Routing CDN Traffic at Scale Using Apache Tomcat

Jeff Elsloo
elsloo@apache.org
Sr. Principal Engineer, Comcast Cable
Jeffrey_Elsloo@cable.comcast.com
About Me

Joined Comcast in 2008, CDN Engineering in 2013

Led development of CDN components “CCR” and “Rascal”

Member of Apache Traffic Control PMC

Lead engineer for CDN Engineering at Comcast

Motorcycle enthusiast
Content Delivery Networks
improve user experience
and network efficiency
User Interface

Business logic with RESTful API

Collect and aggregate metrics

Monitor CDN health

Route traffic to healthy caches
Request Flow

Cache selection based on:
- Closest cachegroup
- on first miss in CCF, by geo
- cache in cachegroup based on consistent hash on URL and health as reported by Traffic Mon
DNS and Localization

Cache selection based on:
- Closest cachegroup
- by CZF
- or if miss in CZF by geo
- cache in cachegroup based on consistent hash on URL and health as reported by Traffic Mon
Consistent Hashing

The mechanism that provides cache efficiency within a CDN

Created by Daniel Lewin and F. Thomson Leighton at MIT\[1\]

Creators founded Akamai Technologies

Allows $K/n$ rehashed Keys for add/removals of nodes

Minimizes impact to CDN during health events, maintenance, etc

User Interface

Business logic with RESTful API

Collect and aggregate metrics

Monitor CDN health

Route traffic to healthy caches
Traffic Router

Java Application deployed in Tomcat 8.5.x

Horizontally scalable and stateless

DNS authoritative for CDN domain name

Routes traffic over DNS and HTTP using consistent hashing

Consumes health state published by Traffic Monitor

Entry point for all requests into a CDN
Core Features

HTTP
- TLS / SNI
- 302 or JSON
- (Client) Steering
- Dispersion
- Response headers
- Request header logging

Localization
- (Deep) Coverage Zone
- Geolocation by delivery service
- Anonymous proxy blocking
- Configurable

DNS
- DNSSEC
- Configurable TTLs
- Static DNS entries
- “Federation”
- EDNS0 client subnet extensions

Consistent Hashing
Delivery service limits
Bypass destinations
API and metrics

9/27/18
http://trafficcontrol.apache.org
Tomcat Integration
Languid Connector

Delays when sockets are opened by Tomcat

TCP resets are better than timeouts or layer 7 errors

Traffic Router uses JMX MBean to communicate when ready

Connector listens for message to complete startup
Custom Key Manager

Integrates with Traffic Ops RESTful API

No Java keystore required

Seamless deployment of certificates without restarting

Integrates with OpenSSL implementation in Tomcat 8.5
Packaging Tomcat 8.5.x

All ATC components are built for CentOS 7.x

Downloaded and packaged when Traffic Router is built

Application’s Tomcat configuration outside of Tomcat’s defaults

traffic_router requires tomcat, tomcat-native and apr

traffic_router systemd configuration in startup.properties
Tomcat Configuration

...<Connector connectionTimeout="10000" maxThreads="10000" port="80" sendReasonPhrase="True"
        mbeanPath="traffic-router:name=languidState" readyAttribute="Ready" portAttribute="Port"
        protocol="com.comcast.cdn.traffic_control.traffic_router.protocol.LanguidNioProtocol"/>
<Connector connectionTimeout="10000" maxThreads="10000" port="443" sendReasonPhrase="True"
        mbeanPath="traffic-router:name=languidState" readyAttribute="Ready" portAttribute="SecurePort"
        protocol="com.comcast.cdn.traffic_control.traffic_router.protocol.LanguidNioProtocol"
        scheme="https" secure="True" SSLEnabled="True" sslProtocol="TLS" clientAuth="False"
        sslImplementationName="com.comcast.cdn.traffic_control.traffic_router.protocol.RouterSslImplementation"/>
<Connector connectionTimeout="10000" maxThreads="10000" port="3333"
        mbeanPath="traffic-router:name=languidState" readyAttribute="Ready" portAttribute="ApiPort"
        protocol="com.comcast.cdn.traffic_control.traffic_router.protocol.LanguidNioProtocol"/>
...

9/27/18 http://trafficcontrol.apache.org
Tuning

Large minimum and max heap values

G1 garbage collector, lowered heap occupancy percentage

System tuning via sysctl, limits via systemd

Tomcat and application timeouts and thread pools

9/27/18

http://trafficcontrol.apache.org
Traffic Router at work
An Average Day

Over 200 million DNS transactions served or routed to the edge

Over 300 million HTTP transactions routed to the edge

Over 35 PB served, or 1.5 LOCPM at the edge

Over 100 billion edge transactions

Over 1 million edge transactions per second

Over 18 Exabytes (1,000,000,000,000,000,000,000 bytes) since 2012
DNS Decisions

Locate zone
Locate static records, or…
Match Delivery Service
Localize client
Select healthy caches
Fill dynamic zone
Serve response
### DNS Delivery Service

$ dig edge.images.cdn.example.com

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>TTL</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>192.168.12.10</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>192.168.175.10</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>192.168.115.31</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>192.168.10.64</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>192.168.29.16</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>192.168.72.6</td>
</tr>
</tbody>
</table>

...or the same request made from San Francisco:

$ dig edge.images.cdn.example.com

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>TTL</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>10.59.132.53</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>10.18.190.53</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>10.16.119.92</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>10.27.117.38</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>10.29.116.17</td>
</tr>
<tr>
<td>edge.images.cdn.example.com</td>
<td>A</td>
<td>30</td>
<td>10.68.51.89</td>
</tr>
</tbody>
</table>

9/27/18

http://trafficcontrol.apache.org
HTTP Decisions

What type of request?

Match Delivery Service(s)

Localize Client

Select healthy caches

Consistent hash on path

Order Subordinates if necessary

Serve response (302 or JSON)
HTTP Delivery Service
Default Response

> GET /foo.m3u8 HTTP/1.1
> Host: tr.linear.cdn.example.com
>
< HTTP/1.1 302 Found
< Location: http://edge-den-02.linear.cdn.example.com/foo.m3u8
< Content-Length: 0
< Date: Thu, 20 Sep 2018 16:53:57 GMT

...or the same request made from San Francisco:

< HTTP/1.1 302 Found
< Location: http://edge-sfb-10.linear.cdn.example.com/foo.m3u8
< Content-Length: 0
< Date: Thu, 20 Sep 2018 16:57:38 GMT
HTTP Delivery Service
No Redirect Response

> GET /foo.m3u8?format=json HTTP/1.1
> Host: tr.linear.cdn.example.com
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Content-Length: 99
< Date: Thu, 20 Sep 2018 16:59:46 GMT
<
* Connection #0 to host tr.linear.cdn.example.com left intact
{
   "location" : "http://edge-den-02.linear.cdn.example.com/foo.m3u8?format=json"
}
Client Steering Delivery Service

Default Response

HTTP/1.1 302 Found
Access-Control-Allow-Origin: *
Location: https://edge-den-02.linear-a.cdn.example.com/foo.m3u8
Content-Type: application/json
Content-Length: 206
Date: Mon, 17 Sep 2018 16:38:18 GMT

* Connection #0 to host tr.linear.cdn.example.com left intact

{
  "locations": [
    "https://edge-den-02.linear-a.cdn.example.com/foo.m3u8",
    "https://edge-den-20.linear-b.cdn.example.com/foo.m3u8",
    "https://edge-den-02.linear-c.cdn.example.com/foo.m3u8"
  ]
}
Client Steering Delivery Service
No Redirect Response

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 242
Date: Mon, 17 Sep 2018 16:46:13 GMT

* Connection #0 to host tr.linear.cdn.example.com left intact
{
  "locations" : [
    "https://edge-den-02.linear-a.cdn.example.com/foo.m3u8?trred=false",
    "https://edge-den-20.linear-b.cdn.example.com/foo.m3u8?trred=false",
    "https://edge-den-02.linear-c.cdn.example.com/foo.m3u8?trred=false"
  ]
}

9/27/18 http://trafficcontrol.apache.org
Retrospective
http://trafficcontrol.apache.org
The Apache Way

Traffic Control is successful because of the community

Tomcat development community is active and full of talented engineers

Traffic Router benefits from the Tomcat community’s expertise

Traffic Control relies heavily on Traffic Server, another excellent community
Path to Tomcat 8.5

Attended Tomcat TLS talks at ACNA 2017

Met Mark Thomas who demonstrated performance gains

Began development after ACNA 2017

Development completed and deployed to production this summer

1-2 orders of magnitude improvement with TLS traffic
Lessons Learned

Stay as current as possible

TLS client certificate authentication, clientAuth="False"

HTTP reason phrase, sendReasonPhrase="True"

Content-Type encoding, ENFORCE_ENCODING_IN_GET_WRITER=false
Thanks

David Neuman, Comcast, ATC chair

Andrew Schmidt, Comcast

Dewayne Richardson, Comcast, ATC PMC

Mark Thomas and the Apache Tomcat team