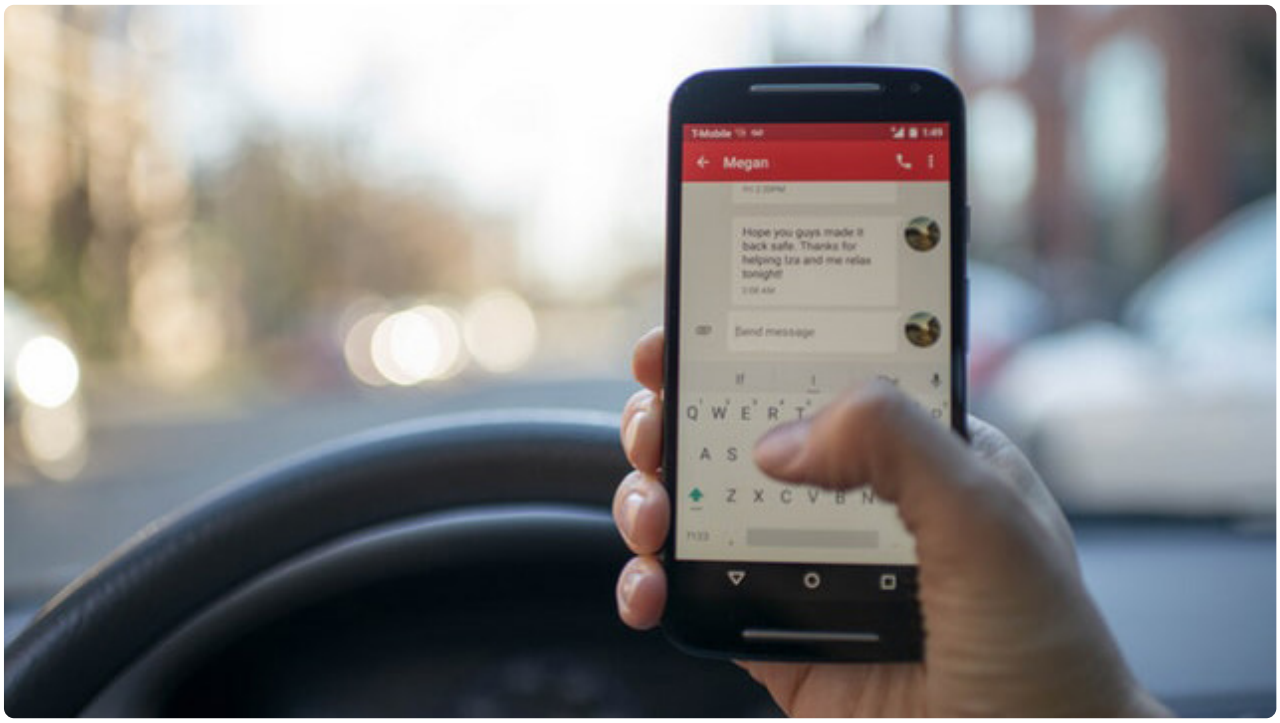
22 压缩压缩再压缩

更新时间：2020-02-25 19:30:52



> 人的一生可能燃烧也可能腐朽，我不能腐朽，我愿意燃烧起来！——奥斯特洛夫斯基

前言

我们知道，对于所有的产品来说，用户的体验性是非常重要的。用户获取数据的速度越快，体验就会越好。所以，我们的研发同学在开发功能的时候会仔细的设计数据类型，格式，数据库的表结构等等，所有的一切都是为了让产品速度更快，用户体验更好(PS:虽然设计的很好，但是经不住PM乱改需求，哎，说起来都是痛)。这些都是我们应用程序级别的优化，那么Nginx作为这么牛逼的服务器，当然也给我们提供了相应的功能，这就是我们本文要说的GZIP压缩。

GZIP功能

GZIP是一种压缩算法，可以把一些大文件压缩为小文件，浏览器在接收到压缩文件的时候，会根据压缩算法解压文件，这样就可以减少网络传输过程中的带宽消耗和时间消耗。

Nginx中通过ngx_http_gzip_module模块实现gzip压缩。

Module ngx_http_gzip_module

这个模块的指令比较少，我们在后面的实练中会解释相应的功能。

实际操作

我们实际操作一下，看看这个gzip相关指令时如何使用的。

首先，我们准备一个静态html文件test.html，如下：

```
[root@953198c7d1f6 nginx]#
[root@953198c7d1f6 nginx]# ls -lah html/test.html
-rw-r--r-- 1 root root 563K Feb 25 05:24 html/test.html
[root@953198c7d1f6 nginx]#
[root@953198c7d1f6 nginx]#
```

不使用gzip压缩

- 配置nginx不打开gzip，配置如下：

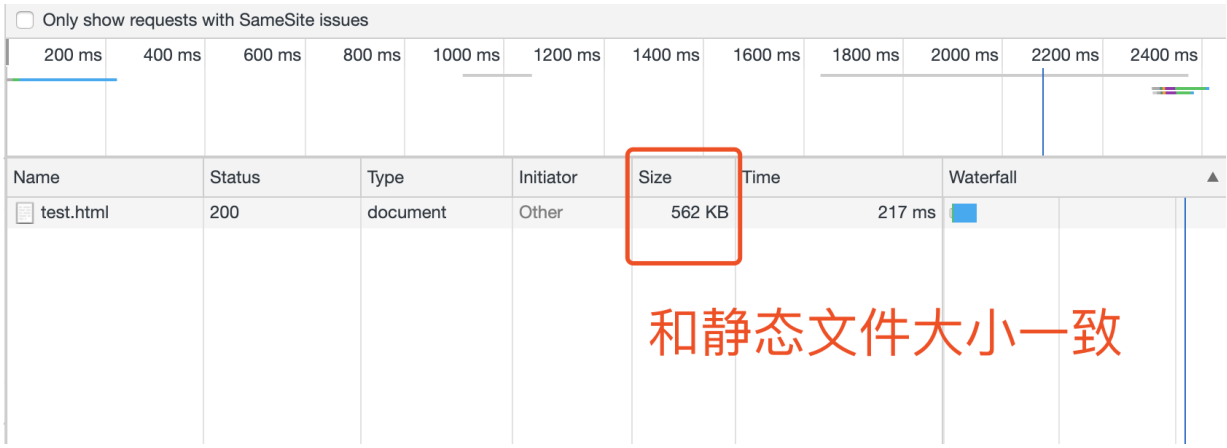
```
    server {
        listen       80;
        server_name  localhost;
#        gzip on;

        location / {
            root   html;
            index  index.html index.htm;
            chunked_transfer_encoding off;
            add_header Cache-Control no-cache;
            add_header Cache-Control private;
            expires -1s;
        }

    }
```

禁止浏览器端缓存

- 启动 Nginx，我们从服务器端请求 test.html 文件，看一下传输的情况：



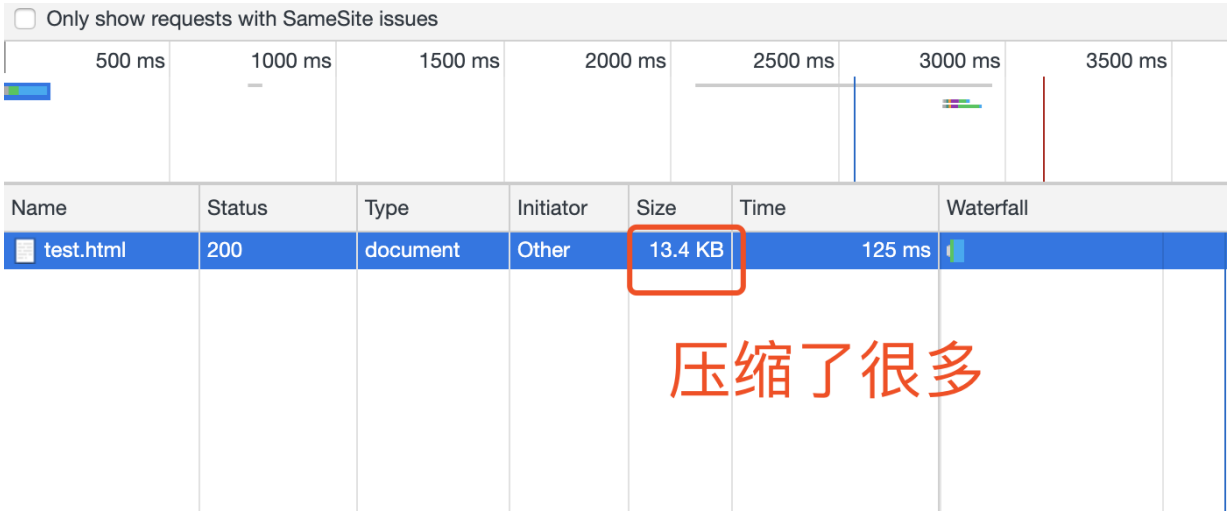
使用 gzip 压缩

将 nginx 配置打开 gzip，配置如下：

```
14      server {
15          listen       80;
16          server_name  localhost;
17          gzip on;
18          gzip_comp_level 6;
19          gzip_buffers 16 8k;
20          gzip_min_length 256;
21          gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss
    text/javascript;
22
23          location / {
24              root   html;
25              index  index.html index.htm;
26              chunked_transfer_encoding off;
27              add_header Cache-Control no-cache;
28              add_header Cache-Control private;
29              expires -1s;
30          }
31
32      }
```

gzip 相关的配置

重启 Nginx，重新访问 test.html 文件，对比一下这一次传输文件的大小：

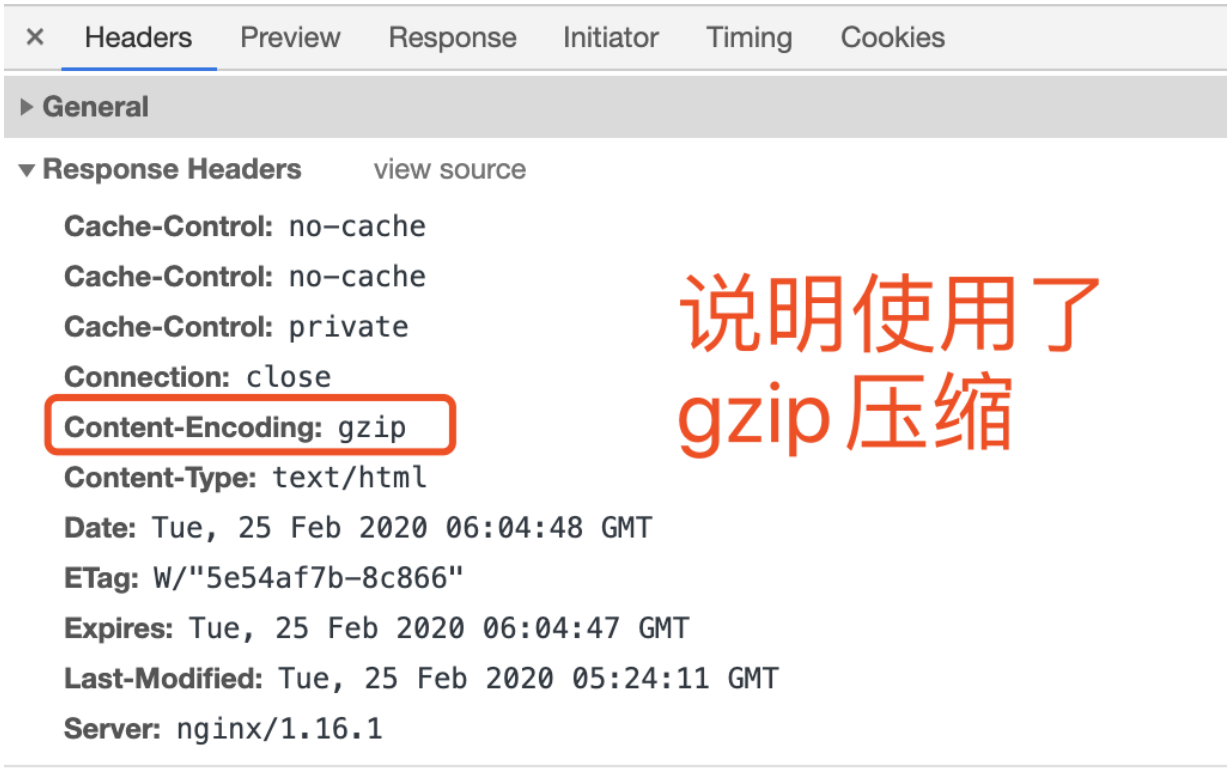


对比我们使用 gzip 前后的操作：

首先，使用 gzip 压缩之后的体积是之前的 1/40 左右，传输体积大大的减小了。

其次，使用 gzip 之后传输耗时从 217ms 降低到了 125ms，耗时降低了将近一半。

除了上面的方法之外，我们也可以通过响应头部来确认是否启用了 gzip 压缩。



指令分析

我们在 nginx 配置文件找那个使用到了几个 gzip 指令，我们说一下最重要的几个指令。

gzip

毫无疑问这个指令时最重要的，它表示是否使用 gzip 压缩内容。

gzip_comp_level 压缩率

这个参数是一个1~9的数值，数值越大，表示压缩比率越高，压缩之后的文件体积越小，传输的越快。
但是这个值并不是越大越好。压缩操作是一个耗CPU的操作，压缩比越大，压缩本身耗时就越多，并且随着压缩比率的提高，文件体积减小的并不是特别明显，所以并不要将这个值设置的特别大，建议设置为4~6。

gzip_min_length起始压缩长度

gzip是一种压缩算法，它会在最终的压缩内容中增加一些标识字段，这样浏览器收到内容之后可以根据这些内容进行解压操作。所以对于体积比较小的内容，尽量不要进行压缩。这样可以节约一些时间。

> 我曾经对一个空字符串进行过gzip压缩，结果压缩之后的长度居然变成了20~

gzip_types压缩文件类型

这个配置项表示的是要对哪些文件类型进行压缩。默认情况下只对text/html类型的文件进行压缩，如果我们想对css以及js文件进行压缩，可以将这些文件对应的MIME Type添加到这个配置项中。
有一个特殊的*，这个值表示对所有的文件类型都进行压缩操作。

上面几个配置项是我们使用gzip过程中经常会用到的选项，我们要牢记这几个的作用。

注意事项

既然gzip这么犀利，那是不是所有的资源都可以进行压缩呢？其实不然，对于文本文件，比如html， css， js等，可以使用gzip进行压缩。而有一些资源则尽量不要进行压缩。

- 图片资源：比如png，jpg类型的图片，他们本身就已经进行了压缩。所以，即使使用了gzip压缩，压缩前后体积大小也不会有明显的变化。甚至会变得更大(因为增加了gzip特有的字段)。Google，baidu都没有对图片进行压缩。
- 大文件和视频类型：压缩这些文件会消耗大量的CPU，并且压缩之后也不一定符合我们的预期。
- 特别小的文件：对这些小文件的压缩，可能会导致越压缩体积越大的尴尬现象（具体原因我们已经在上面阐述）。

总结

这就是Nginx中关于Gzip相关的内容，你懂了吗？

}