State of General Allocator Effort

Fei Deng, Chris Hassell, Susan Hinrichs

ATS Euro Tour

Cork

May 2018

Move Towards General Allocation Libraries

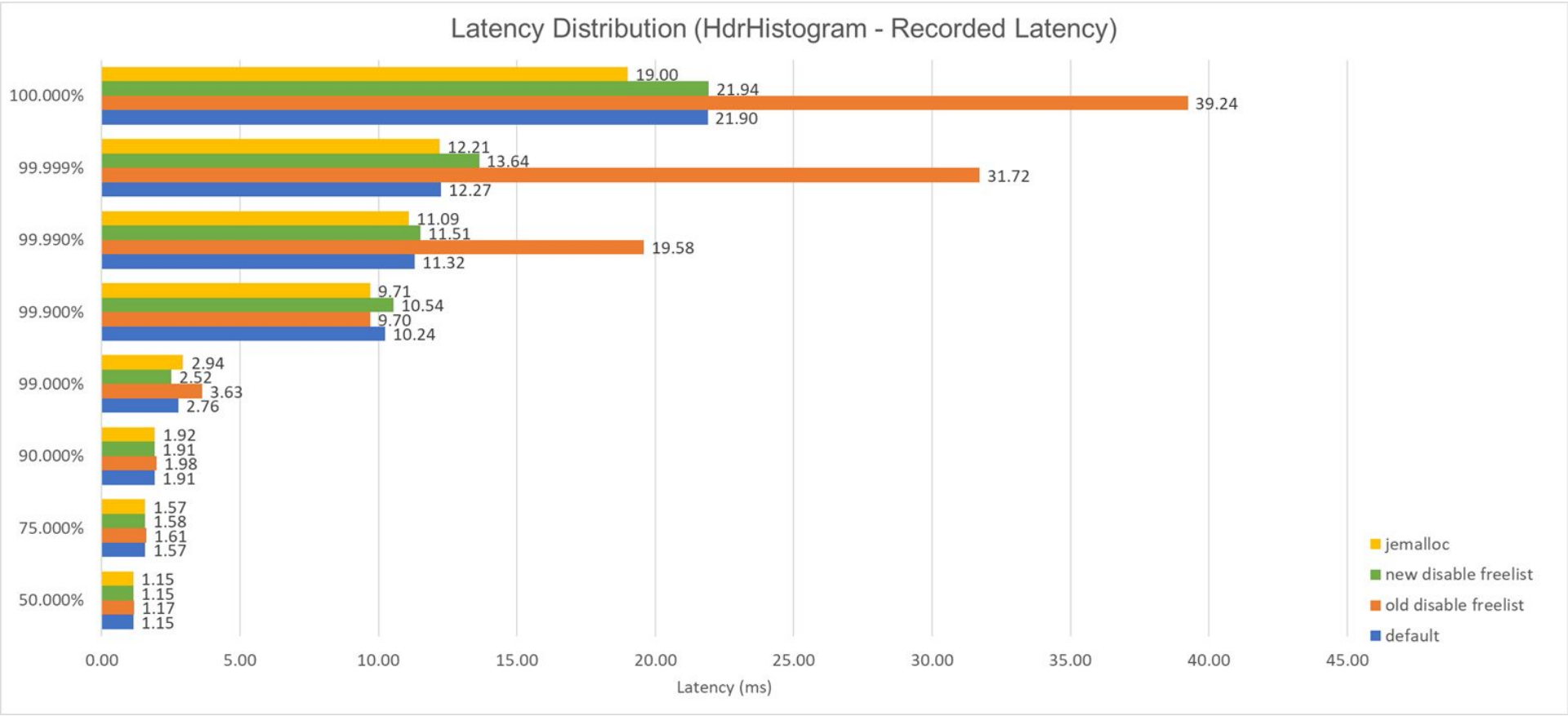
- General libraries have technically surpassed ATS's internal free list
  - Tcmalloc, jemalloc, even glibc
  - Superior control for profiling, debugging, optimizing for NUMA
    - E.g, Kit's experience with jemalloc
  - Avoids problems of over allocating in one size to starve allocation in another size later
- Goal to adopt one or more general allocation libraries and ultimately remove the ATS freelist code
  - Some tracking via github project https://github.com/apache/trafficserver/projects/10

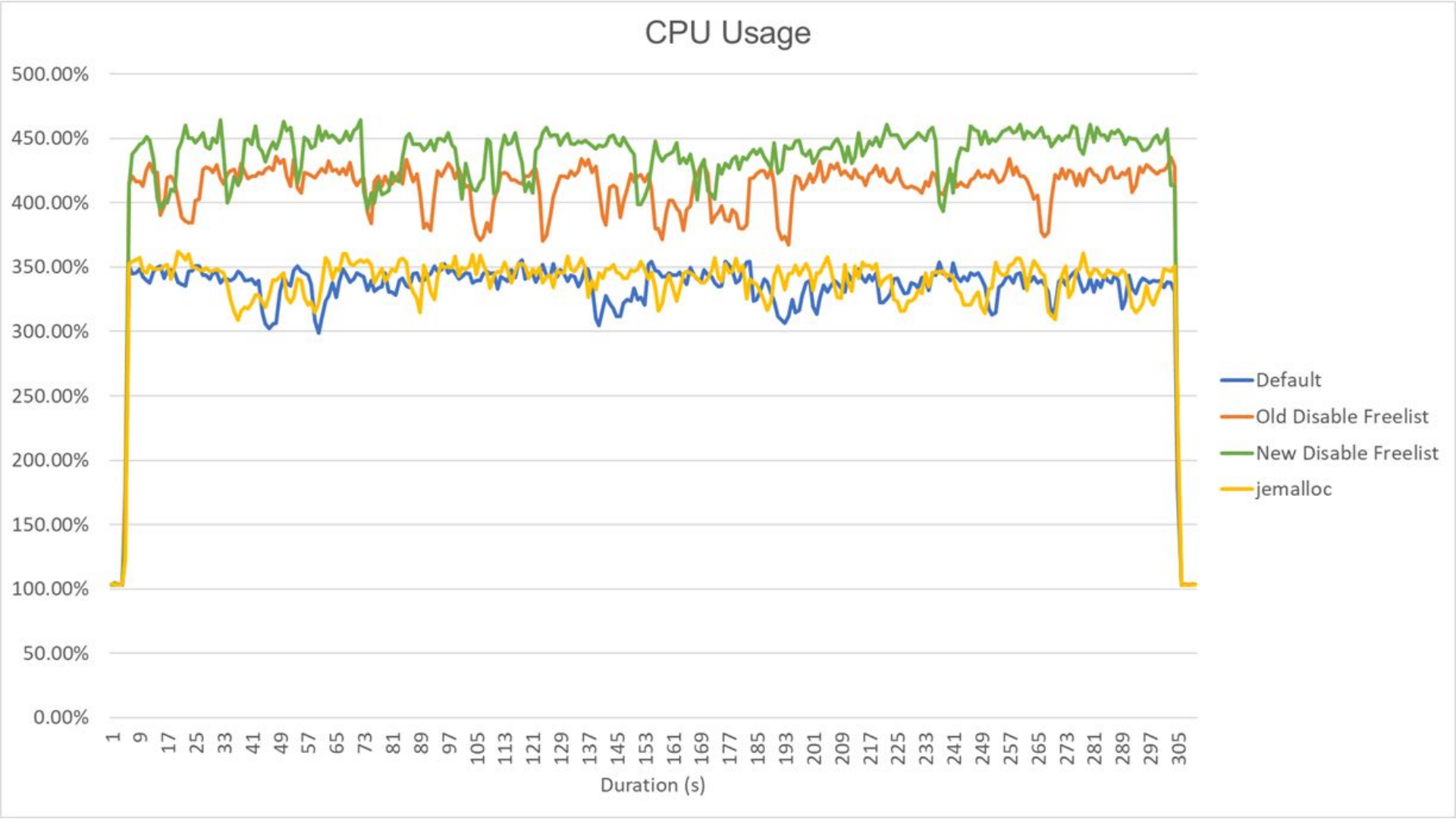
Phase 1 Concerns

- Turning off ATS free least should not reduce performance (much)
  - Issue of leaving per-thread proxy allocator (-f) or not (more than -f)
- When doing a core dump, we need to be able to mark regions of memory as not dumped (IOBuffers). Necessary to avoid truly enormous core files.

Performance

- Fei’s measurements from march https://github.com/apache/trafficserver/issues/3354
- Comparing
  - Default = with freelist
  - Old Disable Freelist = -f option (only disable ATS global allocator)
  - New Disable Freelist = -f option and disable freelist in ProxyAllocator
  - jemalloc = New Disable Freelist and compiled with jemalloc
- Wrk2 workload against cached data
  - ./wrk -t8 -c1000 -d300s -R20000 -L -H"Content-MD5: 1" http://127.0.0.1:8080
  - On a lab prod box


Latency Distribution (HdrHistogram - Recorded Latency)
100.000%
99.999%
99.990%
99.900%
99.000%
90.000%
75.000%
50.000%
19.00
21.94
39.24
21.90
12.21
13.64
31.72
12.27
11.09
11.51
19.58
11.32
9.71
10.54
9.70
10.24
2.94
2.52
3.63
2.76
1.92
1.91
1.98
1.91
1.57
1.58
1.61
1.57
1.15
1.15
1.17
1.15
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
40.00
45.00
Latency (ms)
jemalloc
new disable freelist
old disable freelist
default


CPU Usage
500.00%
450.00%
400.00%
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
1
9
17
25
33
41
49
57
65
73
81
89
97
105
113
121
129
137
145
153
161
169
177
185
193
201
209
217
225
233
241
249
257
265
273
281
289
297
305
Duration (s)
Default
Old Disable Freelist
New Disable Freelist
jemalloc

Dumping Core

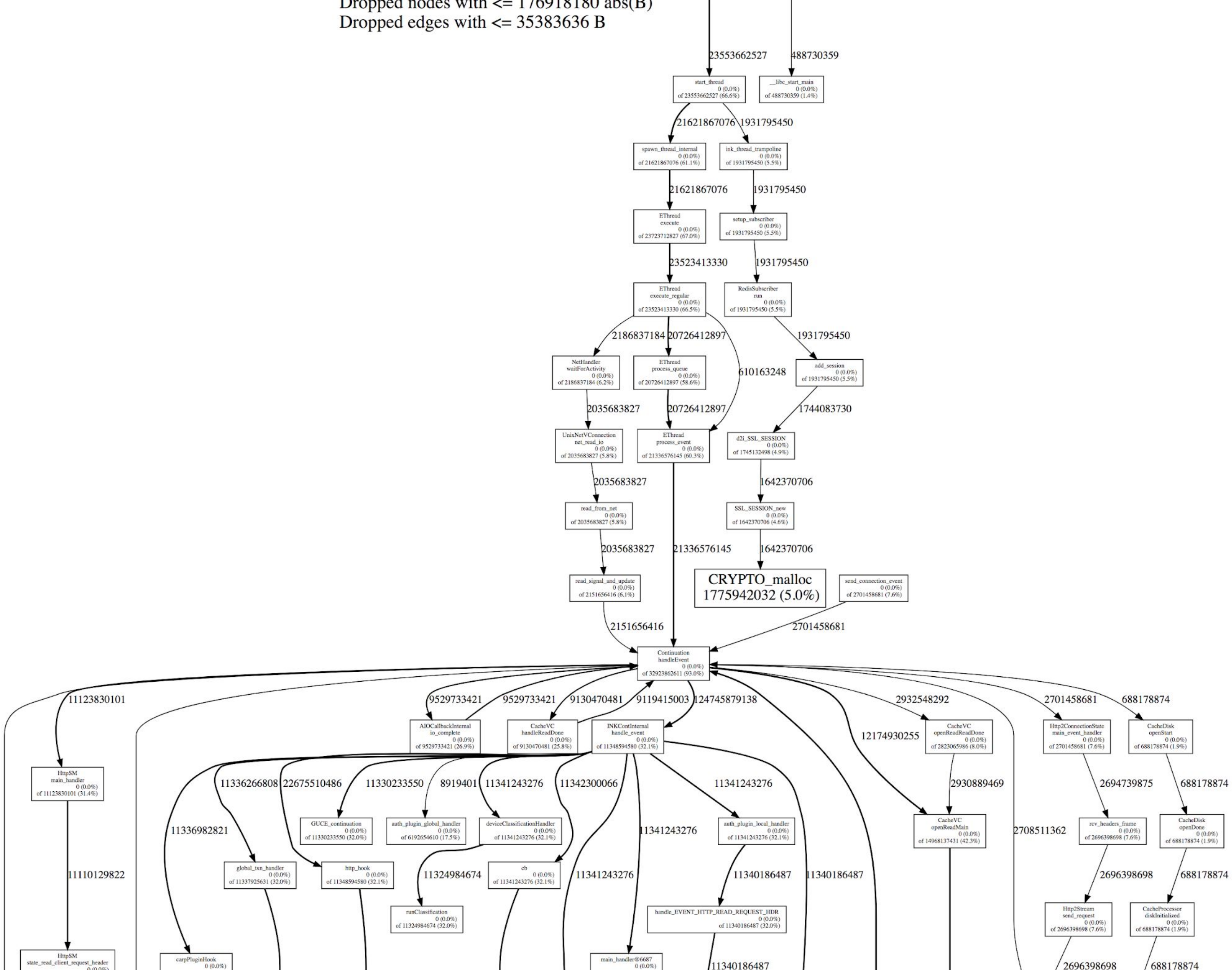
- The original madvise(MADV_DONTDUMP) doesn’t work for the free list case. When data that was allocated with MADV_DONTDUMP is reallocated it may be recallocated in a DODUMP scenario
  - Fei tried to add MADV_DODUMP before freeing for the DONTDUMP case. Unstable.
  - Fei tried to always call MADV_DODUMP or MADV_DONTDUMP in all allocation cases. Performance problems.
- Ended up with a jemalloc-only pool based solution explored by Facebook and Chris.
  - Commit 284fb4d56a1251cbec4a755472d2f1a9f4ac3ffe
  - Downside is that it ties us to a specific allocator.
  - Could be prettier. For first pass, did minimal ATS code change.

State of Phase I

- Core dump issue addressed if you compile with jemalloc
- PR in to augment -f to disable proxy allocator in addition to global allocator
- We hope to start rolling out jemalloc-only this quarter
  - Blocked by other internal dependencies
- Working with jemalloc-5.0.1

Other Benefits of Running with jemalloc

- Finding memory leaks
  - https://github.com/jemalloc/jemalloc/wiki/Use-Case:-Leak-Checking
  - Kit's Summit presentation https://cwiki.apache.org/confluence/download/attachments/70255385/ATSSummit_jemalloc.pptx?version=1&modificationDate=1508884895000&api=v2
  - Tracking leak in SSL session reuse (Image next slide)
- Finding memory corruptions
  - https://github.com/jemalloc/jemalloc/wiki/Use-Case:-Find-a-memory-corruption-bug
  - Fill memory with junk on free. Enabled at runtime via environment variable
  - Fei found at least 5 memory corruptions bugs with a combination of ASAN and junk fill in ATS this spring


Dropped nodes with <= 176918180 abs(B)
Dropped edges with <= 35383636 B
start_thread
__libc_start_main
spawn_thread_internal
ink_thread_trampoline
EThread execute
setup_subscriber
EThread execute_regular
RedisSubscriber run
NetHandler waitForActivity
EThread process_queue
add_session
UnixNetVConnection net_read_io
EThread process_event
d2i_SSL_SESSION
read_from_net
SSL_SESSION_new
read_signal_and_update
CRYPTO_malloc
1775942032 (5.0%)
send_connection_event
Continuation handleEvent
AIOCallbackInternal io_complete
CacheVC handleReadDone
INKContInternal handle_event
CacheVC openReadReadDone
Http2ConnectionState main_event_handler
CacheDisk openStart
HttpSM main_handler
GUCE_continuation
auth_plugin_global_handler
deviceClassificationHandler
auth_plugin_local_handler
CacheVC openReadMain
rcv_headers_frame
CacheDisk openDone
global_txn_handler
http_hook
cb
runClassification
handle_EVENT_HTTP_READ_REQUEST_HDR
Http2Stream send_request
CacheProcessor diskInitialized
HttpSM state_read_client_request_header
carpPluginHook
main_handler@6687

Next Phase

- NUMA manipulation
  - Chris has done some experimentation in this space
  - Jemalloc arenas give us some nice controls
    - Memkind http://memkind.github.io/memkind/
      - Unfortunately their white paper isn't very clear about how they map cores to arenas
- Completely remove ATS free lists
  - Less code to support
  - Simplify the allocation/free code path
  - Perhaps move to tradition new/delete.