Apache Traffic Control & Lua

Matt Mills
Comcast
Who am I & What is this talk about?

- Sr. Systems Engineer - Comcast CDN - Denver, Colorado
- Somewhere in between Dev and Ops
- How I want to use Lua within Traffic Control
- Why I want to build a generic layer for describing programmatic functionality
- Some tangentially related problems
Traffic Control Refresher
Problem:

1. UI, database schema, and API all speak ATS
   a. Self service users don’t understand regex or header rewrite or...
   b. Can’t leverage non-ATS caching proxy without a LOT of development
2. Adding new custom delivery service functionality requires touching too many components

Related problems:

1. CDN changes propagate too slowly and can cause a thundering herd.
2. Changes are not atomic (Queued updates & Snapshot CRConfig)
   a. Different users’ changes can collide
   b. User A can push out User B’s changes before User B is ready
How changes happen in ATC

Make changes in UI

Save changes to API

Database

Per Server
Queue Updates ~20 min

ATS Mid Tier ~20 min

ATS Edge Tier

Per CDN
Snapshot CRCConfig

Traffic Monitor

Traffic Router
Server Config

GET /api/1.2/servers/example_host/configfiles/ats/remap.config
GET /api/1.2/servers/example_host/configfiles/ats/parent.config
GET /api/1.2/servers/example_host/configfiles/ats/hosting.config
GET /api/1.2/profiles/example_profile/configfiles/ats/records.config
GET /api/1.2/profiles/example_profile/configfiles/ats/logs_xml.config

Delivery Service Config (Per Delivery Service)

GET /api/1.2/profiles/example_profile/configfiles/ats/url_sig_example_com.config
GET /api/1.2/profiles/example_profile/configfiles/ats/regex_remap_example_com.config
GET /api/1.2/cdns/example_cdn/configfiles/ats/hdr_rw_example_com.config
GET /api/1.2/cdns/example_cdn/configfiles/ats/cacheurl_example_com.config
GET /api/1.2/cdns/example_cdn/configfiles/ats/cachekey_example_com.config

Traffic Monitor / Traffic Router Config

GET /api/1.2/cdns/example_cdn/configs/monitoring.json
GET /CRConfig-Snapshots/title-vi/CRConfig.json
<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Header Rewrite Rules</td>
<td>set-header Accept-Encoding None [L]</td>
</tr>
<tr>
<td>Mid Header Rewrite Rules</td>
<td></td>
</tr>
<tr>
<td>Traffic Router Additional Response Headers</td>
<td></td>
</tr>
<tr>
<td>Traffic Router Log Request Headers</td>
<td></td>
</tr>
<tr>
<td>Regex remap expression</td>
<td>^/([^/]+)/(.*$) https://$1.example.com/$2</td>
</tr>
<tr>
<td>Cache URL expression</td>
<td></td>
</tr>
<tr>
<td>Raw remap text</td>
<td>@plugin=tslua.so @pparam=/opt/trafficserver/etc/trafficserver/test.lua</td>
</tr>
</tbody>
</table>
What does “custom delivery service functionality” mean?
Support in-url range requests

- Customer had legacy clients that couldn’t support Range: header
- But also clients that DO support and use Range header
- So they put it in the URL ( /path/to/file/range/100-200 )
- Need to parse in-URL range request when present, and convert to normal Range header at the edge
- 11 lines of Lua
Manipulate URLs in flight to “capture” sessions

- Customer wanted our CDN to use another CDN as upstream
- But they can’t change the URLs that they use for playback (streaming video)
- Also, they didn’t want us to talk to the other CDN about it
- Need to manipulate API responses as well as HLS m3u8 manifests
- Replace existing URLs with our CDN edge URL
- 9 Lines of Lua
Speed test (mod_hotair/generator at the edge)

- Generate hot air (bytes) at the edge, no upstream request
- Variable size (1 byte to 1 gigabyte)
- Should respond with 200 OK to POST (upload speed test)
- Upstream requests to other URLs should still work
- Preferably doesn’t crash server or leak memory
- 57 lines of Lua
AWS S3 v4 Signing

- Before the existing ATS s3_auth plugin was updated to support v4 we had a customer requirement to support upstream auth against AWS S3 v4 signing
- Didn’t need to support full feature set, just sign GET requests
- LUA OpenSSL binding library
- Now we can do HMAC SHA256 in Lua!
- 109 Lines of Lua
Simpler stuff

- Add/Remove/Modify headers
- Manipulate parent or origin URL / URI / Query Params / Scheme
- Manipulate parent or origin hostname
- Enabling debug logging
- Config overrideables per delivery service
- A lot of the existing functionality of:
  - Header_rewrite
  - Cache URL / Cache Key
  - Regex remap
What do I want to do?

LUA all the things!
What do I want to do (cont’d)

- Create a structured way to describe programmatic functionality of a Delivery Service (rules)
- Create a Lua plugin for ATS that interprets & executes that structure, but also allows custom expansion
- Replace existing ATS specific database & API elements in Traffic Control
- Add a web UI for manipulating these rules easily to existing Traffic Portal
- Add an administrative UI for managing the defined components of a rule (conditions and actions) as well as adding custom functionality and defining who can access it
<table>
<thead>
<tr>
<th>Type</th>
<th>Target</th>
<th>Operator</th>
<th>Value</th>
<th>Arguments (type specific)</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_internal_request</td>
<td>client_request</td>
<td>==</td>
<td>User defined</td>
<td>User defined</td>
</tr>
<tr>
<td>url</td>
<td>server_request</td>
<td>!=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header</td>
<td>client_response</td>
<td>regex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>incoming_port</td>
<td>server_response</td>
<td>store</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheme</td>
<td></td>
<td>starts_with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>client_request</td>
<td>ends_with</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>store</td>
<td>contains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>url</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>client_request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>store</td>
<td>variable1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Target</td>
<td>Arguments (type specific)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------</td>
<td>---------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_header</td>
<td>client_request</td>
<td>User defined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>remove_header</td>
<td>server_request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_response</td>
<td>client_response</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_uri</td>
<td>server_response</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_uri_args</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_uri_params</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_url_host</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_url_scheme</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>custom_thing_here</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why?

- Moves most ATS specific functionality out of the core of Traffic Control
  - Simpler to upgrade ATS
- Rules functionality can be re-implemented or translated to existing config language on other caches (nginx, varnish, homebrew_cache.exe)
- Allows self-service users (of a CDN) to access a lot more functionality
- Can build simple and “advanced” rules UIs for easier use
- Adding new custom (lua based) functionality is as simple as:
  - Write Lua code & check into CI/CD system (should run tests & deploy)
  - Add newly defined condition or action to admin UI, configure permission levels
  - Users with appropriate permissions can now configure that rule on their delivery services
  - Probably not quite that simple in reality
- Provides better isolation between delivery services
Examples from proof of concept

(This probably isn’t how the final result will look)
{
    name = 'test_set_header-client_request',
    hook_point = '',
    conditions = {
        {
            type = 'uri',
            target = 'client_request',
            args = {},
            operator = '==',
            value = '/test_set_header-client_request',
        },
    },
    actions = {
        {
            type = 'set_header',
            target = 'client_request',
            args = {name='X-test-header', value='This is only a test.'},
        },
    },
}
{  
    name = 'test_set_response',  
    hook_point = '',  
    conditions =  
        {  
        },  
    actions =  
        {  
            {  
                type = 'set_response',  
                target = '',  
                args = {response_code=418, body="I'm a teapot"},  
            },  
        },
    }
}