公告
confach@Gmail.com
我的言论网站(江边闲话集)
了解作者?
公众号:

昵称: 张太国
园龄: 18年3个月
粉丝: 149
关注: 0
+加关注

深入浅出:HTTP/2

上篇文章深入浅出:5G和HTTP里给自己挖了一根深坑, 说是要写一篇关于HTTP/2的文章, 今天来还账了。

本文分为以下几个部分:

1. HTTP/2的背景
2. HTTP/2的特点
3. HTTP/2的协议分析
4. HTTP/2的支持

HTTP/2简介

HTTP/2主要是为了解决现HTTP 1.1性能不好的问题才出现的。当初Google为了提高HTTP性能, 做出了SPDY, 它就是HTTP/2的前身, 后来也发展成为HTTP/2的标准。

HTTP/2兼容HTTP 1.1, 例如HTTP Method, Status code, URI以及大部分Header Fields。

HTTP/2通过以下方法减少latency, 用来改进页面加载的速度。

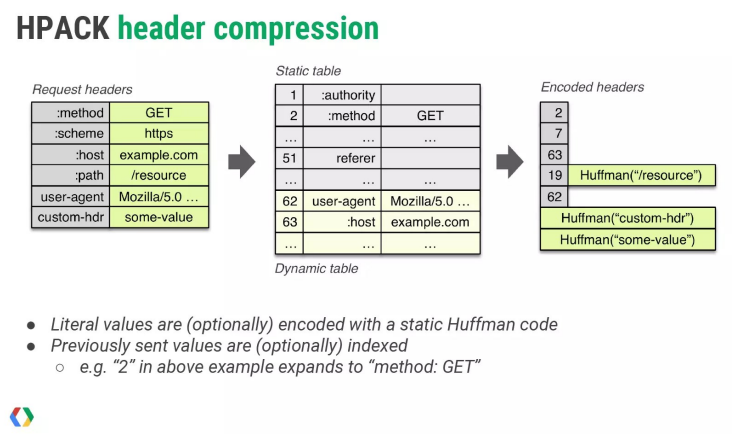
1. HTTP Header的压缩, 采用的是HPack算法。
2. HTTP/2的Server Push, 非常重要的一个特性。
3. 请求的pipeline。
4. 修复在HTTP 1.x的队头阻塞问题。
5. 在单个TCP连接里多工复用请求。

HTTP/2支持HTTP 1.1里的大部分use case, 例如桌面浏览器、移动浏览器、Web API、Web Server、代理服务器、反向代理服务器、防火墙和CDN等。

HTTP/2 头部压缩(HPack)

HPack是HTTP/2 里HTTP头压缩的算法, 具体可以参看https://tools.ietf.org/html/rfc7541。下面简单介绍一下HPack是如何工作的。

见下图, 该图来自Google 的性能专家 Ilya Grigorik 的文章HTTP/2 is here, let's optimize!, 它非常直观地描述了 HTTP/2 中头部压缩的原理:



简单说, HTTP头压缩需要在HTTP/2 Client和服务端之间:

- 维护一份相同的静态表(Static Table), 包含常见的头部名称, 以及特别常见的头部名称与值的组合;
- 维护一份相同的动态表(Dynamic Table), 可以动态地添加内容;
- 基于静态哈夫曼码表的哈夫曼编码(Huffman Coding);

在HTTP头里, 有些key:value是固定, 例如:

```
:method: GET
:scheme: http
```

在编码时, 它们直接用一个index编号代替, 例如***:method:GET***是2, 这些在一个静态表定义。静态表的定义如下, 总共61个Header Name, 点击URL https://tools.ietf.org/html/rfc7541#appendix-A查看所有静态表的定义。

Index	Header Name	Header Value
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
…	…	…
32	cookie	
…	…	…
60	via	
61	www-authenticate	

使用静态表、动态表、以及Huffman编码可以极大地提升压缩效果。对于静态表里的字段, 原来需要N个字符表示的, 现在只需要一个索引即可, 对于静态、动态表中不存在的内容, 还可以使用哈夫曼编码来减小体积。HTTP/2 标准里也给出了一份详细的静态哈夫曼码表(https://tools.ietf.org/html/rfc7541#appendix-B), 它们需要内置在客户端和服务端之中。

关于HPack的算法和实现, 后面专门抽一篇文章来写。

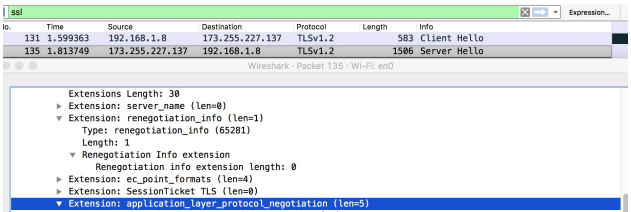
HTTP/2 ALPN

HTTP/2协议里有个negotiation的机制, 让客户端和服务器选择使用HTTP 1.1还是2.0, 这个是由ALPN来实现, 关于ALPN, 可以参看

ALPN(Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension, https://tools.ietf.org/html/rfc7301。

下面是抓包截图, 在TLS里的Client Hello的包里, 我们可以看到ALPN里由H2和HTTP/1.1, 这就是说客户端支持HTTP2以及HTTP 1.1。

当Server收到后, 会识别Client发过来的协议列表, 如果不认识就忽略掉。如果认识多个, 则选择一个最合适的协议发布给Client。也是在Server Hello里的ALPN返回, 见下图。

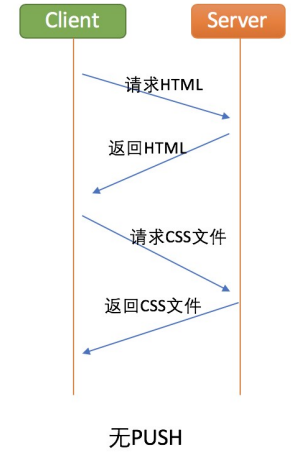


```
Type: application_layer_protocol_negotiation (16)
Length: 5
ALPN Extension Length: 3
▼ ALPN Protocol
    ALPN string length: 2
    ALPN Next Protocol: h2
```

HTTP/2 Server Push机制

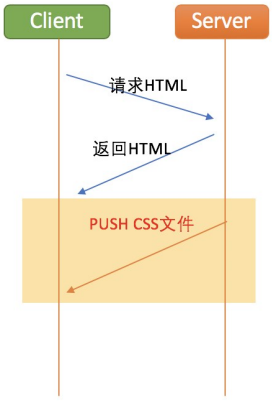
Server Push是HTTP 2最重要的一个特性。

在HTTP 1.1里，在同一个 TCP 连接里面，上一个回应(response)发送完了，服务器才能发送下一个，但在HTTP/2里，可以将多个回应一起发送。



无PUSH

下图是PUSH模式，当请求一个HTML时，如果HTML里有CSS文件，server会一并推给client，而不像在HTTP 1.1下，还需要再发一个CSS的请求。



PUSH模式

根据上图，从理论上PUSH模式下性能会好很多。

举个例子解释一下。下面是一个简单的HTML页面，假设是index.html 。

```
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>This is a sample to illustrate how HTTP/2 works</p>
  <img src="example.png">
</body>
</html>
```

这里有三个文件需要处理：该HTML页面、CSS文件style.css以及图片example.png。在HTTP 1.1里为了处理这三个文件，Client需要发三个请求给Server。

首先，发送一个请求index.html，

```
GET /index.html HTTP/1.1
```

Client解析该HTML文件，继而知道有2个style.css和example.png资源文件下载。

Client继续发送2个请求下载他们。

```
GET /style.css  HTTP/1.1
```

以及

```
GET /example.png  HTTP/1.1
```

一般为了解决这两个问题，像CSS文件，可以把CSS code直接放在HTML里，也可以把example.png转化为base64 code嵌入在HTML里，以上只是把外部资源文件合并到HTML里。

除了上述方法，还有一个优化的方法，就是Preload(预加载)，可以参看这里，https://w3c.github.io/preload/。

所以我们可以把HTML代码改成如下：

```
<link rel="preload" href="/styles.css" as="style">
<link rel="preload" href="/example.png" as="image">
```

那Preload是什么意思呢？就是说下载前一个页面时，可以把相关的资源文件预先加载好，这样感觉起来会快一些。但是有一个关键问题需要注意，**即便是预加载的情况下，也不能减少HTTP请求次数**。

针对上面的问题，我们引出服务器推送(server push)。根据上面的图，我们可以看出，Server还没有收到Client的请求，就把各种资源推送给Client。

拿上面例子继续举例，当Client只请求index.html，但是Server把index.html、style.css、example.png全部发送给浏览器。这样只需要一轮 HTTP 通信，Client就得到了全部资源。

HTTP/2的支持

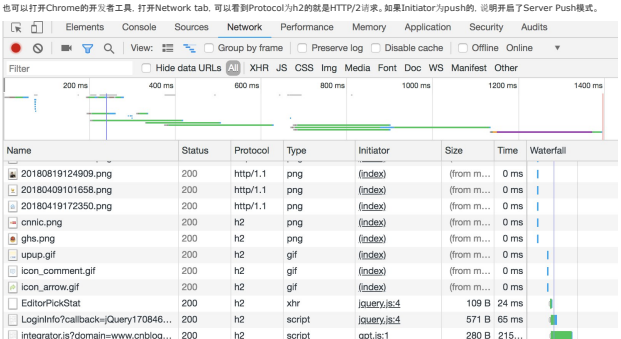
现在主流的软件都支持HTTP/2.

浏览器

基本上大部分浏览器在2015年底都支持HTTP/2了，包括Chrome、Opera、Firefox、IE 11、Safari、Edge。

在Chrome上，可以下载插件**HTTP Indicator**，判断访问的网站是否支持HTTP/2.

也可以打开Chrome的开发者工具，打开Network tab，可以看到Protocol为h2的就是HTTP/2请求。如果Initiator为push的，说明开启了Server Push模式。



pubads_impl_285.js	200	h2	script	gpt.js:1	(from dis…	12 ms	

常用Server软件：

1. Apache HTTPd. 从版本2.4.12开始支持, 通过模块mod_h2来支撑。
2. Apache Tomcat. 从版本8.5开始支持。
3. Jetty从9.3开始支持。
4. Netty从4.1开始。
5. IIS在Win10和WIndows Server 2016支持。
6. Ngnix从1.9.5开始支持HTTP2. 但Server Push功能则在1.13.9才开始。

硬件:

1. Ctrix NetScaler从11.x开始支持
2. F5 BIG-IP从11.6开始。

CDN/Cloud:

1. Akamai
2. AWS
3. Azure
4. Aliyun
5. Tecent Cloud

缓存问题

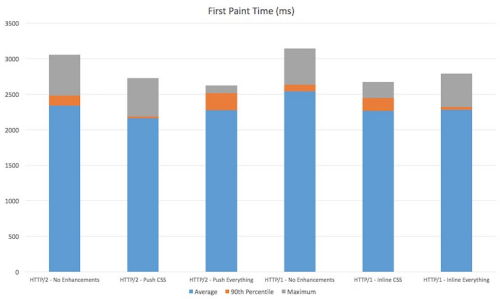
如果开启了Server Push模式, 我们很容易意识到一个问题, 那就是缓存问题。Server见到HTML页面就把外部资源push给Client. 如果没有缓存, 其实很浪费。为了解决这个问题, 可以在第一次请求时push, 后面的请求都不push了。

服务器推送有一个很麻烦的问题。所要推送的资源文件, 如果浏览器已经有缓存, 推送就是浪费带宽。即使推送的文件版本更新, 浏览器也会优先使用本地缓存。下面是 Nginx 官方给出的示例, 根据 Cookie 判断是否为第一次访问(https://www.nginx.com/blog/nginx-1-13-9-http2-server-push/)。

```
server {
    listen 443 ssl http2 default_server;

    ssl_certificate ssl/certificate.pem;
    ssl_certificate_key ssl/key.pem;

    root /var/www/html;
    http2_push_preload on;

    location = /demo.html {
        add_header Set-Cookie "session=1";
        add_header Link $resources;
    }
}

map $http_cookie $resources {
    "~*session=1" "";
    default "</style.css>; as=style; rel=preload, </image1.jpg>; as=image; rel=preload, </image2.jpg>; as=image; rel=preload";
}
```

HTTP/2的性能

有人专门做过测试, https://www.smashingmagazine.com/2017/04/guide-http2-server-push/#measuring-server-push-performance. 借用该文的一张图片,



可以看出, 启用HTTP/2后性能并未大幅度提升, 所以在使用HTTP/2还是谨慎一些, 如果使用不当, 反而会使性能下降。

另外, Ngnix专门撰文描述了7个提高HTTP/2的技巧https://www.nginx.com/blog/7-tips-for-faster-http2-performance/ 。

参考文章:

1. https://en.wikipedia.org/wiki/HTTP/2
2. https://tools.ietf.org/html/rfc7301
3. https://tools.ietf.org/html/rfc7541 (HPack)
4. http://www.ruanyifeng.com/blog/2018/03/http2_server_push.html
5. https://www.nginx.com/blog/nginx-1-13-9-http2-server-push/
6. https://www.smashingmagazine.com/2017/04/guide-http2-server-push/#measuring-server-push-performance
7. https://www.nginx.com/blog/7-tips-for-faster-http2-performance/
8. https://w3c.github.io/preload/
9. http://velocityconf.com/devops-web-performance-2015/public/schedule/detail/42385

分类: 软件工程

张太国
粉丝 - 149 关注 - 0

« 上一篇: 深入浅出 5G和HTTP
» 下一篇: 深入浅出Tomcat/1- 来历和配置文件

posted @ 2018-12-21 09:51 张太国 阅读(45267) 评论(3) 编辑 收藏 举报