Apache Tomcat & Reverse Proxies

Mark Thomas, Staff Engineer

Agenda

- Introductions
- What is a reverse proxy?
- Protocol selection
- httpd module selection
- Connector selection
- Load-balancing and clustering
- Potential problems
- Questions

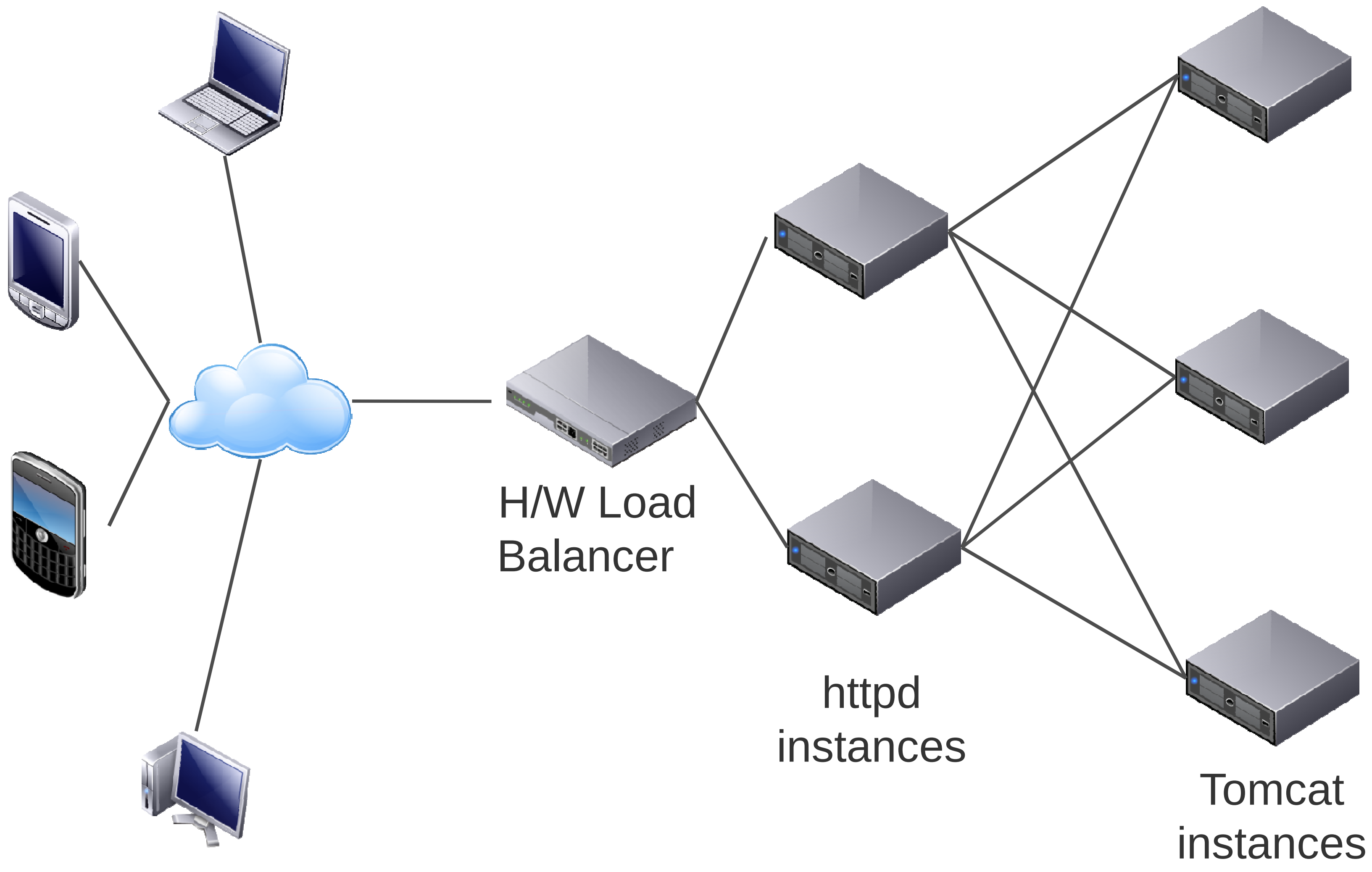
Introductions

Introductions

- **Mark Thomas**
- **Apache Tomcat committer (markt)**
- **Other ASF**
  - Infrastructure team
  - Security
  - Commons
  - Member
- **Staff Engineer at VMware**
  - Tomcat
  - Security
  - tc Server
  - support

What is a reverse proxy?

What is a reverse proxy



Protocol selection

Protocol selection

- **Two options**
  - AJP
  - HTTP
- **Best choice depends on circumstances**
  - No clear winner
- **Both support persistent connections**
  - On a fast LAN or the same machine makes little difference

AJP

- **Not a binary protocol**
  - Common headers and values encoded
  - Other values in plain text
  - Request and response bodies in plain text
- **Request headers must fit in a single AJP message**
  - Default 8192
  - Max 65536
- **Supports passing of SSL termination information**
- **Does not directly support encryption**
  - IPSec, VPN, SSH tunnel, etc.

HTTP

- **Clear text protocol**
  - Easy to read
- **No limit on request header size**
- **Does not directly support providing SSL termination information**
  - Can be added by httpd using custom headers
  - Can be processed by Tomcat using the SSLValve (undocumented)
- **Supports encryption via HTTPS**

AJP vs. HTTP

- **If terminating SSL at httpd and you need the SSL information**
  - Use AJP
- **If you need to encrypt the httpd to Tomcat channel**
  - Use HTTP
- **If you need both**
  - Use HTTP
  - It is (usually) easier to pass SSL information over HTTP than it is to encrypt AJP
- **If you need neither**
  - Pick the one you are more familiar with – debugging problems will be easier

httpd module selection

httpd module selection

- **Avoid**
  - mod_jk2
  - mod_jserv
  - mod_webapp
  - anything else not explicitly mention below

- **Consider**
  - mod_jk
  - mod_proxy
  - (mod_rewrite)

mod_rewrite

- You can replace most of httpd.conf with mod_rewrite directives
- That doesn’t mean that you should
- It is generally more efficient to use the dedicated directive
- There are times (complex load balancing rules) where I’ve used mod_rewrite

mod_jk

- **Only supports AJP**
- **Developed by the Tomcat committers**
  - More frequent releases than httpd
  - Features developed in mod_jk first
- **Non-httpd style configuration**
- **More complex URL mappings are simpler to write**
- **Binaries only provided for Windows**

mod_proxy

- Supports AJP and HTTP
- Included as standard with httpd
- Uses httpd style configuration
- More complex URL mappings are trickier to write
- Binaries provided for most platforms
- mod_proxy_ajp not quite as stable as mod_jk?

mod_jk vs. mod_proxy

- **If you need the latest features**
  - mod_jk

- **If you have complex mapping rules**
  - Consider mod_jk

- **Not on Windows and don’t want to have to compile the module**
  - mod_proxy

- **Already using one of these**
  - Carry on. The costs of changing will probably out-weight the benefits

mod_jk vs. mod_proxy

- **If you have a free choice**
  - Use mod_proxy, the configuration style will be more familiar

Tomcat connector selection

Tomcat connector selection

- **BIO**
  - 100% Java Blocking IO

- **NIO**
  - 100% Java non-blocking IO
    - Waiting for next request
    - Reading HTTP request headers
    - SSL handshake

- **APR/native**
  - Apache APR based native code with JNI providing non-blocking IO
    - Waiting for next request

Tomcat connector selection

- **All connectors block (or simulate blocking) during**
  - Request body read
  - Response body write
- **SSL**
  - BIO & NIO use JSSE
  - APR/native uses OpenSSL
  - OpenSSL is significantly faster
- **Sendfile**
  - NIO and APR/native support sendfile

Tomcat connector selection

- **Comet**
  - NIO and APR/native support Comet

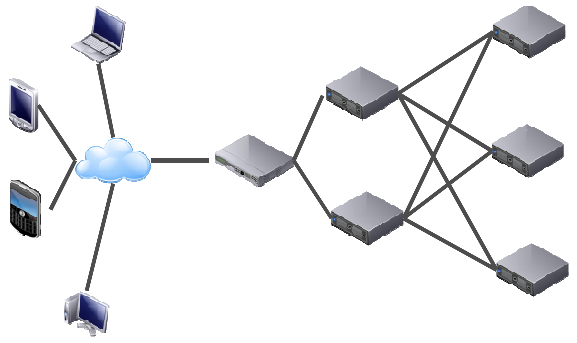
- **WebSocket**
  - All connectors support WebSocket
  - httpd does not support WebSocket when acting as a reverse proxy

BIO vs. NIO vs. APR/native

- **If you use SSL**
  - APR/native
- **Stability**
  - BIO has a slight edge
- **Scalability**
  - NIO or APR/native
- **Need APR/native benefits but with pure Java**
  - NIO

Troubleshooting

Thread exhaustion



- **Need to understand threading models**

- **httpd prefork MPM**
  - 1 thread per process
  - MaxRequestWorkers processes
  - Maximum of 1 * MaxRequestWorkers threads

- **httpd worker MPM**
  - ServerLimit processes
  - ThreadsPerChild threads for each process
  - Maximum of ServerLimit * ThreadsPerChild threads

- **Thread == concurrent request**

Thread exhaustion

- **Each httpd thread may create a connection to each Tomcat instance**
- **Therefore, 2 httpd instances each with 400 threads**
  - Maximum of 800 connections to each Tomcat instance
  - The connections are NOT distributed between the Tomcat instances
  - Connections are persistent by default

- **Connections may have low utilization**
- **BIO requires a thread per connection**
- **BIO connector may run out of threads even when Tomcat is almost idle**

Thread exhaustion

- **Solutions**
  - Use NIO connector as it is non-blocking between requests
  - Don’t use persistent connections between httpd and Tomcat
  - Ensure each Tomcat instance has >= threads than total httpd threads

- **Example**
  - ASF Jira
  - httpd had more threads than Tomcat
  - Didn’t take much load for Tomcat to run out of threads
  - No component was particularly loaded
  - Tomcat, Java, network I/O all blamed
  - 5 second fix (to server.xml to increase the number of threads)
  - (OK, and several minutes for Jira to restart)

Broken links

- **Easiest way to create a lot of hassle for yourself**
  - ProxyPass /foo http://localhost:10180/bar
- **Easiest way to avoid the hassle**
  - ProxyPass /foo http://localhost:10180/foo
- **Don’t change the context path**
- **What can go wrong**
  - Redirects
  - Cookie paths
  - Links
  - Custom headers (e.g. Spring MVC)

Broken links

- **Fixing redirects**
  - Don’t change the context path
  - ProxyPathReverse will fix some but not all HTTP headers

- **Fixing cookie paths**
  - Don’t change the context path
  - ProxyPassReverseCookiePath /bar /foo

- **Fixing links**
  - Don’t change the context path
  - mod_sed, mod_substitute, mod_proxy_html

Broken links

- **Fixing custom headers**
  - Don’t change the context path
  - mod_headers

Security issues

- **Need to be careful when terminating HTTPS at httpd**
- **Tomcat needs to know if request was received over HTTPS**
  - Sessions must not transition from HTTPS to HTTP
  - Cookies created over HTTPS must be marked as secure
- **mod_jk and mod_proxy_ajp just handle this**
- **mod_proxy_http does not**

- **Solutions**
  - Custom headers and the RemoteIpValve
  - Two HTTP connectors
    - HTTP traffic proxied to connector with secure="false"
    - HTTPS traffic proxied to connector with secure="true"

Miscellaneous

- **Virtual host selection**
  - ProxyPreserveHost on
- **Client IP based security**
  - RemoteIpValve

Questions