HTTP/2:
- protocol overview
- use cases
- benchmarks

Nick Shadrin / NGINX
nick@nginx.com
@shadrin
About me

• Nick Shadrin

• Technical Solutions Architect with NGINX

• Based in San Francisco

• 15 years experience with web tech

• nick@nginx.com / @shadrin
Links

All links in one page:

shadrin.org/talks/
Agenda

• Protocol overview
• HTTP/1 and HTTP/2 optimizations
• Troubleshooting
• Benchmarks
• Use of HTTP/2 with NGINX
• Conclusions
# HTTP history

<table>
<thead>
<tr>
<th>Year</th>
<th>Version</th>
<th>RFC#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td>1.0</td>
<td>1945</td>
</tr>
<tr>
<td>1999</td>
<td>1.1</td>
<td>2616 and 7230 .. 7235</td>
</tr>
<tr>
<td>2015</td>
<td>2</td>
<td>7540</td>
</tr>
</tbody>
</table>

see Wikipedia

@shadrin | nick@nginx.com
HTTP/1.1 301 Moved Permanently

Server: nginx/1.9.9
Date: Tue, 19 Jan 2016 00:19:07 GMT
Content-Type: text/html
Content-Length: 184
Connection: close
Location: https://example.com/test
SPDY

- Announced in 2009 by Google
- Since then implemented in all major browsers
- Major goal: reduce page load time
- Major performance enhancements:
  - Compressed headers
  - Flow control
  - Server Push
HTTP/2 overview

• Introduced in 2015 as a standard

• Based on SPDY

• Includes major changes compared to HTTP/1:
  - Binary headers with HPACK
  - Multiple streams
  - Prioritization
  - Server Push

@shadrin | nick@nginx.com
Encryption

• Is encryption required?
  • Protocol spec: "No"
  • Browser vendors: "Yes"
Protocol negotiation

- Upgrade header
- NPN
- ALPN
NPN

• Next protocol negotiation
• Server lists the protocols
• Client picks one
• Google Chrome plans to stop supporting it soon
ALPN

- Application level protocol negotiation
- Client lists the protocols
- Server picks one
- Results in fewer round trips
- Available in openssl 1.0.2
Revise your optimizations

• Domain sharding
• Image sprites
• Concatenating code files
Domain Sharding

• Browser opens 6 connections to the host

• Distribute your resources through multiple domains

• Does it help when you use HTTP/2? - No.
Image Sprites

- Aggregate multiple images in a single file
- Separate images on the client side
- Does it help when you use HTTP/2? - *Somewhat.*
Concatenating code files

• Combine JS and CSS into larger files

• Does it help when you use HTTP/2? - Not significantly.
Revise your optimizations

• Domain sharding
• Image sprites
• Concatenating code files
• All these optimizations add to the management overhead.

@shadrin | nick@nginx.com
HTTP/2 stats today

- Is it time already to use HTTP/2?
  Let's look at caniuse.com
HTTP/2 protocol

Networking protocol for low-latency transport of content over the web. Originally started out from the SPDY protocol, now standardized as HTTP version 2.

HTTP/2 usage (caniuse.com)
@shadrin | nick@nginx.com
HTTP/2 protocol

A networking protocol for low-latency transport of content over the web. Originally started out from the SPDY protocol, now standardized as HTTP version 2.

HTTP/2 usage (caniuse.com)

@shadrin | nick@nginx.com
HTTP/2 usage (w3techs.com)

@shadrin | nick@nginx.com
Downsides

• Greater overhead for the single connection.

• You might not need SSL.

• HTTP/1.x optimizations hurt.

• Big downloads don’t benefit.

• Your customers may not care.
HTTP/2 is unreadable

@shadrin | nick@nginx.com
Troubleshooting HTTP/2 with Wireshark

1. Set the key log file ENV variable:
   export SSLKEYLOGFILE=/Users/path/file.key

2. Open your browser:
   open -a Google\ Chrome

3. Set the key file in Wireshark
HyperText Transfer Protocol 2

Stream: HEADERS, Stream ID: 3, Length 131

- Length: 131
- Type: HEADERS (1)
- Flags: 0x04
  - .... ....0 = End Stream: False
  - .... .1.. = End Headers: True
  - .... 0... = Padded: False
  - ..0. .... = Priority: False
  - 00.0 ..0. = Unused: 0x00

0... ..... ..... ..... ..... ..... = Reserved: 0x00000000
.000 0000 0000 0000 0000 0000 0000 0000 0011 = Stream Identifier: 3
[Pad Length: 0]

Header Block Fragment: 88760b6e67696e782f312e392e34611d4d6f6e2c20313820...
[Header Length: 217]

- Header: :status: 200
- Header: server: nginx/1.9.4
- Header: date: Mon, 18 Jan 2016 23:51:37 GMT
- Header: content-type: text/css
- Header: last-modified: Sun, 19 Jul 2015 18:00:27 GMT
- Header: etag: W"55abe5bb-df"
- Header: content-encoding: gzip

Padding: <MISSING>
Benchmarks
Benchmarks from nginx.conf 2015

Test Environment

Hardware: Intel Core i7-4770S, 16Gb of RAM, no disk I/O was involved
Kernel: Linux 4.0.9-gentoo
Network: loopback, 1400 MTU, netem
Server: nginx 1.9.5
Client: Chromium 45.0.2454.85 (64-bit) via Selenium WebDriver

The results were analysed using minisstat:
http://www.freebsd.org/cgi/man.cgi?query=minisstat

Please note, that absolute numbers are irrelevant. Look at the trend.
Benchmark setup

- NGINX 1.10.0
- Ubuntu 16.04
- Openssl 1.0.2
- Chrome
- Measuring full page reload
Test 1

- HTTPS with NO keepalive (worst setup)
- Plain HTTP
- HTTP/2
Test 2

- HTTPS with keepalive
- Plain HTTP with keepalive
- HTTP/2
Test 3

- Plain HTTP
- HTTP/2
Some numbers

- 40ms / 50 objects:
  - HTTP/1: 510ms
  - HTTP/2: 250ms
  ~2 times faster

- 200ms / 100 objects:
  - HTTP/1: 4.0s
  - HTTP/2: 1.1s
  ~4 times faster
Benchmark Conclusions

• Know what exactly you are measuring
  • One object or fast network = minimal impact
• Full page load - HTTP/2 is three to four times faster
  • on high latency network
  • with more complex page
• Long latency affects HTTP/1 more than HTTP/2
• Page complexity affects HTTP/1 more than HTTP/2
NGINX support for h2

user@server$ ./configure --with-http_v2_module --with-http_ssl_module
[...]
user@server$ nginx -V
nginx version: nginx/1.9.9
built by gcc 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04)
built with OpenSSL 1.0.1f 6 Jan 2014
TLS SNI support enabled
configure arguments: --with-http_v2_module --with-http_ssl_module
NGINX config for h2

server {
  listen 443 ssl http2;
  server_name .example.com;
  ssl_certificate /etc/nginx/ssl/example.com.crt;
  ssl_certificate_key /etc/nginx/ssl/example.com.key;
  ssl_protocols TLSv1.2;
  root /data/example.com;
  location / {
    proxy_pass http://backend.example.com/;
  }
}

@shadrin | nick@nginx.com
NGINX logs for h2

• $request = GET /url HTTP/2.0

• 10.2.2.2 - - [18/Jan/2016:16:51:40 -0800] "GET / HTTP/2.0" 200 3470 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36"
NGINX Amplify for monitoring
How to contribute

• hg.nginx.org
• github.com/nginx
• wiki.nginx.org
• nginx.org/mailman
• www.nginx.com/developer-license
  Password is OSCON16

@shadrin | nick@nginx.com
Take a picture

All links used in this talk:
https://shadrin.org/talks/

**NGINX conference** in Austin this September!
Promo code **NG16NS25** for 25% off

Get Started with NGINX Plus
Free Developer License. Code: **OSCON16**
www.nginx.com/developer-license/

Stickers and t-shirts are at the NGINX booth in the Expo Hall

@shadrin | nick@nginx.com