Proxing to tomcat with httpd

Jean-Frederic Clere
Principal Software Engineer / Red Hat
What I will cover

• Proxy what and why.

• Protocols
  – AJP
  – HTTP/HTTPS (1.1)
  – HTTP/2 (H2 and H2C)
  – Others proxy and Other protocols (web-socket etc)

• Configuration
  – mod_jk, mod_proxy, http/1.1 basic, h2c, h2
  – https /TLS proxying

• Demo

• QUESTIONS?

10/12/22
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)
What is Proxy?

- Something between the application server and the internet.
- Load-balancer
- Failover
- Protocol termination
  - TLS/SSL
  - HTTP/2 and (soon) HTTP/3
- Understands a protocol and possible upgrades.
Why a proxy?

- Control the load
- Serve static pages
- Control requests: mod_security / mod_rewrite etc
- Dynamic configuration (mod_balancer/mod_cluster…)
- Protocol translations
AJP

• When
  − Easy TLS/SSL forwarding
• Limitations
  − No upgrade
  − Header size
  − No encryption
  − Limited “authentication” (secret)
• mod_proxy_ajp and mod_jk
HTTP and HTTPS 1.1

• When:
  – No SSL forwarding
  – Using SSLValve

• HTTP/HTTPS:
  – HTTPS might be needed (Encryption/Authentication)
  – HTTPS on tomcat (openssl again?)
  – HTTP if you trust your intranet. (really?)

• Other reasons:
  – HTTP is more developed than AJP
Comparisons mod_jk / mod_proxy

![Comparison Graph](image.png)

File Size

- 4k/IB.bin
- 8k/IB.bin
- 16k/IB.bin
- 32k/IB.bin
- 64k/IB.bin
- 128k/IB.bin
- 256k/IB.bin
- 512k/IB.bin
- 1MB.bin

Concurrency 240

Kbytes / second

- mod_jk
- proxy_aej
- proxy_http11

APACHECON 2022
Comparisons mod_jk / mod_proxy

![Graph showing the comparison of CPU usage for different file sizes (4KiB, 8KiB, 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB). The graph includes three lines representing mod_jk, proxy_ajp, and proxy_http11. The x-axis represents file size, and the y-axis represents CPU usage. The chart highlights the performance difference among the three modules at different file sizes, with mod_jk showing a lower CPU usage in most cases.]

File Size: [4KiB, 8KiB, 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB]

CPU Usage: [0, 20, 40, 60, 80, 100, 120]

Concurrency: 240
Conclusion AJP/HTTP

- No big difference mod_proxy_ajp/mod_jk
- AJP more easy (no Valve needed)
- AJP not encrypted
- AJP has no upgrade
H2C

- h2c is only for reverse proxy
- Supported by httpd
Demultiplexing h2 in proxy

- Keep the back-end unchanged
- Keep the overhead of h2 in the proxy
Other proxies

- HAPerxy (in the cloud / openshift for example)
- mod_cluster (httpd dynamic load balancer)
- Undertow proxy (jboss servlet container)
- Ingress (in kubernetes, well Nginx or GCE)
- Traffic Server
- Nginx
Other protocols

- Jboss-remoting
- Mix HTTP/1.1 websockets
- mod_proxy_wstunnel
- ProxySet "ws://localhost:8080/" upgrade=jboss-remoting
- LoadModule proxy_wstunnel_module
  modules/mod_proxy_wstunnel.so
So proxy or not proxy

• Fail-over : yes
• H2 and old HTTP/1.1 tomcat : yes? Really? Danger?
• Pure java tomcat + TLS/SSL : yes
• Otherwise: Not needed
mod_jk configuration

• Httpd.conf

  LoadModule  jk_module  modules/mod_jk.so
  JkMount  /jkaj/*  worker1
  JkWorkersFile  conf/workers.properties

• properties

  # Define 1 real worker using ajp13
  worker.list=worker1
  worker.worker1.type=lb
  worker.worker1.balance_workers=clusterdev03,clusterdev04
  # Set properties for workers (ajp13)
  worker.clusterdev03.type=ajp13
  worker.clusterdev03.host=192.168.0.130
  worker.clusterdev03.port=8009
  worker.clusterdev04.type=ajp13
  worker.clusterdev04.host=192.168.0.140
  worker.clusterdev04.port=8009
mod_proxy_ajp configuration

• Httpd.conf

LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so

<Proxy balancer://ajp>
#192.168.0.140 192.168.0.130 clusterdev04 / 03
BalancerMember ajp://192.168.0.130:8009
BalancerMember ajp://192.168.0.140:8009
</Proxy>

10/12/22  ProxyPass /tcaj balancer://ajp/tcaj
mod_proxy_httpd configuration

- **Httpd.conf**

  ```
  LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
  LoadModule proxy_module modules/mod_proxy.so
  LoadModule proxy_http_module modules/mod_proxy_http.so
  LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
  LoadModule proxy_balancer_module modules/mod_proxy_balancer.so

  <Proxy balancer://http>
  BalancerMember http://192.168.0.130:8080
  BalancerMember http://192.168.0.140:8080
  </Proxy>
  
  ProxyPass /tchp balancer://http/tchp
  ```
H2C configuration

• Httpd.conf

LoadModule http2_module modules/mod_http2.so

Protocols h2 http/1.1

LoadModule proxy_module modules/mod_proxy.so

LoadModule proxy_http_module modules/mod_proxy_http.so

LoadModule proxy_http2_module modules/mod_proxy_http2.so

ProxyPass "/tch2" "h2c://192.168.100.215:8888/tch2"
H2C configuration

• server.xml

```xml
<Connector port="8888" protocol="HTTP/1.1" redirectPort="8443">
</Connector>
```
Using TLS (1 tomcat)

server.xml

```xml
    address="localhost"
    maxThreads="150" SSLEnabled="true">
    <SSLHostConfig>
        <Certificate
            certificateFile="/home/jfclere/CERTS/localhost/newcert.pem"
            certificateKeyFile="/home/jfclere/CERTS/localhost/newkey.txt.pem"/>
    </SSLHostConfig>
</Connector>
```
Using TLS (1 httpd)

httpd.conf

SSLProxyEngine on
SSLProxyCACertificateFile  "/etc/pki/CA/cacert.pem"

ProxyPass="/examples" "https://localhost:8443/examples"
ProxyPassReverse="/examples" "https://localhost:8443/examples"
Using TLS (2 tomcat)

server.xml

<Valve className="org.apache.catalina.valves.SSLValve" />
Using TLS (2 httpd)

Httpd.conf add headers for Valve

# export the ssl variables
SSLOptions +StdEnvVars +ExportCertData

# Use mod_headers to add them as headers.
RequestHeader set SSL_CLIENT_CERT "%{SSL_CLIENT_CERT}s"
RequestHeader set SSL_CIPHER "%{SSL_CIPHER}s"
RequestHeader set SSL_SESSION_ID "%{SSL_SESSION_ID}s"
RequestHeader set SSL_CIPHER_USEKEYSIZE "%{SSL_CIPHER_USEKEYSIZE}s"
Demo

8007: just encrypt httpd-tomcat (makes no sense!!!).
8000: encrypt both. (makes sense)
8888: encrypt both. (makes sense) and get client certificates. (remember PEM vs pem)
9999: h2c proxy
9998: h2 proxy


Remember: LogLevel proxy_module:debug (or ssl_module:debug) will help!
Questions?

Thank you!

- jfclere@gmail.com
- users@tomcat.apache.org
- Repo with the examples for the demo:
  - https://github.com/jfclere/AC2022