Diagnostics & Debugging

Eric Schwartz

YAHOO!

Outline

- Debugging intro.
- How to get basic diagnostic information out of ATS.
- Per client IP debugging.
- Wire tracing.

Debugging Intro: Debug Tags

- ATS debug tags allow for printing of debug messages from different parts of traffic_server to diags.log and traffic.out.
- How to use:
  - Enabled via records.config:
    - CONFIG proxy.config.diags.debug.enabled INT 1
    - CONFIG proxy.config.diags.debug.tags STRING your_tags
  - Regex matching, can use multiple tags, separated by | (regex “or”)
- Disadvantages:
  - All that regex matching slows traffic_server down a ton!

YAHOO!

Debugging Intro: Debug Tags

- Common/useful debug tags:
  - http - messages in HttpSM, HttpTransact, etc.
  - url - url rewriting messages
  - dns|hostdb - HostDB & DNS messages
  - ssl - SSL-related messages
  - cache - Cache operations
  - plugin specific tags (header_rewrite, etc.)
  - . - will turn on all debug tags (this is a lot of messages).
  - and many more!
- If you write an ATS plugin, you can write your own tag & print debug messages with an API:
  - void TSDebug(const char *tag, const char *format_str, …)

YAHOO!

Debugging Intro: SSL/TLS

Fast setup of ATS for debugging SSL/TLS traffic:

- Configure explicit SSL server port:
  - ex: proxy.config.http.server_ports STRING 443:ssl
- Create a self-signed certificate and corresponding key using OpenSSL:
  - openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days XXX -nodes
  - You must change the permissions on these PEM files (and the directories they're in) so that ATS can open them.
- Modify ssl_multicert.config ssl_cert_name and ssl_key_name to include these for your test targets.

Debugging Intro: http/2

- Configure explicit http/2 server port:
  - ex: proxy.config.http.server_ports STRING 443:proto=http;http2:ssl
- Or, simply enable proxy.config.http2.enabled.
- For test client use nghttp, available as part of the nghttp2 library from Tatsuhiro Tsujikawa.
  - Available at https://nghttp2.org/
  - Github at https://github.com/tatsuhiro-t/nghttp2/
- Test client syntax very similar to cURL and spdycat.
- HTTP/2 talk yesterday went into more detail, talked about h2load.

Debugging Intro: gdb

- Basic gdb tip:
  - You can start traffic_server without running the trafficserver binary. If you're not looking for specific traffic_manager/traffic_server interactions, this can be a useful way to do some simple debugging of stuff in traffic_server core or plugins.
- gdb -p or gdb --pid can be used to debug a running traffic_server process if you get its PID.

Debugging Intro: gdb

- Debugging across forks.
  - Useful for debugging traffic_manager and traffic_server interactions.
  - set follow-fork-mode child: causes gdb to debug the child process and follow the execution flow into it.
  - set detach-on-fork off: causes gdb to keep control of the parent process, effectively halting that process which is usually desirable/required. In particular with respect to ATS, it stops health checks from being done, which would cause the child process to be killed out from under the debugger.
  - Breakpoints:
    - Set via file and line number, ex: HttpSM.cc:5637.
    - Have to be very careful when debugging across forks. You'll get a message warning you about "future library load" and should tell gdb it's ok. If you do this incorrectly you'll get no feedback.

traffic_ctl (for ATS 6.0.0 and newer)

- As of ATS 6.0.0, traffic_line is deprecated in favor of traffic_ctl.
- Functionality is very similar to traffic_line, syntax is different, useful for diagnostics and debugging:
  - Accessing statistics (like traffic_line -r):
    - traffic_ctl metric get METRIC
  - Accessing config values:
    - traffic_ctl config get RECORD
  - Changing config values (like traffic_line -s -v):
    - traffic_ctl config set RECORD VALUE
      - Updates a specific config variable.
    - traffic_ctl config reload
      - Reloads entire config.

traffic_ctl & stats (for ATS 6.0.0 and newer)

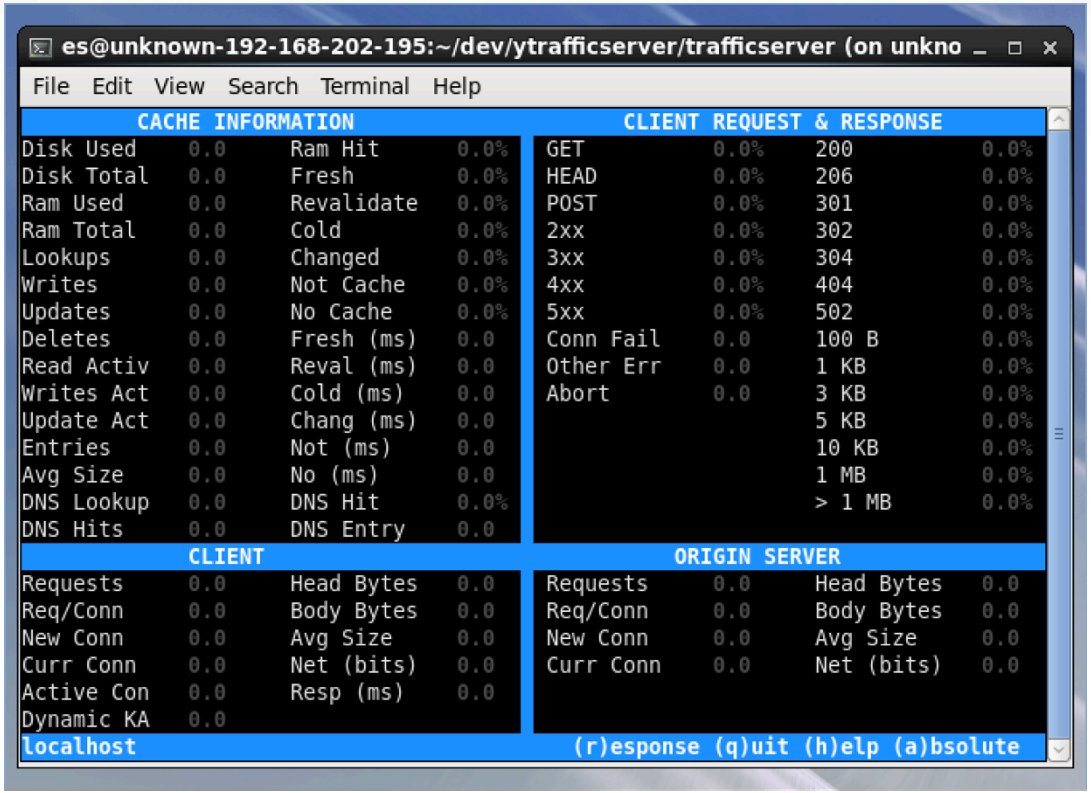
- Regex matching (like traffic_line -m):
  - traffic_ctl metric match REGEX
  - traffic_ctl record match REGEX
- Must be run as a user with permissions to access Traffic Server unix socket and change values. Typically root and whatever user you've configured Traffic Server to run as. In 6.0.0 and newer, reading values is permitted by any user by default.
- See all core statistics:
  - traffic_ctl metric match proxy
  - Other useful stat groups:
    - proxy.process.net
    - proxy.process.ssl
    - proxy.node.cache

YAHOO!

Basic Diagnostic Information: traffic_top and perf-top

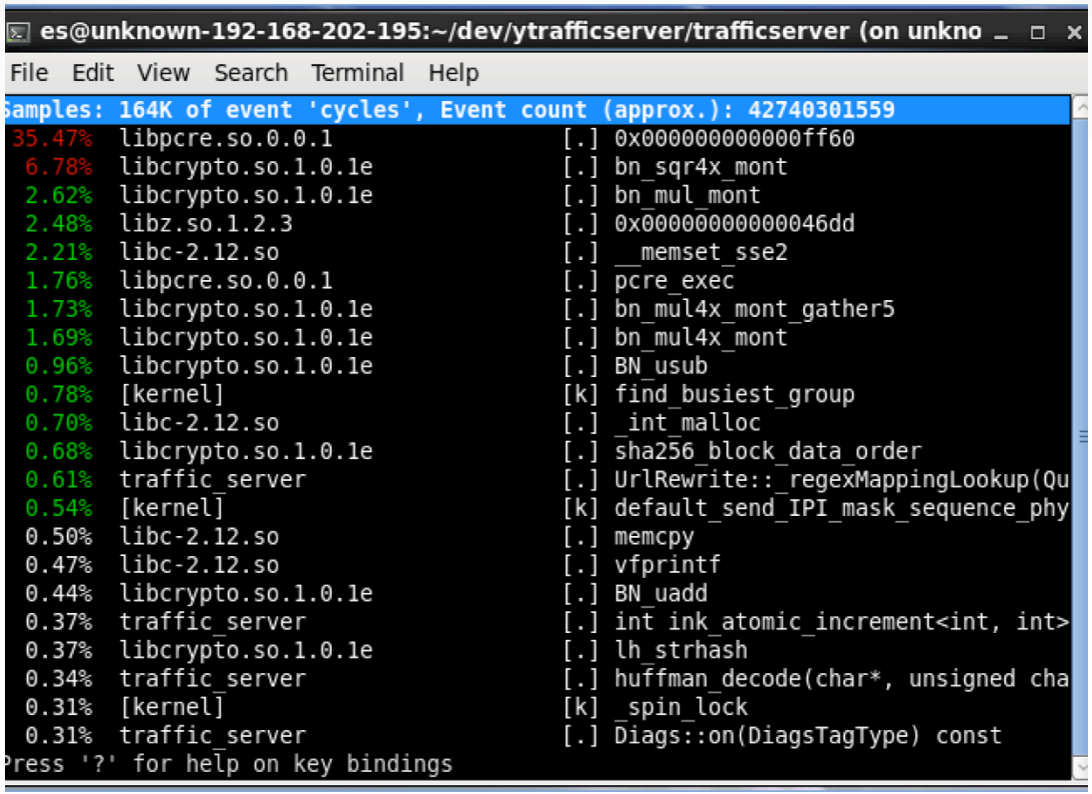
- traffic_top is a useful command line tool that is helpful for getting an overview of what the server is doing (cache hit rate, dns hit rate, response code distribution, RPS, etc.)
  - Usage: traffic_top [-s seconds] [URL|hostname|hostname:port]
  - Output on next slide.
- perf top is a useful linux tool for doing system profiling for the host on which ATS is running.
  - Linux man page.

Basic Diagnostic Information: traffic_top and perf-top



YAHOO!

Basic Diagnostic Information: traffic_top and perf-top



YAHOO!

Basic Diagnostic Information: Slow Logs

- Available as a records.config setting: proxy.config.http.slow.log.threshold.
  - Time given in milliseconds.
- When slow log threshold is reached, ATS will dump debugging stats for that connection. This includes milestones allowing you to see which parts of the transaction really slowed things down.
- Slow log message:
  - [Nov 9 15:34:59.963] Server {0x2abd7df0a700} ERROR: [0] Slow Request: client_ip: 127.0.0.1:56764 url: https://www.yahoo.com/ status: 200 unique id: bytes: 282818 fd: 0 client state: 0 server state: 9 ua_begin: 0.000 ua_read_header_done: 0.000 cache_open_read_begin: 0.000 cache_open_read_end: 0.000 dns_lookup_begin: 0.000 dns_lookup_end: 0.037 server_connect: 0.037 server_first_read: 0.191 server_read_header_done: 0.191 server_close: 0.511 ua_close: 0.511 sm_finish: 0.511 plugin_active: -1.000

YAHOO!

Basic Diagnostic Information: Memory Dumping

- Available as a records.config setting: proxy.config.dump_mem_info_frequency
  - Time given in seconds.
- Will dump memory info by free list name into traffic.out.
- Very useful for locating memory leaks. Can compare subsequent dumps to see if allocated space is increasing for anything.
- Outputs tables that look like this:

```
     allocated      |       in-use       | type size  |   free list name
--------------------|--------------------|------------|---------------------------------
                   0 |                  0 |    2097152 | memory/ioBufAllocator[14]
                   0 |                  0 |    1048576 | memory/ioBufAllocator[13]
                   0 |                  0 |     524288 | memory/ioBufAllocator[12]
                   0 |                  0 |     262144 | memory/ioBufAllocator[11]
                   0 |                  0 |     131072 | memory/ioBufAllocator[10]
```

YAHOO!

Per Client IP Debugging

- Currently in the process of being submitted to open source ATS.
- Available as a dynamic records.config setting:
  - proxy.config.diags.debug.client_ip STRING
- Turns on debug logging to diags.log and traffic.out for a specific client IP address.
- Performance impact is minimal.
  - Log tag check is done after IP address check to minimize number of times a regex check is done.
- Can be enabled in production.
  - Can have an engineer hitting a live production system from a known IP address attempt to reproduce the issue and see what information can be gleaned from the logs.

SSL Wire Tracing

- Allows for tracing of SSL/TLS traffic immediately after the SSL_read call.
- Is done on at the connection level, so will trace for the entire session until the connection is returned to the pool.
- Tracing is also enabled for corresponding origin connections and this association is shown in logs, so if anything is going wrong within ATS (such as post bodies not getting forwarded) can see in differences between client and origin traces.
- Also useful for seeing errors in the SSL handshake. Wire tracing will also display these messages for activated connections.
- Can be run in production if tracing for a small % of total traffic.

SSL Wire Tracing

- Enabled via a records.config setting:
  - proxy.config.ssl.wire_trace_enabled INT
- Can trace based on client IP, SNI information and sampling:
  - proxy.config.ssl.wire_trace_addr STRING
  - proxy.config.ssl.wire_trace_server_name STRING
  - proxy.config.ssl.wire_trace_percentage INT
- Outputs traces to error.log.

- proxy.config.ssl.wire_trace_percentage can be combined with other settings or used on its own:
  - Trace 1% of all traffic to all origins:
    - proxy.config.ssl.wire_trace_percentage INT 1
  - Trace 100% of traffic coming from IP 206.190.37.108:
    - proxy.config.ssl.wire_trace_addr STRING 206.190.37.108
    - proxy.config.ssl.wire_trace_percentage INT 100
  - Trace 10% of traffic to www.yahoo.com:
    - proxy.config.ssl.wire_trace_server_name STRING www.yahoo.com
    - proxy.config.ssl.wire_trace_percentage INT 10

YAHOO!

Recent Enhancements to SSL Wire Tracing

- More detailed SSL Error messaging getting the error string from OpenSSL ERR_peak_last_error()
- Made wire tracing dynamically available.
  - Can enable in an existing prod system using traffic_line or traffic_ctl

Other Uses for Wire Tracing: Traffic Capture

- Useful for traffic capture.
- Because it's session-based and tags the start and end of each trace, we can capture both the client and corresponding origin traces for an entire session.
  - This correspondence is included in the tags for the origin traces.
- These can then be parsed to recreate an entire client-ATS-origin session of request/response pairs.
  - We post process with a python script that reconstructs each session as a JSON, grabbing the real requests and responses from the logs and can then use these to do testing.
- We can then replay these sessions against test versions of ATS, exposing them to a facsimile of real production traffic.

YAHOO!

The End/Questions?

YAHOO!