Debugging ATS

Bryan Call & Brian Geffon

Common Issues

- Request failing
- Coredump
- Memory leak
- CPU usage high
- Performance
- Networking

Diags/Debug Tags

- Diags/debug tags
  - Uses a regular expression to determine what tags to show
  - Runs **a lot slower** with diags on
    - Might not be able to run in production
  - Configuration and command line options
  - Show location
    - File, line, function

Diags/Debug Tags Example

- Configuration options

```
$ sudo traffic_line -s proxy.config.diags.debug.tags -v http
$ sudo traffic_line -s proxy.config.diags.debug.enabled -v 1
$ sudo traffic_line -s proxy.config.diags.show_location -v 1
```

- Example

```
[Oct 20 09:01:09.530] Server {0x2b0499f24700} DEBUG: <HttpTunnel.cc:1269
(consumer_handler)> (http_tunnel) [42002] consumer_handler [user agent - tunnel
VC_EVENT_WRITE_READY]
[Oct 20 09:01:09.530] Server {0x2b0499f24700} DEBUG: <HttpTunnel.cc:1082
(producer_handler)> (http_tunnel) [42002] producer_handler [http server - tunnel
VC_EVENT_READ_READY]
[Oct 20 09:01:09.530] Server {0x2b0499f24700} DEBUG: <HttpTunnel.cc:1122
(producer_handler)> (http_redirect) [HttpTunnel::producer_handler] enable_redirection: [1
0 1] event: 100
```

Diags/Debug Tags Example

- Or on the command line

```
sudo traffic_server -T ssl
```

```
[Oct 17 14:49:03.410] Server {0x7fff7dcc5300} DEBUG: (ssl.session_cache) Created new ssl session cache 0x7fce92f16930 with 256 buckets each with size max size 400
[Oct 17 14:49:03.413] Server {0x7fff7dcc5300} DEBUG: (ssl.session_cache) ssl context=0x7fce92e39990: using session cache options, enabled=2, size=102400, num_buckets=256, skip_on_contention=0, timeout=0, auto_clear=1
[Oct 17 14:49:03.413] Server {0x7fff7dcc5300} DEBUG: (ssl.session_cache) enabling SSL session cache with ATS implementation
[Oct 17 14:49:03.432] Server {0x7fff7dcc5300} DEBUG: (ssl) Using 'localhost.crt' in hash for session id context
[Oct 17 14:49:03.433] Server {0x7fff7dcc5300} DEBUG: (ssl) setting SNI callbacks with for ctx 0x7fce92e39990
[Oct 17 14:49:03.434] Server {0x7fff7dcc5300} DEBUG: (ssl) indexed '*' with SSL_CTX 0x7fce92e39990 [0]
[Oct 17 14:49:03.434] Server {0x7fff7dcc5300} DEBUG: (ssl) ssl ocsp stapling is enabled
[Oct 17 14:49:03.434] Server {0x7fff7dcc5300} DEBUG: (ssl) can not get issuer certificate!
```

Via Header

- Information about the request
  - Cached / not cached
  - Wrote to cache
  - Errors
  - And more...
- Via decoder
  - Human readable
  - http://trafficserver.apache.org/tools/via
  - Traffic line in 5.1:

```
$ traffic_line --decode_via "[uScHs f p eN:t cCHi p s ]"
```

Via Header Example

- Configuration option

```
$ sudo traffic_line -s proxy.config.http.insert_response_via_str -v 3
```

- Example

```
$ curl -s -D - http://example.com | grep ^Via:
Via: http/1.1 example.com (ApacheTrafficServer/5.1.0 [uSc sSf pSeN:tOc  i p sS])
```

Via Decoder - http://trafficserver.apache.org/tools/via

Via header parser

Enter your Via header into the box below to parse it:

uSc sSf pSeN:tOc i p s$ Parse header

Proxy request results:

Request headers received from client:	simple request (not conditional)
Result of Traffic Server cache lookup for URL:	no cache lookup performed
Response information received from origin server:	served
Result of document write-to-cache:	no cache write performed
Proxy operation result:	served
Error codes (if any):	no error

Operational results:

Tunnel info:	tunneling because cache is turned off
Cache-type and cache-lookup cache result values:	cache miss or no cache lookup / no cache lookup
ICP status:	no icp
Parent proxy connection status:	no parent proxy
Origin server connection status:	connection opened successfully

Request Failing

- Sources of common error codes.
  - 404 - remap errors, downstream returning 404, how to differentiate
  - 503 - origin down or unresponsive
  - 403 - how to differentiate between permission issue or
  - 000 - unknown - client not sending bytes

Core Files

A core file is simply a full dump of a process including full memory space.

Getting a core file

- Records.config

```
CONFIG proxy.config.core_limit INT -1  # Equivalent to ulimit -c unlimited
```

- Sysctl

```
sysctl -w kernel.core_pattern = /tmp/%e.core.%p
```

Remember to demangle symbols in crash reports (c++filt)

```
/usr/local/bin/traffic_server - STACK TRACE:
/lib64/libpthread.so.0[0x2aafe810eca0]
/usr/local/bin/traffic_server(_Z16mime_scanner_getP11MIMEScannerPPKcS2_S3_S3_Pbbi+0x2c2)[0x5cf752]
/usr/local/bin/traffic_server(_Z21http_parser_parse_reqP10HTTPParserP7HdrHeapP11HTTPHdrImplPPKcS6_bb+0x113)[0x5c4e73]
/usr/local/bin/traffic_server(_ZN7HTTPHdr9parse_reqEP10HTTPParserP14IOBufferReaderPib+0x1a7)[0x5c11d7]
```

VS

```
/usr/local/bin/traffic_server(mime_scanner_get(MIMEScanner*, char const**, char const*, char const**, char const**, bool*, bool,
int)+0x2c2)[0x5cf752]
/usr/local/bin/traffic_server(http_parser_parse_req(HTTPParser*, HdrHeap*, HTTPHdrImpl*, char const**, char const*, bool, bool)
+0x113)[0x5c4e73]
/usr/local/bin/traffic_server(HTTPHdr::parse_req(HTTPParser*, IOBufferReader*, int*, bool)+0x1a7)[0x5c11d7]
```

Intro to GDB

- It’s helpful to install trafficserver with debug info packages.

- What would a sequence of commands that would help facilitate a bug report?

1) **bt full**
2) **list**
3) **info thread**

GDB: Filing better bug reports

sudo gdb /usr/bin/traffic_server ‘corefile’

(1) Backtracing

(gdb) bt full

```
#0  0x00000031f5e32925 in raise () from /lib64/libc.so.6
#1  0x00000031f5e34105 in abort () from /lib64/libc.so.6
#8  0x000000000075fd4a in EThread::process_event (this=0x2b0c97402010, e=0x2b0ccd4f8940, calling_code=2) at
UnixEThread.cc:145
        c_temp = 0x2b0d0b55b4d0
        lock = {m = {m_ptr = 0x2b0cdc1835e0}, lock_acquired = true}
#9  0x0000000000760065 in EThread::execute (this=0x2b0c97402010) at UnixEThread.cc:224
        done_one = true
        e = 0x2b0ccd4f8940
        NegativeQueue = {<DLL<Event, Event::Link_link>> = {head = 0x1f513f0}, tail = 0x1f513f0}
        next_time = 1412955871075946883
```

GDB: Filing better bug reports

(2) list

```
(gdb) f 6      # enter frame 6
(gdb) list     # show the code around the frame
985  int
986  INKContInternal::handle_event(int event, void *edata)
987  {
988    if (m_free_magic == INKCONT_INTERN_MAGIC_DEAD) {
989      ink_release_assert(!"Plugin tries to use a continuation which is
deleted");
990    }
991    handle_event_count(event);
```

GDB: Filing better bug reports

(3) info thread

```
(gdb) info thread
  58 Thread 0x2b0c9da19700 (LWP 30667)  0x00000031f5ee9153 in epoll_wait () from
/lib64/libc.so.6
  38 Thread 0x2b0d6ceb3700 (LWP 30726)  0x00000031f660ea5d in accept () from
/lib64/libpthread.so.0
  27 Thread 0x2b0c9e322700 (LWP 30676)  0x00000000005d403b in HttpSM::
state_api_callout (this=0x2b0d0fe77ae0, event=0, data=0x0) at HttpSM.cc:1422
  25 Thread 0x2b0d84b89700 (LWP 30731)  0x00000031f660b98e in
pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
  24 Thread 0x2b0cb8657700 (LWP 30693)  0x00000031f660b98e in
pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
```

GDB Macro

- Run complex gdb commands easily

- Great for production debugging

```
sudo gdb ./traffic_server -p $(pidof traffic_server) -batch -x [macro_file]
```

GDB Macro Example

```
$ cat session.macro
break HttpSessionManager::acquire_session
cont
p hostname
quit
$ sudo gdb ./traffic_server $(pidof traffic_server) -batch -x session.macro
...
234    HttpServerSession *to_return = NULL;
$1 = 0x1b26c59 "127.0.0.1"
```

Detecting Memory Leaks

- Freelists can be difficult to debug

  $ sudo traffic_lne -s proxy.config.dump_mem_info_frequency -v 300

- `SIGUSR1` -- Generate a dump at any time of the freelist usage.

  $ sudo kill -SIGUSR1 $(pidof traffic_server)

- IOBuffer Debugging

  $ sudo traffic_lne -s proxy.config.res_track_memory -v 1

Detecting Memory Leaks Example

- Class allocator information

```
[TrafficServer] using root directory '/usr/local'
     allocated      |        in-use      | type size  |   free list name
--------------------|--------------------|------------|---------------------------------
             262144 |               8192 |       8192 | memory/ioBufAllocator[6]
             524288 |                  0 |       4096 | memory/ioBufAllocator[5]
                  0 |                  0 |       2048 | memory/ioBufAllocator[4]
              49152 |              29952 |         96 | memory/eventAllocator
            1646592 |            1642464 |         96 | memory/mutexAllocator
               8192 |                320 |         64 | memory/ioBlockAllocator
               6144 |                288 |         48 | memory/ioDataAllocator
              30720 |                960 |        240 | memory/ioAllocator
```

Detecting Memory Leaks Example

- Detailed IOBuf allocation information

  $ sudo traffic_line -s proxy.config.res_track_memory -v 1

```
                                        Location |          Size In-use
-------------------------------------------------+-----------------------
  memory/IOBuffer/ProtocolProbeSessionAccept.cc:67 |                    0
                    memory/IOBuffer/HttpSM.cc:5719 |                    0
                    memory/IOBuffer/HttpSM.cc:6321 |                    0
           memory/IOBuffer/HttpServerSession.cc:87 |                    0
                     memory/IOBuffer/Cache.cc:2749 |                 8192
                                             TOTAL |                 8192
```

Detecting Memory Leaks

- Valgrind is a good tool as a last resort (it's very slow).

```
--tool=memcheck --leak-check=full
```

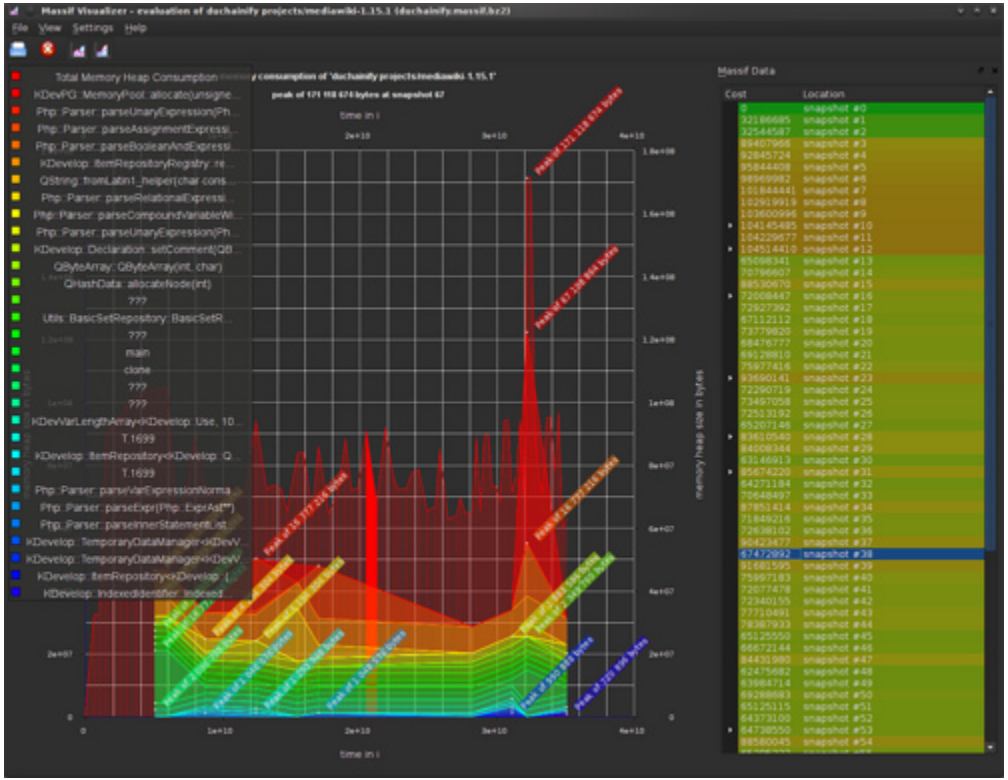
```
--tool=massif
```

- Because the freelists are never returned to the OS freelist items will show up as "possibly lost."

Detecting Memory Leaks

- If the leak isn’t in the freelist you can also consider:
  - jemalloc
  - tcmalloc

Massif visualizer



CPU Usage

- perf - sampling profiler

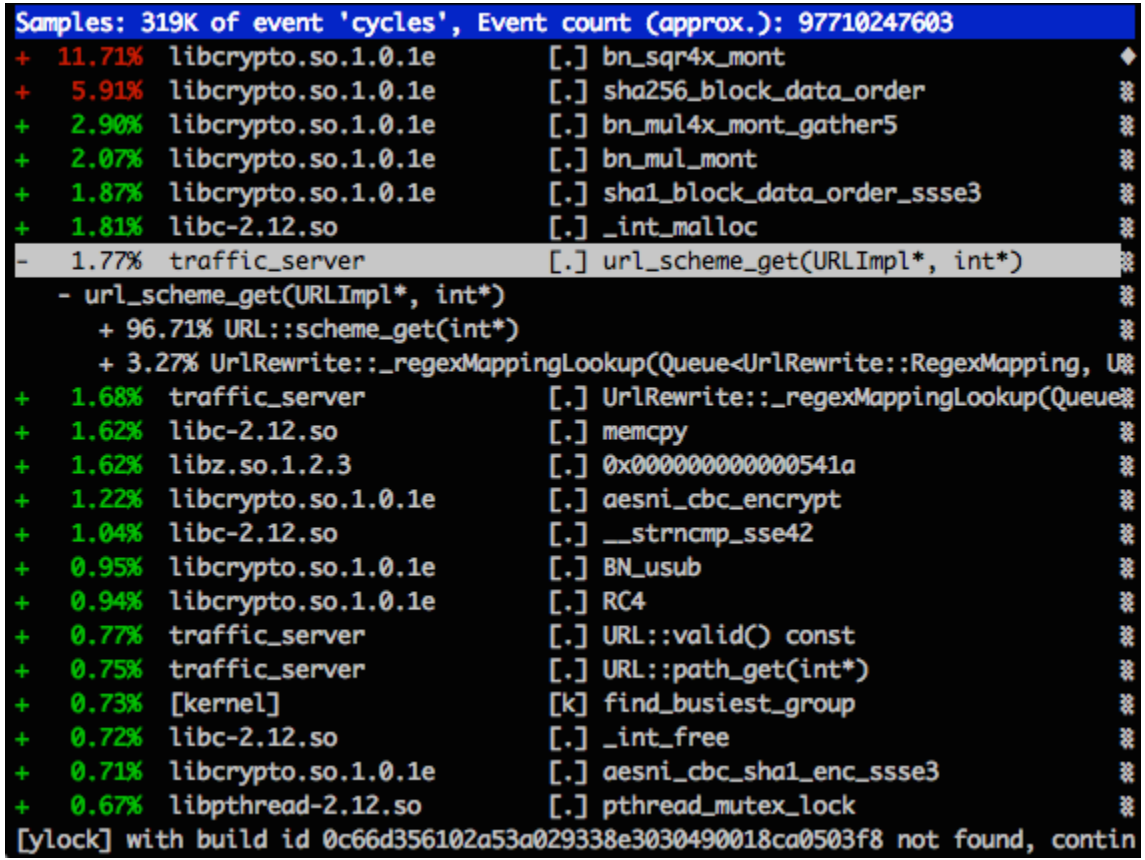
```
$ sudo perf record -g -p $(pidof traffic_server)
$ sudo perf report
```

OR

```
$ sudo perf top -g -p $(pidof traffic_server)
```

- Call Graph support
  - Option: -g or -G (depends on the version)

Perf Output


Samples: 319K of event 'cycles', Event count (approx.): 97710247603
+ 11.71% libcrypto.so.1.0.1e [.] bn_sqr4x_mont
+ 5.91% libcrypto.so.1.0.1e [.] sha256_block_data_order
+ 2.90% libcrypto.so.1.0.1e [.] bn_mul4x_mont_gather5
+ 2.07% libcrypto.so.1.0.1e [.] bn_mul_mont
+ 1.87% libcrypto.so.1.0.1e [.] sha1_block_data_order_ssse3
+ 1.81% libc-2.12.so [.] _int_malloc
- 1.77% traffic_server [.] url_scheme_get(URLImpl*, int*)
- url_scheme_get(URLImpl*, int*)
+ 96.71% URL::scheme_get(int*)
+ 3.27% UrlRewrite::_regexMappingLookup(Queue<UrlRewrite::RegexMapping, U
+ 1.68% traffic_server [.] UrlRewrite::_regexMappingLookup(Queue
+ 1.62% libc-2.12.so [.] memcpy
+ 1.62% libz.so.1.2.3 [.] 0x00000000000541a
+ 1.22% libcrypto.so.1.0.1e [.] aesni_cbc_encrypt
+ 1.04% libc-2.12.so [.] __strncmp_sse42
+ 0.95% libcrypto.so.1.0.1e [.] BN_usub
+ 0.94% libcrypto.so.1.0.1e [.] RC4
+ 0.77% traffic_server [.] URL::valid() const
+ 0.75% traffic_server [.] URL::path_get(int*)
+ 0.73% [kernel] [k] find_busiest_group
+ 0.72% libc-2.12.so [.] _int_free
+ 0.71% libcrypto.so.1.0.1e [.] aesni_cbc_sha1_enc_ssse3
+ 0.67% libpthread-2.12.so [.] pthread_mutex_lock
[ylock] with build id 0c66d356102a53a029338e3030490018ca0503f8 not found, contin

Performance

- Keep-alive

```
$ sudo traffic_line -s proxy.config.http.keep_alive_enabled_in -v 1
$ sudo traffic_line -s proxy.config.http.keep_alive_enabled_out -v 1
$ sudo traffic_line -s proxy.config.http.keep_alive_post_out -v 1
$ sudo traffic_line -s proxy.config.http.auth_server_session_private -v 0
```

Performance - slowlog

```
# time in is ms
$ sudo traffic_line -s proxy.config.http.slow.log.threshold -v 100
$ grep -i Slow diags.log
[Oct 14 15:33:16.194] Server {0xb5b5000} ERROR: [3] Slow Request: client_ip:
127.0.0.1:63691 url: http://www.yahoo.com/ status: 301 unique id:  bytes: 1450 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: 0.000 server_connect: 0.000 server_first_read: 0.097
server_read_header_done: 0.097 server_close: 0.105 ua_close: 0.105 sm_finish:
0.106
```

Networking

- At times it’s necessary to inspect individual packets, or even manipulate them.

- wireshark (tcpdump) and scapy are lifesavers.

Networking - tcpdump

To print all IPv4 HTTP packets to and from port 80, i.e. print only packets that contain data, not, for example, SYN and FIN packets and ACK-only packets. (IPv6 is left as an exercise for the reader.)

```
tcpdump -i any ’tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)’
```

```
man 8 tcpdump
```

Packet Capturing -- What about SSL?

- Disable Perfect Forward Secrecy
- Get access to the private key
- tshark / wireshark make this easy.

Networking -- Charles

- Sometimes Charles proxy is also hugely useful for SSL debugging (depending on what you're trying to accomplish).

- Also charles is great for bandwidth limiting / timeout simulations.