Things you want to know about Lua in ATS

PRESENTED BY **Kit Chan (kichan@yahoo-inc.com)** | October 27, 2016

Agenda

- ts_lua plugin
  - Basic
  - Advanced
  - Putting it together!
  - What’s next?
- Lua scriptlet
  - Metric configuration
  - Log configuration
  - Storage configuration
  - What’s next?

ts_lua Plugin

The Basic

- can be used as global or remap plugin
- support adding hooks
- Client/Server Request
  - headers
  - method/domain/port/path/query params/matrix params
- Client/Server Response
  - status code/status string
  - headers

```
function send_response()
    local req_host = ts.client_request.header.Host
    local result = string.reverse(req_host)
    ts.client_response.header['Rhost'] = result
    return 0
end

function do_remap()
    ts.hook(TS_LUA_HOOK_SEND_RESPONSE_HDR, send_response)
    return 0
end
```

The Basic

```
function send_response()
    local req_host = ts.client_request.header.Host
    local result = string.reverse(req_host)
    ts.client_response.header['Rhost'] = result
    return 0
end

function do_global_read_request()
    ts.hook(TS_LUA_HOOK_SEND_RESPONSE_HDR, send_response)
    return 0
end
```

YAHOO!

Advanced

- API Support
  - Debug and error messages - TSDebug/TSError
  - Configuration Change - TSHttpTxnConfigIntSet
  - HTTP/2 server Push - TSHttpTxnServerpUsh
  - Async Fetch - TSFetchUrl
  - Schedule Continuations - TSContSchedule
  - Many other APIs.
- Other Features
  - request/response transformation
  - intercept/server intercept
- Unit Testing with “busted”
- https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/ts_lua.en.html

HTTP/2 Server Push

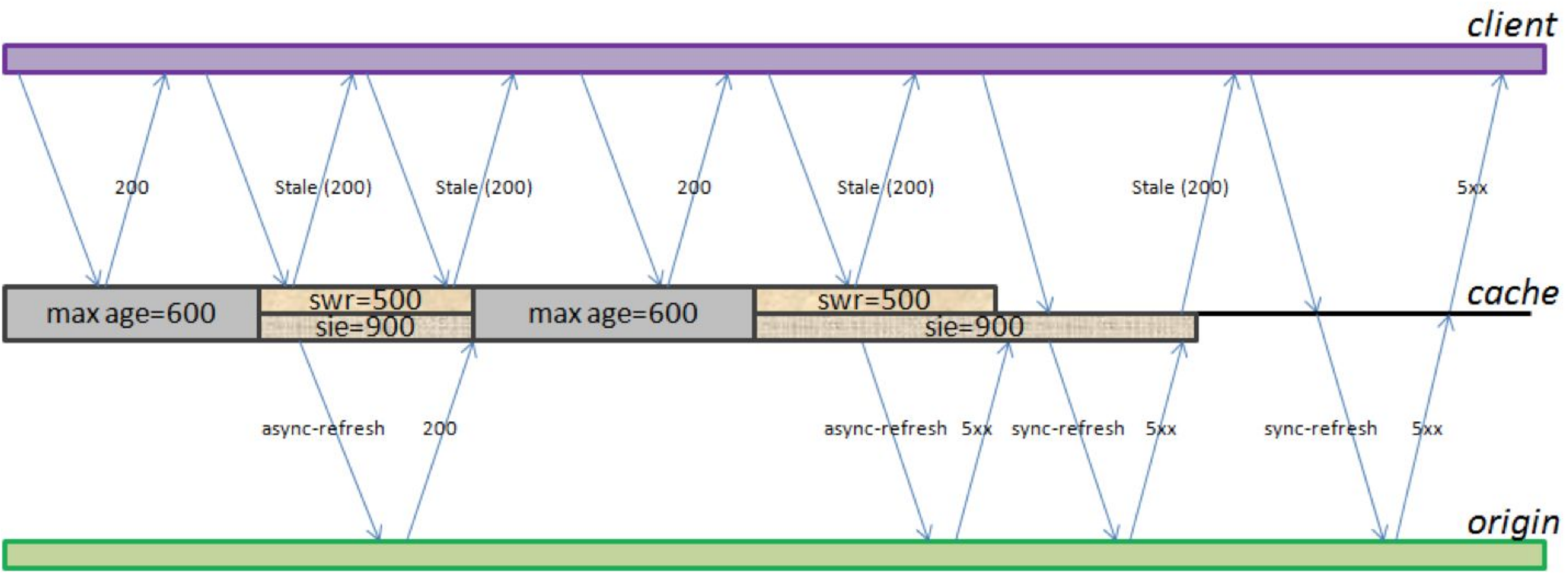
```
function do_remap()

    -- get client protocol stack
    local stack = {ts.http.get_client_protocol_stack()}
    for k,v in pairs(stack) do
      if(v == 'h2') then
        -- pushing an asset
        ts.http.server_push('https://testsecure.com/test.js')
      end
    end

    return 0
end
```

Putting it together!

- RFC 5861 - Stale-while-revalidate & Stale-if-error
  - Disclaimer: Just to demonstrate a slightly more complicated program for ts_lua
  - Not a production-ready implementation of the RFC



YAHOO!

Stale-If-Error

```
function do_global_read_request()
  ts.hook(TS_LUA_HOOK_CACHE_LOOKUP_COMPLETE, cache_lookup)
end

function cache_lookup()
  local inner = ts.http.is_internal_request()
  if inner ~= 0 then
    -- internal request - always make internal requests to be a cache miss so we retrieve from origin
    ts.http.set_cache_lookup_status(TS_LUA_CACHE_LOOKUP_MISS)
  else
    -- external request
    local cache_status = ts.http.get_cache_lookup_status()
    if cache_status == TS_LUA_CACHE_LOOKUP_HIT_STALE then       -- stale hit?
      local url = ts.client_request.get_url() or ''
      -- add extra query parameter to request
      url = url .. '?async=yes'
      local ct = {
        header = ts.client_request.get_headers()
      }
      local res = ts.fetch(url, ct)

      if res.status == 200 then
        ts.http.server_intercept(process, res.header, res.body ) -- response good, do intercept
      else
        ts.http.set_cache_lookup_status(TS_LUA_CACHE_LOOKUP_HIT_FRESH) -- response bad, return cache
      end
    end
  end
  return 0
end
```

Stale-If-Error - continued

```
function process(header, body)
  ts.debug('server intercept')

  local resp = 'HTTP/1.1 200 OK\r\n'

  for k,v in pairs(header) do
    resp = resp .. k .. ': ' .. v .. '\r\n'
  end
  resp = resp .. '\r\n' .. body

  ts.say(resp)
end
```

Stale-While-Revalidate

```
function cache_lookup()
  ts.debug('cache-lookup')

  local inner = ts.http.is_internal_request()
  if inner ~= 0 then

    -- always make internal requests to be a cache miss so we retrive from origin
    ts.http.set_cache_lookup_status(TS_LUA_CACHE_LOOKUP_MISS)
  else
    -- mark stale hit as fresh hit and do an async request
    local cache_status = ts.http.get_cache_lookup_status()
    if cache_status == TS_LUA_CACHE_LOOKUP_HIT_STALE then      -- stale hit
      ts.http.set_cache_lookup_status(TS_LUA_CACHE_LOOKUP_HIT_FRESH)

      -- schedule a continuation for async fetch
      ts.schedule(TS_LUA_THREAD_POOL_NET, 0, async)
    end
  end

  return 0
end

function do_global_read_request()
  -- retrieve URL and header for later use
  ts.ctx['url'] = ts.client_request.get_url()
  ts.ctx['headers'] = ts.client_request.get_headers()

  ts.hook(TS_LUA_HOOK_CACHE_LOOKUP_COMPLETE, cache_lookup)
end
```

YAHOO!

Stale-While-Revalidate - continued

```
function async()
  ts.debug("async")
  local url = ts.ctx['url'] or ''
  -- add extra query parameter to async request
  url = url .. '?async=yes'
  local ct = {
    header = ts.ctx['headers']
  }
  local res = ts.fetch(url, ct)
  if res.status == 200 then
    ts.debug('pushing')
    local purl = ts.ctx['url']
    local presp = 'HTTP/1.0 200 OK\r\n'
    local header = res.header
    for k, v in pairs( header) do
      presp = presp.. k .. ': ' .. v .. '\r\n'
    end
    presp = presp .. '\r\n' .. res.body
    local phdr = {}
    for k, v in pairs(ts.ctx['headers']) do
      phdr[k] = v
    end
    phdr['Content-Length'] = string.format('%d', string.len(presp))
    local pct = {
      header = phdr,
      method = 'PUSH',
      body = presp
    }
    local pres = ts.fetch(purl, pct)
  end
end
```

What’s next?

- Bug fixes & clean up
- support use case of new protocol plugin
- support SSL hooks
- support lifecycle hooks
- any API missing?
- moving out of experimental?

Lua scriptlet

Metrics Configuration

- James Peach did it !! (TS-4099)
- Replace stats.config.xml with metrics.config
- Optional in 6.2.0, Mandatory in 7.0.0
  - proxy.config.stats.enable_lua in 6.2.0
- https://docs.trafficserver.apache.org/en/latest/admin-guide/files/metrics.config.en.html

```
-- snippet in metrics.config
float 'proxy.node.cache.hits_ratio' [[
  return
    proxy.node.cache.hits /
    ( proxy.node.cache.hits +
      proxy.node.cache.misses +
      proxy.node.cache.revalidates
    )
]]

integer 'proxy.node.dns.lookups_per_second' [[
  local self = ...

  return rate_of_10s(self,
    function() return proxy.process.dns.total_dns_lookups end
  )
]]
```

Log Configuration

- James Peach did it again !! (TS-4548)
- in 7.0.0
- replace logs_xml.config with logging.config
- TS-4739 - a tool for upgrading existing log config to new format
- https://docs.trafficserver.apache.org/en/latest/admin-guide/files/logging.config.en.html

```
-- sample logging.config
minimalfmt = format {
  Format = '%<chi> : %<cqu> : %<pssc>'
}

refreshhitfilter = filter.accept('pssc MATCH REFRESH_HIT')

log.ascii {
  Filename = 'minimal',
  Format = minimalfmt,
  Filters = { refreshhitfilter }
}
```

Storage Configuration

- TS-5015 (just open)
- store.config replaces storage.config

```
-- sample store.config
store {
    Path = 'var/trafficserver/1',
    Size = '256M',
    Id = 'XXX',
    Volume = 2
}

store {
    Path = 'var/trafficserver/2',
    Size = '256M',
    Id = 'YYY',
    Volume = 2
}

store {
    Path = 'var/trafficserver/3',
    Size = '256M',
    Id = 'ZZZ',
    Volume = 3
}
```

How?

- create a BindingInstance
- attach configuration objects to the BindingInstance
- add your own binding functions/constants/etc
- run the configurable file written in lua

- Check out TS-4548 & TS-5015

What’s next?

- splitdns.config / congestion.config
- volume.config / hosting.config
- ssl_multicert.config
- parent.config
- plugin.config
- records.config
- remap.config
- log_hosts.config

- ip_allow.config / cache.config
  - Can they be removed in favor of ts_lua plugin?

- Plugins / tsconfig

Thanks!