

## 16 | 把大象装进冰箱：HTTP传输大文件的方法

2019-07-03 Chrono

《透视HTTP协议》

课程介绍 >



讲述：Chrono

时长 11:15 大小 12.89M



上次我们谈到了 HTTP 报文里的 body，知道了 HTTP 可以传输很多种类的数据，不仅是文本，也能传输图片、音频和视频。

早期互联网上传输的基本上都是只有几 K 大小的文本和小图片，现在的情况则大有不同。网页里包含的信息实在是太多了，随随便便一个主页 HTML 就有可能上百 K，高质量的图片都以 M 论，更不要说那些电影、电视剧了，几 G、几十 G 都有可能。

相比之下，100M 的光纤固网或者 4G 移动网络在这些大文件的压力下都变成了“小水管”，无论是上传还是下载，都会把网络传输链路挤的“满满当当”。

领资料

所以，如何在有限的带宽下高效快捷地传输这些大文件就成了一个重要的课题。这就好比是已经打开了冰箱门（建立连接），该怎么把大象（文件）塞进去再关上门（完成传输）呢？

今天我们就一起看看 HTTP 协议里有哪些手段能解决这个问题。

## 数据压缩

还记得上一讲中说到的“数据类型与编码”吗？如果你还有印象的话，肯定能够想到一个最基本的解决方案，那就是“**数据压缩**”，把大象变成小猪佩奇，再放进冰箱。

通常浏览器在发送请求时都会带着“**Accept-Encoding**”头字段，里面是浏览器支持的压缩格式列表，例如 gzip、deflate、br 等，这样服务器就可以从中选择一种压缩算法，放进“**Content-Encoding**”响应头里，再把原数据压缩后发给浏览器。

如果压缩率能有 50%，也就是说 100K 的数据能够压缩成 50K 的大小，那么就相当于在带宽不变的情况下网速提升了一倍，加速的效果是非常明显的。

不过这个解决方法也有个缺点，gzip 等压缩算法通常只对文本文件有较好的压缩率，而图片、音频视频等多媒体数据本身就已经是高度压缩的，再用 gzip 处理也不会变小（甚至还有可能会增大一点），所以它就失效了。

不过数据压缩在处理文本的时候效果还是很好的，所以各大网站的服务器都会使用这个手段作为“保底”。例如，在 Nginx 里就会使用“gzip on”指令，启用对“text/html”的压缩。

## 分块传输

在数据压缩之外，还能有什么办法来解决大文件的问题呢？

压缩是把大文件整体变小，我们可以反过来思考，如果大文件整体不能变小，那就把它“拆开”，分解成多个小块，把这些小块分批发给浏览器，浏览器收到后再组装复原。

这样浏览器和服务端都不用在内存里保存文件的全部，每次只收发一小部分，网络也不会被大文件长时间占用，内存、带宽等资源也就节省下来了。

这种“**化整为零**”的思路在 HTTP 协议里就是“**chunked**”分块传输编码，在响应报文里用头字段“**Transfer-Encoding: chunked**”来表示，意思是报文里的 body 部分不是一次性发过来的，而是分成了许多的块（chunk）逐个发送。

这就好比是用魔法把大象变成“乐高积木”，拆散了逐个装进冰箱，到达目的地后再施法拼起来“满血复活”。

领资料



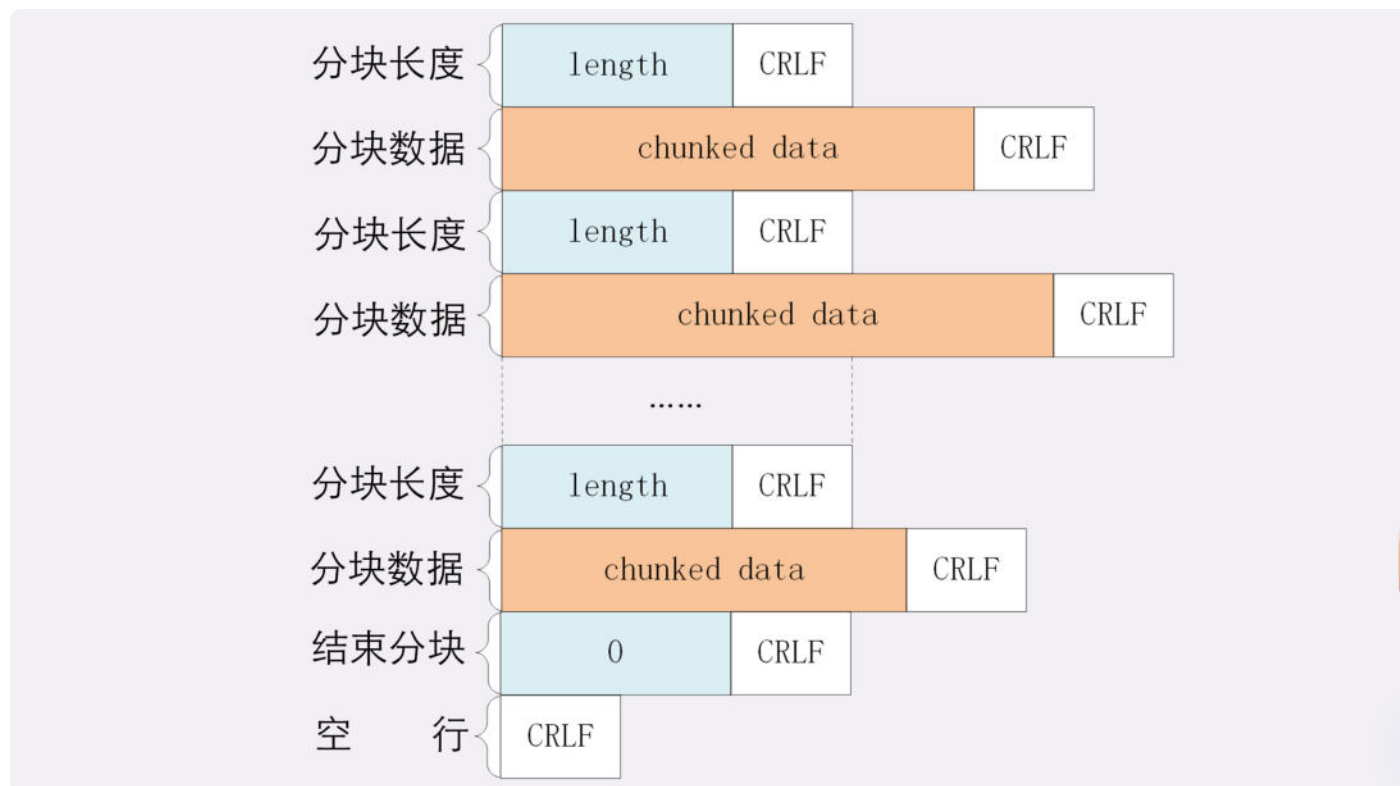
分块传输也可以用于“流式数据”，例如由数据库动态生成的表单页面，这种情况下 body 数据的长度是未知的，无法在头字段“**Content-Length**”里给出确切的长度，所以也只能用 chunked 方式分块发送。

“Transfer-Encoding: chunked”和“Content-Length”这两个字段是**互斥的**，也就是说响应报文里这两个字段不能同时出现，一个响应报文的传输要么是长度已知，要么是长度未知（chunked），这一点你一定要记住。

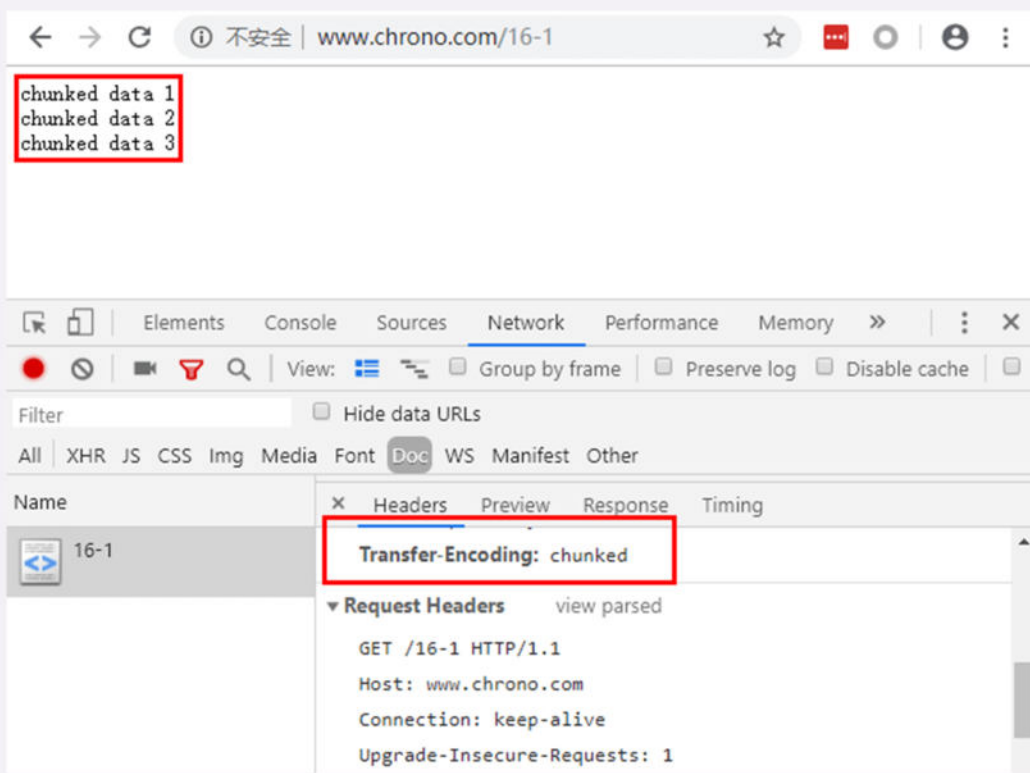
下面我们来看一下分块传输的编码规则，其实也很简单，同样采用了明文的方式，很类似响应头。

1. 每个分块包含两个部分，长度头和数据块；
2. 长度头是以 CRLF（回车换行，即\r\n）结尾的一行明文，用 16 进制数字表示长度；
3. 数据块紧跟在长度头后，最后也用 CRLF 结尾，但数据不包含 CRLF；
4. 最后用一个长度为 0 的块表示结束，即“0\r\n\r\n”。

听起来好像有点难懂，看一下图就好理解了：



实验环境里的 URI“/16-1”简单地模拟了分块传输，可以用 Chrome 访问这个地址看一下效果：



不过浏览器在收到分块传输的数据后会自动按照规则去掉分块编码，重新组装出内容，所以想要看到服务器发出的原始报文形态就得用 Telnet 手工发送请求（或者用 Wireshark 抓包）：

复制代码

```
1 GET /16-1 HTTP/1.1
2 Host: www.chrono.com
```

因为 Telnet 只是收到响应报文就完事了，不会解析分块数据，所以可以很清楚地看到响应报文里的 chunked 数据格式：先是一行 16 进制长度，然后是数据，然后再是 16 进制长度和数据，如此重复，最后是 0 长度分块结束。

领资料



```
Content-Type: text/plain
Transfer-Encoding: chunked
Connection: keep-alive
```

```
f
chunked data 1
```

```
f
chunked data 2
```

```
f
chunked data 3
```

```
0
```

## 范围请求

有了分块传输编码，服务器就可以轻松地收发大文件了，但对于上 G 的超大文件，还有一些问题需要考虑。

比如，你在看当下正热播的某穿越剧，想跳过片头，直接看正片，或者有段剧情很无聊，想拖动进度条快进几分钟，这实际上是想获取一个大文件其中的片段数据，而分块传输并没有这个能力。

HTTP 协议为了满足这样的需求，提出了“**范围请求**”（range requests）的概念，允许客户端在请求头里使用专用字段来表示只获取文件的一部分，相当于是**客户端的“化整为零”**。

范围请求不是 Web 服务器必备的功能，可以实现也可以不实现，所以服务器必须在响应头里使用字段“**Accept-Ranges: bytes**”明确告知客户端：“我是支持范围请求的”。

如果不支持的话该怎么办呢？服务器可以发送“Accept-Ranges: none”，或者干脆不发送“Accept-Ranges”字段，这样客户端就认为服务器没有实现范围请求功能，只能老老实实地收发整块文件了。

请求头 **Range** 是 HTTP 范围请求的专用字段，格式是“**bytes=x-y**”，其中的 x 和 y 是以字节为单位的数据范围。

领资料



要注意 x、y 表示的是“偏移量”，范围必须从 0 计数，例如前 10 个字节表示为“0-9”，第二个 10 字节表示为“10-19”，而“0-10”实际上是前 11 个字节。

Range 的格式也很灵活，起点 x 和终点 y 可以省略，能够很方便地表示正数或者倒数的范围。假设文件是 100 个字节，那么：

- “0-”表示从文档起点到文档终点，相当于“0-99”，即整个文件；
- “10-”是从第 10 个字节开始到文档末尾，相当于“10-99”；
- “-1”是文档的最后一个字节，相当于“99-99”；
- “-10”是从文档末尾倒数 10 个字节，相当于“90-99”。

服务器收到 Range 字段后，需要做四件事。

第一，它必须检查范围是否合法，比如文件只有 100 个字节，但请求“200-300”，这就是范围越界了。服务器就会返回状态码 **416**，意思是“你的范围请求有误，我无法处理，请再检查一下”。

第二，如果范围正确，服务器就可以根据 Range 头计算偏移量，读取文件的片段了，返回状态码“**206 Partial Content**”，和 200 的意思差不多，但表示 body 只是原数据的一部分。

第三，服务器要添加一个响应头字段 **Content-Range**，告诉片段的实际偏移量和资源的总大小，格式是“**bytes x-y/length**”，与 Range 头区别在没有“=”，范围后多了总长度。例如，对于“0-10”的范围请求，值就是“bytes 0-10/100”。

最后剩下的就是发送数据了，直接把片段用 TCP 发给客户端，一个范围请求就算是处理完了。

你可以用实验环境的 URI“/16-2”来测试范围请求，它处理的对象是“/mime/a.txt”。不过我们不能用 Chrome 浏览器，因为它没有编辑 HTTP 请求头的功能（这点上不如 Firefox 方便），所以还是要用 Telnet。


例如下面的这个请求使用 Range 字段获取了文件的前 32 个字节：

领资料





```
1 GET /16-2 HTTP/1.1
2 Host: www.chrono.com
3 Range: bytes=0-31
```

 复制代码

返回的数据是（去掉了几个无关字段）：

```
1 HTTP/1.1 206 Partial Content
2 Content-Length: 32
3 Accept-Ranges: bytes
4 Content-Range: bytes 0-31/96
5
6 // this is a plain text json doc
```

 复制代码

有了范围请求之后，HTTP 处理大文件就更加轻松了，看视频时可以根据时间点计算出文件的 Range，不用下载整个文件，直接精确获取片段所在的数据内容。

不仅看视频的拖拽进度需要范围请求，常用的下载工具里的多段下载、断点续传也是基于它实现的，要点是：

- 先发个 HEAD，看服务器是否支持范围请求，同时获取文件的大小；
- 开 N 个线程，每个线程使用 Range 字段划分出各自负责下载的片段，发请求传输数据；
- 下载意外中断也不怕，不必重头再来一遍，只要根据上次的下载记录，用 Range 请求剩下的那一部分就可以了。

## 多段数据

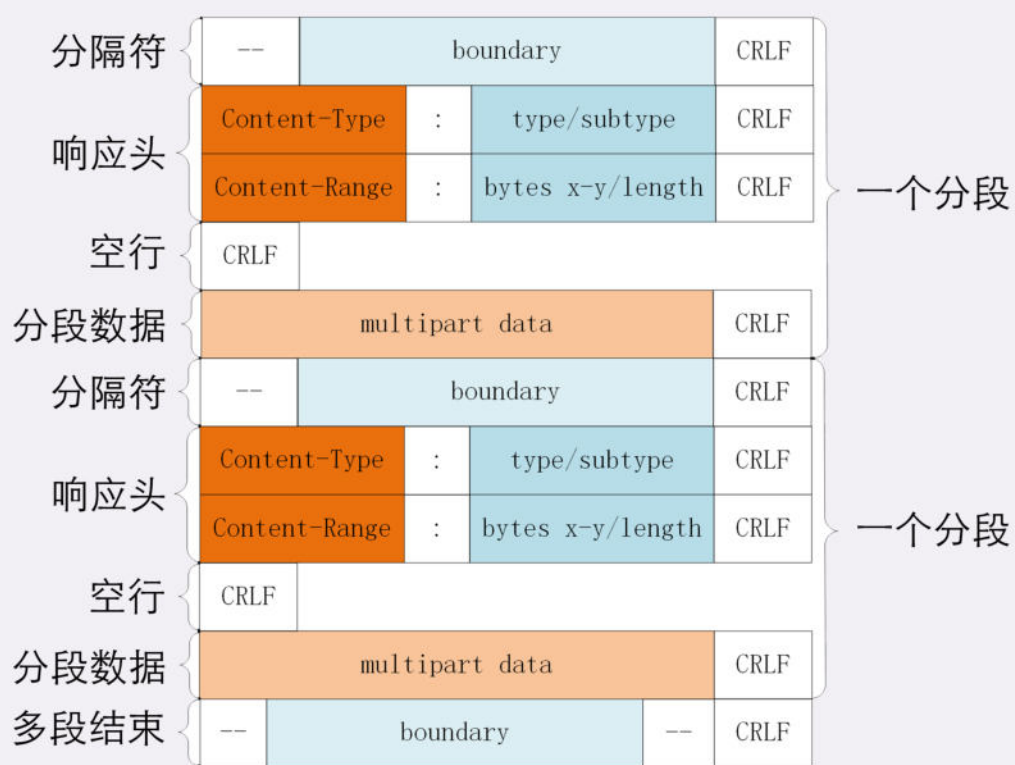
刚才说的范围请求一次只获取一个片段，其实它还支持在 Range 头里使用多个“x-y”，一次性获取多个片段数据。

这种情况需要使用一种特殊的 MIME 类型：“**multipart/byteranges**”，表示报文的 body 是由多段字节序列组成的，并且还要用一个参数“**boundary=xxx**”给出段之间的分隔标记。

多段数据的格式与分块传输也比较类似，但它需要用分隔标记 boundary 来区分不同的片段，可以通过图来对比一下。

 领资料





每一个分段必须以“--boundary”开始（前面加两个“-”），之后要用“Content-Type”和“Content-Range”标记这段数据的类型和所在范围，然后就像普通的响应头一样以回车换行结束，再加上分段数据，最后用一个“--boundary--”（前后各有两个“-”）表示所有的分段结束。

例如，我们在实验环境里用 Telnet 发出有两个范围请求：

复制代码

```
1 GET /16-2 HTTP/1.1
2 Host: www.chrono.com
3 Range: bytes=0-9, 20-29
```

得到的就会是下面这样：

复制代码

```
1 HTTP/1.1 206 Partial Content
2 Content-Type: multipart/byteranges; boundary=000000000001
3 Content-Length: 189
4 Connection: keep-alive
5 Accept-Ranges: bytes
6
7
8 --000000000001
9 Content-Type: text/plain
```

领资料





```
10 Content-Range: bytes 0-9/96
11
12 // this is
13 --000000000001
14 Content-Type: text/plain
15 Content-Range: bytes 20-29/96
16
17 ext json d
18 --000000000001--
```

报文里的“--000000000001”就是多段的分隔符，使用它客户端就可以很容易地区分出多段 Range 数据。

## 小结

今天我们学习了 HTTP 传输大文件相关的知识，在这里做一下简单小结：

1. 压缩 HTML 等文本文件是传输大文件最基本的方法；
2. 分块传输可以流式收发数据，节约内存和带宽，使用响应头字段“Transfer-Encoding: chunked”来表示，分块的格式是 16 进制长度头 + 数据块；
3. 范围请求可以只获取部分数据，即“分块请求”，实现视频拖拽或者断点续传，使用请求头字段“Range”和响应头字段“Content-Range”，响应状态码必须是 206；
4. 也可以一次请求多个范围，这时候响应报文的数据类型是“multipart/byteranges”，body 里的多个部分会用 boundary 字符串分隔。

要注意这四种方法不是互斥的，而是可以混合起来使用，例如压缩后再分块传输，或者分段后再分块，实验环境的 URI“/16-3”就模拟了后一种的情形，你可以自己用 Telnet 试一下。

## 课下作业

1. 分块传输数据的时候，如果数据里含有回车换行（\r\n）是否会影响分块的处理呢？
2. 如果对一个被 gzip 的文件执行范围请求，比如“Range: bytes=10-19”，那么这个范围是应用于原文件还是压缩后的文件呢？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。





## == 课外小贴士 ==

- 01 gzip 的压缩率通常能够超过 60%，而 br 算法是专为 HTML 设计的，压缩效率和性能比 gzip 还要好，能够再提高 20% 的压缩密度。
- 02 Nginx 的“gzip on”指令很智能，只会压缩文本数据，不会压缩图片、音频、视频。
- 03 Transfer-Encoding 字段最常见的值是 chunked，但也可以用 gzip、deflate 等，表示传输时使用了压缩编码。注意这与 Content - Encoding 不同，Transfer - Encoding 在传输后会被自动解码还原出原始数据，而 Content - Encoding 则必须由应用自行解码。
- 04 分块传输在末尾还允许有“拖尾数据”，由响应头字段 Trailer 指定。
- 05 与 Range 有关的还有一个 If-Range，即条件范围请求，将在第 20 讲介绍。

分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

生成海报并分享

赞 22

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 15 | 海纳百川：HTTP的实体数据

下一篇 17 | 排队也要讲效率：HTTP的连接管理

## 学习推荐

# JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费



领资料

## 精选留言 (80)

写留言



Aaaaaaaaayou

2019-07-03

老师，有个问题：http交给tcp进行传输的时候本来就会分块，那http分块的意义是什么呢？

作者回复: 在http层是看不到tcp的，它不知道下层协议是否会分块，下层是否分块对它来说没有意义，不关心。

在http里一个报文必须是完整交付，在处理大文件的时候就很不方便，所以就要分块，在http层面方便处理。

chunked主要是在http的层次来解决问题。

共 13 条评论 >

👍 68



chengzise

2019-07-03

1. 分块传输中数据里含有回车换行（\r\n）不影响分块处理，因为分块前有数据长度说明
2. 范围是应用于压缩后的文件

作者回复: 1正确。

2需要分情况，看原文件是什么形式。如果原来的文件是gzip的，那就正确。如果原文件是文本，而是在传输过程中被压缩，那么就应用于压缩前的数据。

总之，range是针对原文件的。



👍 53



赵健

2019-07-04

“Transfer-Encoding: chunked”和“Content-Length”这两个字段是互斥的，也就是说响应报文里这两个字段不能同时出现，一个响应报文的传输要么是长度已知，要么是长度未知（chunked），这一点你一定要记住。老师请问下，为啥分块意味着长度未知，后面不是提到块里面有个长度头嘛？而且单个块应该是一次http传输的内容，既然块里有长度头，那这次传输的内容长度也就能算出来，这次http的Content-Length 也就知道啊！是我理解错了吗

作者回复: 举个例子，从GitHub上下载源码包，GitHub要实时压缩实时发送，而不是一下子压缩好再发送，这样body的长度一开始就是未知的。

所以就要用分块编码，压缩一部分，就发一部分，这部分的长度是已知的，但总长度只有压缩完才能知道。

chunked编码用在“流式”收发数据的时候，通常数据是即时生成的，也就是动态数据。

领资料



**秋水共长天一色** 🇨🇳

2019-07-29

老师，我有些问题需要问问您。

- 1.比如我在视频网上看电影，我们经常能看到进度条里面有一条灰色的缓存进度，我是否能理解成这个进度就是分块传输的一个进度显示吗？
  - 2.刚刚我有看到评论说过一个问题就是分块传输的时候是由一个请求和一个响应完成的，如果我们在抓一个需要10分钟才能完成分块传输的请求时，我是不是就会看到这个请求在这10分钟内都是一个正在响应的状态吗？
  - 3.为什么我们在对一些视频网站看视频抓包的时候却无法捕抓到这个请求呢？
  - 4.如果我们在看完视频后在浏览器缓存里发现一些片段式的视频文件，能否就说明这个是用分块传输呢？
  - 5.如果我们在看视频拖动进度条到10分30秒，到最后视频会从10分20秒开始播放，能否说明10分30秒的这个分块的头是在10分20秒呢？
  - 6.请问多段数据能理解成一次性获取分块传输里多个连续的分块的数据的意思吗？
- 还有就是非常感谢老师把这些知识点讲的那么细，我近期多个面试里都有被问到相关的知识，多亏老师的讲解我才能顺利应付，谢谢老师！！

作者回复：

- 1.网络视频不一定用的就是http协议，也可能是其他的专用协议，所以不能简单地判断就是分块传输。如果是http协议，对于大文件通常都是range请求，也不一定用chunk分块。
- 2.是的，这是由http的请求-应答工作模式决定的。不管是不是chunk，只要响应没有结束，这个来回就不会完成。
- 3.也可能用的是其他协议。
- 4.应该是range的分段，不是chunk，chunk只是在传输过程中分块，最后到客户端会是一个完整的文件。
- 5.视频文件的分段计算比较复杂，课程里面只是作为简单的示例，实际情况不一定就是这么简单。
- 6.分段的range和分块的chunk是两个完全无关的概念，不要弄混了。chunk是传输时分成小块逐个发送，range是取大文件中间的一部分。
- 7.不客气，后面的https、http/2也继续努力吧。

[领资料](#)**小桶**

分块传输，客户端只需要发一次请求，还是发多次请求呢？使用分块传输时，客户端与服务器是怎样工作的呢

作者回复: http传输永远是一个请求一个响应的工作模式，只是响应是chunked分块，body数据不是一次性发过来的，而是分批分块发送，但仍然是在一个报文里。

客户端发送请求后等待响应，服务器组织数据，分块发送，最后一个分块是结束标志。客户端依次接收分块，收到结束标志后就把数据拼成完整的报文。

共 7 条评论 >

👍 30



一步

2019-07-04

对于问题2,range是针对原文的还是压缩后的，可以想象一下看视频的时候，我们拖拽进度条请求的range范围是针对原视频长度的，如果针对压缩后的，那么我们实际拖拽的范围和响应的数据范围就不一致了

作者回复: 说的很好。



👍 15



-W.LI-

2019-07-03

老师好!在带宽固定的情况下，范围请求没发提高下载速度。如果服务器对客户端每个累链接限速的情况下，可通过多线程并发下载，提高下载速度是么?还有几个问题

分块传输:顺序传一次一小块

范围请求:支持跳跃式传输，还可以并发获取不同的range最后合并。

多段数据:一次请求多个范围，范围可以不连续是么?如果必须联系的话和请求一个大范围没差别了。

这几个拒的例子都是服务端这么返回的。

客户端上传的时候怎么使用呢?老师后面会讲么。

只读到了这么点，希望老师补充下每个的作用，和解决的问题，谢谢老师。

作者回复: 1，是的，通过多线程并发下载，提高下载速度。

2，范围可以不连续，例子里就是这样。

3，客户端上传的时候也可以用chunked、gzip，但不能用range。

注意这些字段的类型，只要是实体字段，那就在请求响应里都可以用。

领资料







13



一只鱼

2020-04-10

针对课下作业2：

情况一：如果服务器上只有 gzip 之后的文件，没有原文件，那范围请求针对的就是 gzip 之后的文件；

情况二：如果服务器上有原文件(未压缩)，只是在传输过程中被 gzip，那范围请求针对的就是未压缩的原文件。

这里拓展一下，假如在服务器和客户端之间有一个 cdn，那么 cdn 缓存的是文件的某个范围吗？cdn 会根据请求头判断缓存里面有没有这个范围的结果，如果有就直接返回，并没有再根据 bytes 进行计算？

作者回复：说的对。

第二个问题，如果按标准 http 协议的做法就是只缓存一个范围，但这样的效果不会很好，因为缓存的这部分被重复利用的机会很小。

更好的方法是 cdn 返回给客户端的同时，把整个文件都从源站取下来，之后就可以自己计算范围给客户端，不用再从源站取了。



8



darren

2021-04-07

不分块：http 把客户端需要的东西整个交给 tcp，由 tcp 切块后发送给客户端，客户端接受后在 tcp 层组装完整发给浏览器使用。

分块：http 把客户端需要的东西切分成 1、2、3 到 n 块，然后将 1 块发给 tcp，tcp 将 1 块再次切分后发给客户端，客户端接受后在 tcp 组装成块 1 发给 http 层。然后服务器与客户端用同样的方式发送块 2、块 3 到块 n。客户端的 http 在接收完所有块后组装成一个完整的响应。整个过程使用同一个 tcp 连接，块 1 到块 n 如是挨个发送的。如果是 http2，则基于多路复用技术块 1 到块 n 可以同时发送。所以分块抓包 http 只能抓到一个包，如果抓 tcp 的包，分不分块，都会抓到很多包。

分段：分段就是对某个资源的一部分进行请求（类似于把一个大文件切分成很多小文件，类似压缩中的分卷功能，然后客户端只对这些小文件中的一部分进行请求）

分段是对需要哪些资源进行一种说明，分块是一种传输机制，完全不同的两个东西，只是名字比较像。

请老师指教理解不正确的地方，另外想问一下老师分块的时候每个块都会复制一次响应头吗，还是只有块 1 带有请求头。

领资料



作者回复: 总结的非常好, 很清晰。

chunk数据实际上是一个很大的请求, 只有一个header, 只是逐个地发出去, 而不是一次性发。

可以在实验环境里抓包看看。



6



ly

2019-08-18

有几个小疑问:

分块传输:

对于一个500Mb的数据, 客户端应该是发送N次http请求, 每次http请求只传输其中一部分, 每次都是采用了分块传输的body格式, 那么每次都会重新建立TCP连接吗 (三次握手)?

另外文章提到分块传输中的“流式数据”, 这个流式数据怎么理解呢?

对于多段数据:

服务端在响应body里面的每一段都会指定Content-Type和Content-Range, 总感觉其中的Content-Type字段是多余的, 难道body里面的不同分段, Content-Type可能不一样?

作者回复:

1. 请参考一下讲长连接的那一讲, 现在的http请求应该都是使用的长连接, 不会每次都重新建立连接, 否则效率太低。

2. 流式数据就是分块发送, 像tcp那样一块一块地发过来, 而不是像普通http那样一次完整地全发过来。

3. 多段数据可以是multipart, 每段数据是有可能不同的, 比如发出一个请求, 服务器返回了一个txt和json, 所以要这么做。当然这种情况比较少见, 但协议就必须考虑到这种情况。

共 2 条评论 >



5



wheat7

2019-07-12

chunk的核心问题并不是所谓把大象装进冰箱, 是为了解决应用层在没有content-length的时候知道数据在哪里结束, chunk和普通传输方式都是在一个http报文里传输的, 只是在body里相当于又加了一层协议或者是编码, 数据无论如何是在一头大象里, 在一个http报文中传输, 大的数据传输使用chunk和不使用传输方式并没有什么区别。

领资料



作者回复: 说的很有道理。

不过两者还是有区别的, chunked主要用于流式数据, 一开始不能知道准确的大小。

共 4 条评论 >

👍 5



lesserror

2019-12-04

老师, 请问一下: “分块传输也可以用于“流式数据””。该怎么理解这个“流式数据”这句话呢?

作者回复: 流式数据就是stream, 虽然还是用一个一个的chunk发过来的, 但从更高层次上看, 它像是从源头持续不断地、慢慢地“流”过来的, 而不是一次性、一整块发过来的。

可以对比一下tcp和udp, tcp是流式数据, 而udp是一个一个的数据包。



👍 3



响雨

2019-07-05

响应报文返回的数据太大, 所以采用了chunk分块传输的话, 那响应报文在传输完成前是什么样子, 响应行和头过来了, 响应体还在流式传递, 那响应体内的数据该怎么展示?

作者回复: 就是按照格式分成一些片段, 逐块发送, 浏览器收到后去掉分隔符, 再拼成原来的样子。



👍 3



Gopher

2019-07-03

这个专栏质量很棒, 老师很负责, 知识讲解很通透, 很容易就get、解惑了。

哈哈, 特此留言就是想说, 老师, 你认真做事的样子真帅(\*•̀•́\*)o^

作者回复: 承蒙夸奖, 不胜感激。



👍 3



djfhchdh

2019-07-03

1、因为分块数据是明文传输, 如果数据里有\r\n, 是会影响分块处理的

领资料



## 2、个人感觉应该是应用于原文件

作者回复: 1不对, 因为chunked格式里已经有长度了。

2正确。

看来有不少同学对第二个问题比较迷惑, 我再说具体一点。

比如说, 有一个1M的纯文本, range请求其中的500K, 然后服务器编码为gzip (Content-Encoding: gzip), 压缩成200k, 浏览器收到后解压缩, 就得到了这部分的500k数据。



3



四月长安

2019-08-16

http数据包封装好交给下层tcp协议的时候, 应该是作为tcp数据部分所要传输的内容吧, ip协议数据报最大传输65535字节的数据, 这65535的数据减去tcp的首部, 应该就是tcp所能容纳的负载极限了吧, 所以如果是这样的话http数据的分块应该粒度更小才是吧, 或许一个tcp负载里边就有好多http分块? 请老师指正, 感谢🙏

作者回复: http层面的分块与tcp的分块是互不相关的, http的分块可以很大也可以很小, 这些数据到了tcp由tcp去自行组织, 是合并在一个块里还是跨越多个块都没有关系, 我们不需要为此操心。

其实tcp层再往下的mac层, 一个包最多1500字节, 但tcp也不需要考虑这个, 只要下层能够保证上层正确传输就好。



2



彧豪

2019-07-21

老师我有几个疑问:

1. 比如总共是1314的数据, 响应头会这么写`Content-Range: bytes 0-1313/1314`, 为何会少1?
2. 浏览器如何开N个线程下载数据?多个ajax请求?

作者回复: 1, 计算范围从0开始计数, 用的是C语言惯例。

2, 浏览器的技术不是很清楚, 抱歉了。

共 2 条评论 >



2

领资料





苦行僧

2019-07-04

老师 我们的业务就是视频处理 经常涉及的就是视频下载本地处理 使用的是ftp协议 如果使用http会更快?

作者回复: 这个需要实际测试, http的优势是灵活, 相关的工具多, 传输效率上都是依赖于底层的tcp, 应该差不多。



2



白了少年头

2019-07-03

1.数据里有回车换行, 会影响分块的处理

2.范围应用于压缩后文件

不知道对不对, 辛苦老师解答一下, 谢谢!

作者回复: 1不对, 前面已经有同学回答了, 因为分块包含了长度, 所以回车换行不影响。

2需要分情况, 看原文件是什么形式。如果原来的文件是gzip的, 那就正确。如果原文件是文本, 而是在传输过程中被压缩, 那么就应用于压缩前的数据。

总之, range是针对原文件的。



2



一粟

2019-07-03

迅雷下载或者在线视频播放器是不是在使用分块或者任意请求功能?

作者回复: 只要是用http传输, 就会用range, 但他们也有可能用自己的协议而不是http, 比如rtmp。



2

领资料

