNIX 记录 MTIME 只能精确到秒）。
3. 某些服务器不能精确的得到文件的最后修改时间。

为此，HTTP/1.1 引入了 Etag。Etag 仅仅是一个和文件相关的标记，可以是一个版本标记，例如：v1.0.0；或者说 “627-4d648041f6b80” 这么一串看起来很神秘的编码。但是 HTTP/1.1 标准并没有规定 Etag 的内容是什么或者说要怎么实现，唯一规定的是 Etag 需要放在 “” 内。

**If-Range**

用于判断实体是否发生改变，如果实体未改变，服务器发送客户端丢失的部分，否则发送整个实体。一般格式：

If-Range: Etag | HTTP-Date

也就是说，If-Range 可以使用 Etag 或者 Last-Modified 返回的值。当没有 ETage 却有 Last-modified 时，可以把 Last-modified 作为 If-Range 字段的值。

例如：

If-Range: “627-4d648041f6b80”   
If-Range: Fri, 22 Feb 2013 03:45:02 GMT

If-Range 必须与 Range 配套使用。如果请求报文中没有 Range，那么 If-Range 就会被忽略。如果服务器不支持 If-Range，那么 Range 也会被忽略。

如果请求报文中的 Etag 与服务器目标内容的 Etag 相等，即没有发生变化，那么应答报文的状态码为 206。如果服务器目标内容发生了变化，那么应答报文的状态码为 200。

用于校验的其他 HTTP 头信息：If-Match/If-None-Match、If-Modified-Since/If-Unmodified-Since。

**工作原理**

Etag 由服务器端生成，客户端通过 If-Range 条件判断请求来验证资源是否修改。请求一个文件的流程如下：

第一次请求：

1. 客户端发起 HTTP GET 请求一个文件。
2. 服务器处理请求，返回文件内容以及相应的 Header，其中包括 Etag（例如：627-4d648041f6b80）（假设服务器支持 Etag 生成并已开启了 Etag）状态码为 200。

第二次请求（断点续传）：

1. 客户端发起 HTTP GET 请求一个文件，同时发送 If-Range（该头的内容就是第一次请求时服务器返回的 Etag：627-4d648041f6b80）。
2. 服务器判断接收到的 Etag 和计算出来的 Etag 是否匹配，如果匹配，那么响应的状态码为 206；否则，状态码为 200。

**检测服务器是否支持断点续传**

CURL 实现检测：

[root@localhost ~]# curl -i --range 0-9 http://www.baidu.com/img/bdlogo.gif

HTTP/1.1 206 Partial Content

Date: Mon, 21 Nov 2016 05:26:29 GMT

Server: Apache

P3P: CP=" OTI DSP COR IVA OUR IND COM "

Set-Cookie: BAIDUID=0CD0E23B4D4F739954DFEDB92BE6CE03:FG=1; expires=Tue, 21-Nov-17 05:26:29 GMT; max-age=31536000; path=/; domain=.baidu.com; version=1

Last-Modified: Fri, 22 Feb 2013 03:45:02 GMT

ETag: "627-4d648041f6b80"

Accept-Ranges: bytes

Content-Length: 10

Cache-Control: max-age=315360000

Expires: Thu, 19 Nov 2026 05:26:29 GMT

Content-Range: bytes 0-9/1575

Connection: Keep-Alive

Content-Type: image/gif

GIF89a[root@localhost ~]#

能够找到 Content-Range，则表明服务器支持断点续传。有些服务器还会返回 Accept-Ranges，输出结果 Accept-Ranges: bytes ，说明服务器支持按字节下载。

http://blog.csdn.net/liang19890820/article/details/53215087

分类: [HTTP-TCP等各种协议](https://www.cnblogs.com/findumars/category/640688.html)