A nighttime photograph of a cityscape featuring a prominent brick tower with a spire on the left, a multi-arched bridge with lights in the middle, and a river with light trails in the foreground. The scene is illuminated by city lights and a twilight sky.

HTTP/2, HTTP/3 and SSL/TLS

State of the Art in Our Servers

Jean-Frederic Clere

What I will cover

- HTTP/3
- HTTP/2
 - HTTP/2 and ALPN
- Servers
 - Apache HTTPD
 - Tomcat
 - Traffic server
- Demos
- Questions?

Who I am

Jean-Frederic Clere

Red Hat

Years writing JAVA code and server software

Tomcat committer since 2001

Doing OpenSource since 1999

Cyclist/Runner etc

Lived 15 years in Spain (Barcelona)

Now in Neuchâtel (CH)

APACHECON Europe

HTTP/3 (March 2018, in progress!)

- Use QUIC / TLS-1.3 / UDP
- To transport HTTP like HTTP/2
- Initial connection TCP + Alt-Svc
 - Response Alt-Svc: h3=":56666":
- problems:
 - UDP ports closed
 - UDP slower than TCP in Kernels
 - Needs extra CPU (?)
- Specifications:
 - Several RFC (6 RFC)

HTTP/3 implementations:

- Curl (http3-direct and Alt-Svc way)
- Libraries
 - ngtcp2
 - quiche
- Apache Tomcat: Problem UDP socket API incomplete
- Apache HTTPD: need time probably like http/2
- Traffic Server: planed for the next version (end 2019)
 - See ATS proto / curl demo

HTTP/3 more info:

- Demo ATS / curl
- Playing browsers:
 - test browser
 - Interop matrix
 - chrome activation

Why HTTP/2

- HTTP/1.1: June 1999 (RFC 2616)
 - 1999:
 - 1 page ~ 1kB HTML
 - 2019:
 - 1 page ~ 3MB HTML + IMAGES + JS + CSS etc
- Protocol:
 - Not adapted / inefficient / etc

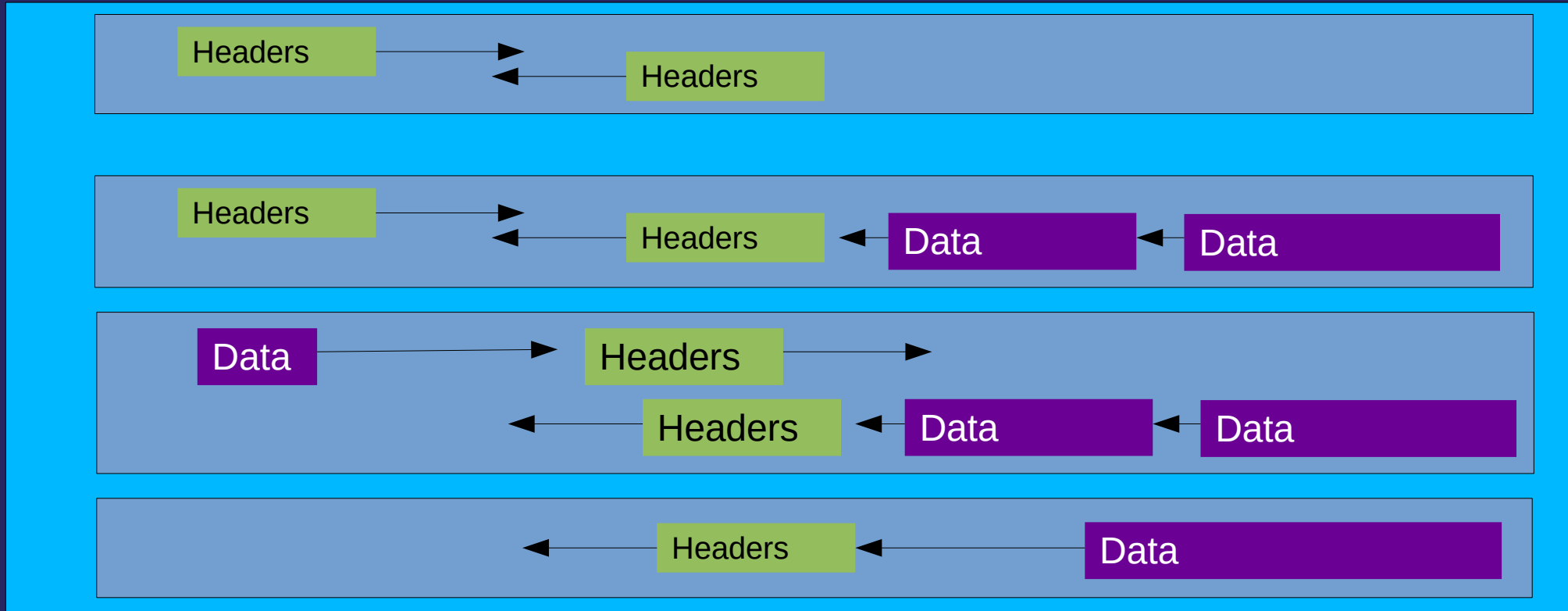
HTTP/2 general

- HTTP/2:
 - Binary
 - Frame
 - Multiplex
 - Based on SPDY
 - TLS everywhere:
 - Browsers use https and strong ciphers
 - No forward proxy
 - h2c: Clear text only with reverse proxy (proxy to back-end server)

HTTP/2 general

- Two specifications:
 - Hypertext Transfer Protocol version 2 - RFC7540
 - HPACK - Header Compression for HTTP/2 - RFC7541
- By the Internet Engineering Task Force
- ALPN Application-Layer Protocol Negotiation - RFC 7301

HTTP/2 Multiplexed



HTTP/2 : more

- HTTP headers compression
 - ~ 80 % save
- Request priority
 - Both sides
- Server Push
 - Prevent round trip to get element of a page
 - Faster / better rendering on browsers.

HTTP/2 With Browsers

- Browser with HTTP/2 and TLS
 - FireFox 34
 - Chrome 40 (with ALPN before was NPN)
 - IE 11
 - Opera and Safari 9
- Stats from docs.trafficserver and ci.trafficserver:
 - 80% is over HTTP/2 (data from last year)
- → go for it now!

ALPN Client Hello (Firefox)

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	:::1	:::1	TCP	94	46254→8443 [SYN]
2	0.0000320000	:::1	:::1	TCP	94	8443→46254 [SYN,
3	0.0000490000	:::1	:::1	TCP	86	46254→8443 [ACK]
4	0.0003110000	:::1	:::1	TLSv1.2	603	Client Hello
5	0.0003210000	:::1	:::1	TCP	86	8443→46254 [ACK]
6	0.0010060000	:::1	:::1	TLSv1.2	232	Server Hello, Cha
7	0.0010190000	:::1	:::1	TCP	86	46254→8443 [ACK]
8	0.0012570000	:::1	:::1	TLSv1.2	137	Change Cipher Spe
9	0.0014710000	:::1	:::1	TLSv1.2	243	Application Data
10	0.0014940000	:::1	:::1	TLSv1.2	318	Application Data
11	0.0018590000	:::1	:::1	TLSv1.2	130	Application Data
12	0.0019060000	:::1	:::1	TLSv1.2	124	Application Data
13	0.0030900000	:::1	:::1	TLSv1.2	124	Application Data
14	0.0031280000	:::1	:::1	TLSv1.2	122	Application Data

ALPN Extension Length: 39

- ALPN Protocol
 - ALPN string length: 5
 - ALPN Next Protocol: h2-16
 - ALPN string length: 5
 - ALPN Next Protocol: h2-15
 - ALPN string length: 5
 - ALPN Next Protocol: h2-14
 - ALPN string length: 2
 - ALPN Next Protocol: h2
 - ALPN string length: 8
 - ALPN Next Protocol: spdy/3.1
 - ALPN string length: 8
 - ALPN Next Protocol: http/1.1

Extensions: status request

ALPN Server Hello (tomcat)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	:::1	:::1	TCP	94	46254→8443 [SYN] Seq=0 Win=
2	0.000032000	:::1	:::1	TCP	94	8443→46254 [SYN, ACK] Seq=0
3	0.000049000	:::1	:::1	TCP	86	46254→8443 [ACK] Seq=1 Ack=1
4	0.000311000	:::1	:::1	TLSv1.2	603	Client Hello
5	0.000321000	:::1	:::1	TCP	86	8443→46254 [ACK] Seq=1 Ack=5
6	0.001006000	:::1	:::1	TLSv1.2	232	Server Hello, Change Cipher
7	0.001019000	:::1	:::1	TCP	86	46254→8443 [ACK] Seq=518 Ack=
8	0.001257000	:::1	:::1	TLSv1.2	137	Change Cipher Spec, Hello Re
9	0.001471000	:::1	:::1	TLSv1.2	243	Application Data
10	0.001494000	:::1	:::1	TLSv1.2	318	Application Data
11	0.001859000	:::1	:::1	TLSv1.2	130	Application Data
12	0.001906000	:::1	:::1	TLSv1.2	124	Application Data
13	0.003090000	:::1	:::1	TLSv1.2	124	Application Data
14	0.003128000	:::1	:::1	TLSv1.2	122	Application Data

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Compression Method: null (0)
Extensions Length: 14

- Extension: renegotiation_info
 - Type: renegotiation_info (0xff01)
 - Length: 1
 - Renegotiation Info extension
- Extension: Application Layer Protocol Negotiation
 - Type: Application Layer Protocol Negotiation (0x0010)
 - Length: 5
 - ALPN Extension Length: 3
 - ALPN Protocol
 - ALPN string length: 2
 - ALPN Next Protocol: h2

Requirements

- OpenSSL for our 3 servers
 - At least 1.0.2c
- Tomcat (8.5 / trunk)
 - Tomcat-native (1.2.6 / trunk) or java9
- Httpd (2.4.17 / trunk)
 - HTTP/2 C Library (libnghttp2)
- TrafficServer (since ATS v5.3.2).
 - Nothing except openssl.

Status

- Tomcat (trunk/8.5)
 - Full support / released as stable.
 - Needs servlet 4.0 (JSR 369) for server PUSH API
 - Can't be full JAVA until JDK9 (ALPN support)
- Httpd (available since 2.4.17)
 - Full support (since 2.4.20)
- TrafficServer (since 5.3.0) (flow control 6.1)
 - Priorities (6.2.0) and Server PUSH (7.0.0)

TC connector server.xml

```
<Connector
  port="8002"
  scheme="https"
  SSLEnabled="true"
  ciphers="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"
  SSLCertificateFile="/home/jfclere/CERTS/newcert.pem"
  SSLCertificateKeyFile="/home/jfclere/CERTS/newkey.txt.pem"
  protocol="org.apache.coyote.http11.Http11AprProtocol">
  <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
```

```
<Connector port="8003" protocol="HTTP/1.1"
  SSLEnabled="true" scheme="https" secure="true"
  keystoreFile="conf/.keystore" keystorePass="changeit"
  socket.directBuffer="true" socket.directSslBuffer="true">
  <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
```

Tomcat / configuration

In bin/setenv.sh:

```
LD_LIBRARY_PATH=/home/jfclere/tomcat-native/native/.libs
```

```
export LD_LIBRARY_PATH
```

And the libtcnative-1.so linked with openssl-1.0.2c, checking with ldd:

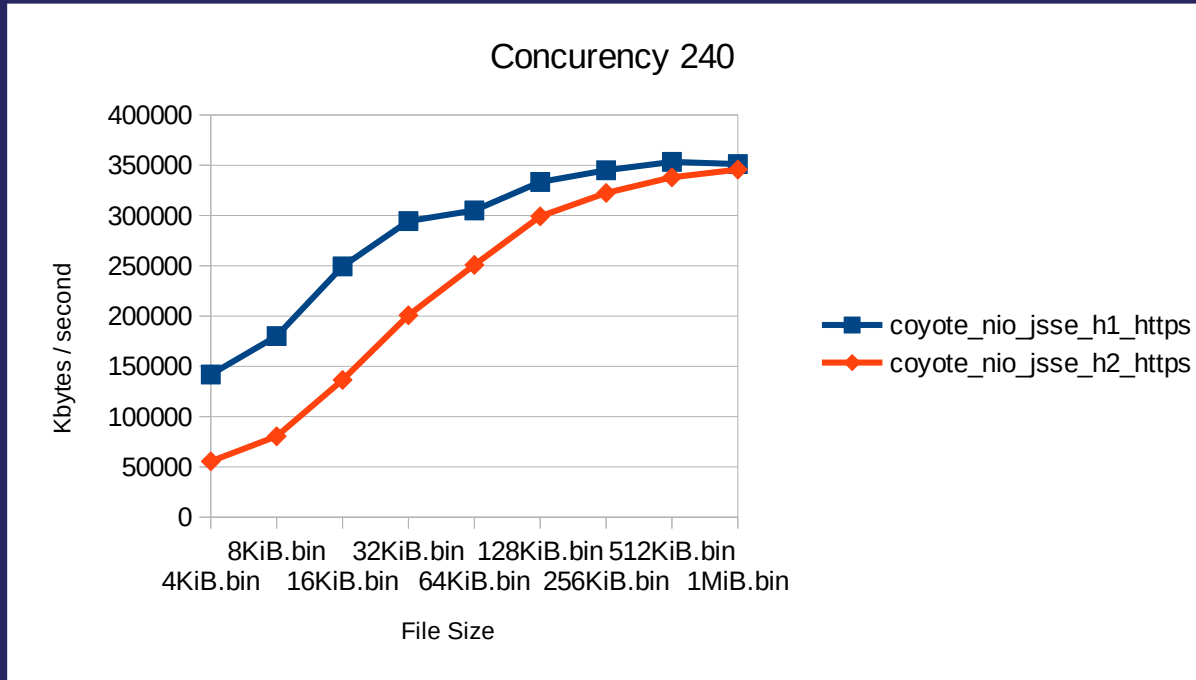
```
libssl.so.1.0.0 => /home/jfclere/OPENSSL-1.0.2c/lib/libssl.so.1.0.0 (0x00007f6ab147b000)
```

```
libcrypto.so.1.0.0 => /home/jfclere/OPENSSL-1.0.2c/lib/libcrypto.so.1.0.0 (0x00007f6ab1028000)
```

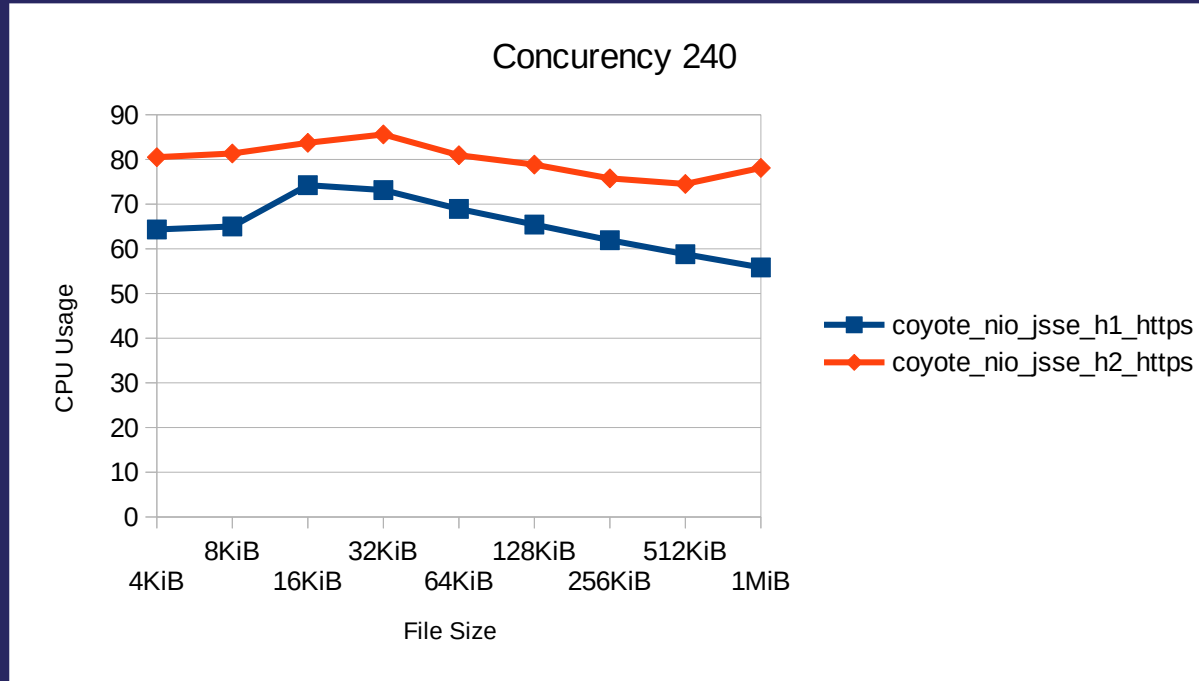
```
libapr-1.so.0 => /home/jfclere/APR-1.4.x/lib/libapr-1.so.0 (0x00007f6ab0dfa000)
```

Usually the openssl of recent distribution (fedora 23) will work.

Tomcat / Performances



Tomcat / Performances



Tomcat / Demo

- No server push (may be change it: SimpleImagePush)
- Multiplexing
- headers compression
- HTML page:
 - That requires a lot (~1000) of (~4Kbytes) images to render.

TrafficServer / Configuration

- records.config
 - CONFIG proxy.config.ssl.number.threads INT 0
 - **CONFIG proxy.config.http.server_ports STRING 8888:ssl**
 - CONFIG proxy.config.url_remap.pristine_host_hdr INT 1
 - CONFIG proxy.config.http2.enabled INT 1
 - CONFIG proxy.config.ssl.TLSv1_1 INT 1
 - CONFIG proxy.config.ssl.TLSv1_2 INT 1
- ssl_multicert.config:
 - **dest_ip=* ssl_cert_name=newcert.pem ssl_key_name=newkey.txt.pem**
- remap.config:
 - **map / http://127.0.0.1:8080**
- ip_allow.config:
 - src_ip=**192.168.1.38** action=ip_allow method=ALL
 - src_ip=: -ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff action=ip_allow method=ALL

TrafficServer / Demo

- Like tomcat one
- Uses http/1.1 tomcat nio connector on 8080 as back-end.

HTTPd / Configuration

- httpd.conf:

```
LoadModule h2_module modules/mod_h2.so
```

```
Listen 8006
```

```
<VirtualHost *:8006>
```

```
Protocols h2 http/1.1
```

```
ProtocolsHonorOrder on
```

```
SSLEngine on
```

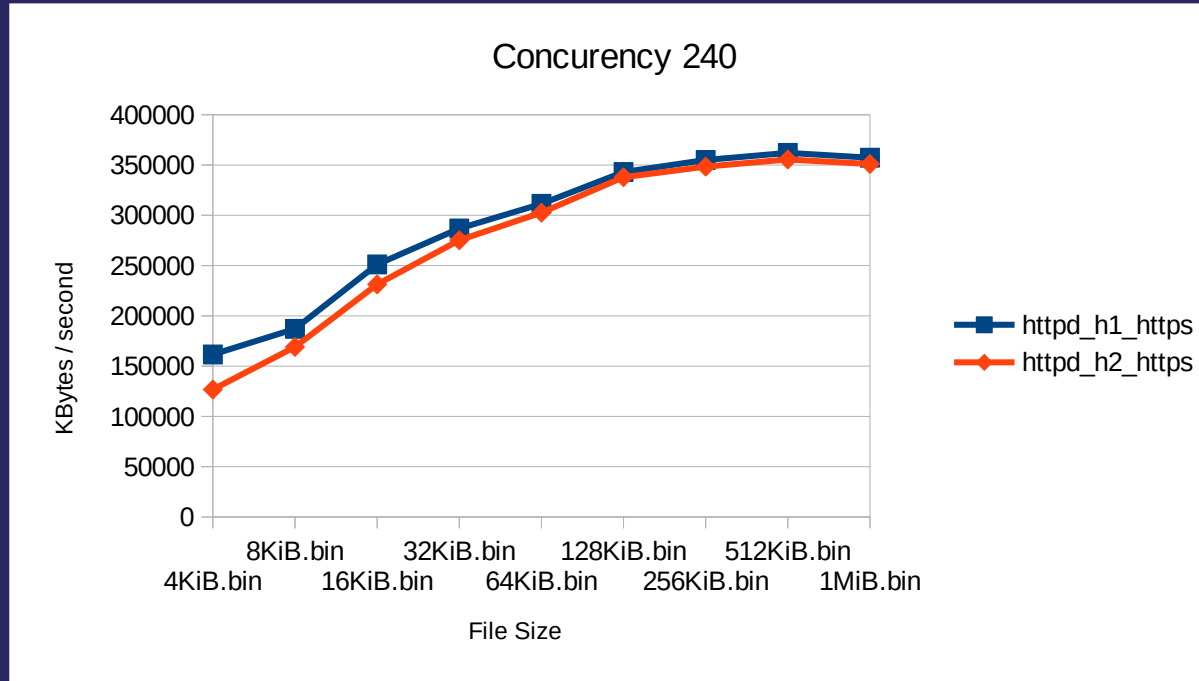
```
SSLCertificateFile "/home/jfclere/CERTS/newcert.pem"
```

```
SSLCertificateKeyFile "/home/jfclere/CERTS/newkey.pem"
```

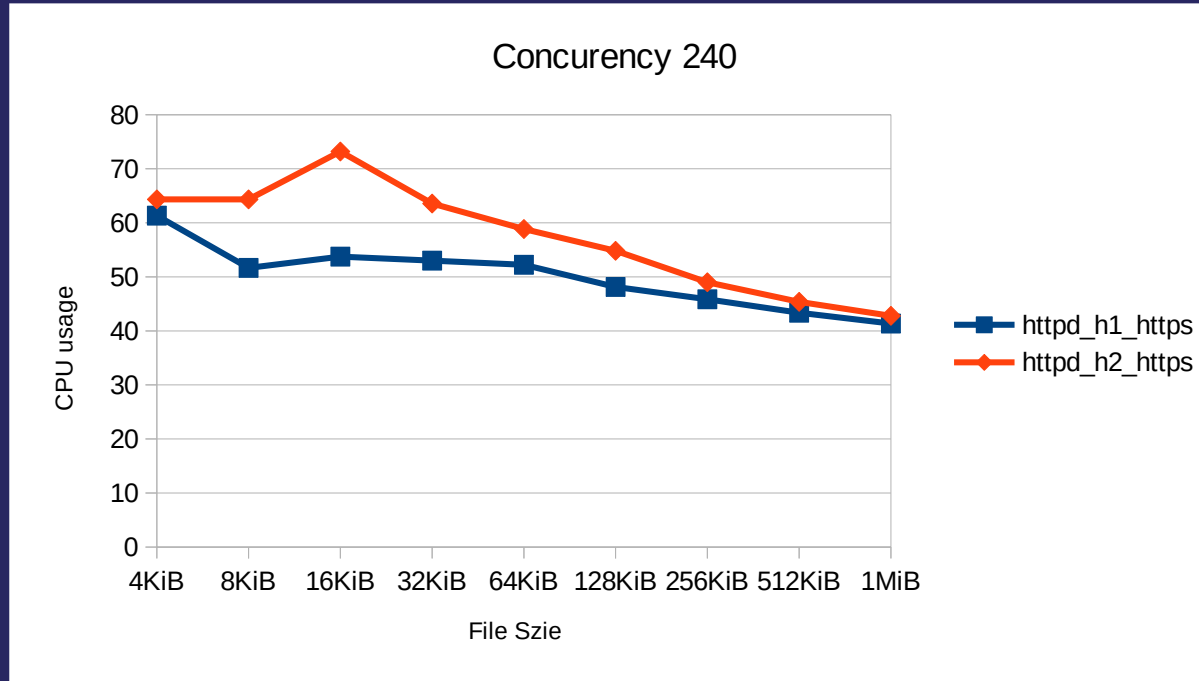
```
SSLCACertificateFile "/etc/pki/CA/cacert.pem"
```

```
</VirtualHost>
```


HTTPd / Performances



HTTPd / Performances



HTTPd / Configuration proxy

- httpd.conf:

```
LoadModule http2_module modules/mod_http2.so
```

```
LoadModule proxy_http2_module modules/mod_proxy_http2.so
```

```
Listen 8006
```

```
<VirtualHost *:8006>
```

```
    Protocols h2 http/1.1
```

```
    ProtocolsHonorOrder on
```

```
    SSLEngine on
```

```
    ...
```

```
    ProxyPass "/" "h2c://localhost:8003/"
```

```
</VirtualHost>
```

HTTPd / Demo

- Like the tomcat one:
 - `htdocs/http2.html`
 - `htdocs/images/` the images.

HTTP/2 ready?

- Conclusion:
 - Using HTTP/2 without PUSH is already good.
 - “safer” crypto is good but expensive.
 - No need to rewrite application to get the gains.

HTTP/2 : GO FOR IT

HTTP/3 ready?

- Conclusion:
 - Still draft protocol H3-23 (at writing).
 - UDP versus TCP.
 - Needs a fork of openssl... (0-RTT).
 - No need to rewrite application to get the gains.

HTTP/3 : WAIT

Questions?

Thank you!

- jfclere@gmail.com
- users@tomcat.apache.org
- users@httpd.apache.org
- users@trafficserver.apache.org
- <https://http2.github.io/>
- Demo generator: https://github.com/jfclere/h2_demos
- HTTP/3 see curl docs: [http3-explained by Daniel](#)

THANK YOU

Jean-Frederic Clere

@jfclere

jfclere@gmail.com