Apache Tomcat

Tomcat Clustering: Part 1 – Reverse Proxies

Mark Thomas, 15 April 2015
Introduction

• Apache Tomcat committer since December 2003  
  – Markt@apache.org
• Tomcat 8 release manager
• Member of the Servlet, WebSocket and EL expert groups
• Consultant Software Engineer @ Pivotal
• Currently focused on Apache Tomcat 9
Terminology
Reverse Proxy

bz.apache.org
httpd instance

Bugzilla (Main)
(bz.apache.org/bugzilla)
httpd instance

Bugzilla (AOO)
(bz.apache.org/ooo)
httpd instance

Bugzilla (SpamAssassin)
(bz.apache.org/SpamAssassin)
httpd instance
Reverse Proxy

- Looks like a single host to the clients
- Usually multiple hosts
- Different services on different hosts
  - May also be geographically distributed
- Can be used to add features
  - e.g. Use httpd as a reverse proxy for Tomcat to add Windows authentication (no longer necessary)
Reverse Proxies
Agenda

- Protocol selection
- httpd module selection
- Tomcat connector implementation selection
- Troubleshooting
- Questions
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 this makes little difference
Protocol Selection

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
Protocol Selection

AJP

- Supports passing of SSL termination information

- Does not directly support encryption
  - IPSec, VPN, SSH tunnel, etc.

- Supports ping to validate connection status
Protocol Selection

HTTP

• Clear text protocol
  – Easy to read

• No limit on request header size

• No dedicated ping
Protocol Selection

HTTP

• Does not directly support providing SSL termination information
  – Can be added by httpd using custom headers
  – Can be processed by Tomcat using the SSLValve

• Supports encryption via HTTPS
Protocol Selection
AJP vs. HTTP

• If terminating SSL at httpd and you need SSL the information
  – Use AJP

• If you need to encrypt the httpd to Tomcat channel
  – Use HTTP
Protocol Selection

AJP vs. 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

• Avoid
  – mod_jk2
  – mod_jserv
  – mod_webapp
  – anything else not explicitly mention below

• Consider
  – mod_jk
  – mod_proxy
  – (mod_rewrite)
httpd Module Selection

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
httpd Module Selection

mod_rewrite

- mod_jk and mod_proxy can route based on environment variables
- Use mod_rewrite and/or mod_setenvif to determine the routing info
- Set the routing configuration with mod_jk / mod_proxy
httpd Module Selection

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
httpd Module Selection

mod_jk

• Doesn’t directly support URL re-writing

• Make sure you are using the latest documentation  
  – http://tomcat.apache.org/connectors-doc/

• The status worker can be used for monitoring and management
httpd Module Selection
mod_proxy

- Supports AJP and HTTP
- Included as standard with httpd
- Uses httpd style configuration
- More complex URL mappings are trickier to write
- Built-in support for URL re-writing (not all use cases)
- Binaries provided for most platforms
httpd Module Selection

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
httpd Module Selection
mod_jk vs mod_proxy

• If you will be load-balancing
  – mod_jk’s management interface is probably better

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

• If you have a free choice
  – Use mod_proxy, the configuration style will be more familiar
Tomcat Connector Selection

• BIO
  – Default for all version to Tomcat 8
  – Removed from Tomcat 9 onwards
  – 100% Java Blocking IO

• NIO
  – Default from Tomcat 8 onwards
  – 100% Java non-blocking IO
    ▪ Waiting for next request
    ▪ Reading HTTP request headers
    ▪ SSL handshake
Tomcat Connector Selection

- **NIO2**
  - Introduced in Tomcat 8
  - 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
    - Reading HTTP request headers (9.0.x onwards)
Tomcat Connector Selection

• All connectors block (or simulate blocking) during
  – Request body read
  – Response body write

• SSL
  – BIO, NIO & NIO2 use JSSE
  – APR/native uses OpenSSL
  – OpenSSL is significantly faster

• Sendfile
  – NIO, NIO2 and APR/native support sendfile
Tomcat Connector Selection

• Comet
  – Has been removed for Tomcat 9
  – NIO, NIO2 and APR/native support Comet

• WebSocket
  – All connectors support WebSocket
  – httpd does not support WebSocket very well when acting as a reverse proxy
  – BIO fakes the non-blocking support
Tomcat Connector Selection

BIO vs. NIO vs. NIO2 vs. APR/native

- If you use SSL
  - APR/native

- Stability
  - NIO, BIO

- Scalability
  - NIO, NIO2, APR/native
Troubleshooting
Thread Exhaustion

• Need to understand threading models

• httpd prefork MPM
  – 1 thread per process
  – MaxRequestWorkers processes
  – Maximum of 1 * MaxRequestWorkers threads
Troubleshooting
Thread Exhaustion

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

• Thread == concurrent request
Troubleshooting

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
Troubleshooting
Thread Exhaustion

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

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 \( \geq \) threads than total httpd threads
- Configure timeouts
  - I have seen cases where httpd tried to use a timed out connection
- Use distance to create preferred groups
Troubleshooting
Thread Exhaustion: Example

• Reverse proxy for ASF Jira 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 (edit maxThreads in server.xml)
• (OK, and several minutes for Jira to restart)
Troubleshooting

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
Troubleshooting

Broken Links

• Often marketing wants http://name.com rather than http://name.com/app

• Consider a simple redirect from / to /app
  – /app becomes visible to end users once they use the app
  – Much easier to implement and maintain

• Deploy your application as ROOT
  – Use ROOT##label if you need to add a version number or similar
Troubleshooting
Broken Links: What can go wrong

• Redirects
  – Redirect to wrong path

• Cookie paths
  – Cookies are not returned by client

• Links
  – Created for wrong URL

• Custom headers (e.g. Spring MVC)
Troubleshooting
Broken Links: Solutions

• 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
Troubleshooting
Broken Links: Solutions

• Fixing links
  – Don’t change the context path
  – mod_sed, mod_substitute, mod_proxy_html
  – Fragile solution and a significant maintenance overhead

• Fixing custom headers
  – Don’t change the context path
  – mod_headers
Troubleshooting

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
Troubleshooting

Security Issues: Solutions

- Custom headers and the RemotelpValve
- Two HTTP connectors
  - HTTP traffic proxied to connector with secure="false"
  - HTTPS traffic proxied to connector with secure="true"
Troubleshooting

Miscellaneous

- Virtual host selection
  - ProxyPreserveHost on

- Client IP based security
  - RemoteIpValve
Questions