The Curious Case of Memory Growth

A Debugging Story

Anita Zhang
engineerd managerd (Software Engineering Manager)
Agenda

01 Setting the Stage
02 Searching for Answers
03 Root Cause Explained
04 Hindsight
Setting the Stage
Setting the Stage

Specification

- CentOS Stream 8
  - Could also reproduce on Fedora 34
- Upgrading from systemd 247 to systemd 248 (latest is 251)
  - Built from the specfile in the Hyperscale SIG
    - Based on the Fedora rawhide specfile for systemd 248.2
- Kernel 5.6+
What We Saw

• Average memory for systemd-journald grew from ~17 MB to ~50 MB

• Issue reported as the systemd 248 update was ongoing
  – Memory growth did not directly correlate with systemd roll out
Probing a Host

```
# pmap -x 800
800: /usr/lib/systemd/systemd-journald

<table>
<thead>
<tr>
<th>Address</th>
<th>Kbytes</th>
<th>RSS</th>
<th>Dirty</th>
<th>Mode</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>000055908738e000</td>
<td>160</td>
<td>152</td>
<td>0</td>
<td>r-x--</td>
<td>systemd-journald (deleted)</td>
</tr>
<tr>
<td>00005590875b5000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>r----</td>
<td>systemd-journald (deleted)</td>
</tr>
<tr>
<td>00005590875b7000</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>rw---</td>
<td>systemd-journald (deleted)</td>
</tr>
<tr>
<td>00005590875b8000</td>
<td>378952</td>
<td>378796</td>
<td>378796</td>
<td>rw---</td>
<td>[ anon ]</td>
</tr>
</tbody>
</table>

...some lines omitted for brevity...
```
Was it the roll out?

- Normally systemd daemons restart into the new binary during updates.
- Noticed that systemd-journald did not!
  - This is fixed in systemd 249+
- Correctly monitoring systemd-journald 247 vs 248 confirmed the regression was due to the roll out.
Searching for Answers
Suspicious Commits?

$ git log v247..v248 --oneline --no-merges src/libsystemd/sd-journal/ src/journal

...some lines omitted for brevity...
0eaee8281d journald: when we fail to add a new entry to a journal, return the seqno
258190a0d5 mmap-cache: drop ret_size from mmap_cache_get()
104fc4be11 mmap-cache: bind prot(ection) to MMapFileDescriptor
073f50a099 mmap-cache: separate context and window list cache hit accounting
3a595c597a mmap-cache: replace stats accessors with log func
Memory Leak?

```bash
# valgrind --leak-check=full --show-leak-kinds=all /usr/lib/systemd/systemd-journald

...some lines omitted for brevity...
==1042650== LEAK SUMMARY:
==1042650==    definitely lost: 0 bytes in 0 blocks
==1042650==    indirectly lost: 0 bytes in 0 blocks
==1042650==      possibly lost: 0 bytes in 0 blocks
==1042650==    still reachable: 8,192 bytes in 2 blocks
==1042650==         suppressed: 0 bytes in 0 blocks
```
Bisect

- Started bisecting starting with the “suspicious” commits.
- Put each commits’ build of systemd-journald on separate hosts.
  - Increased logging on those hosts.
- Needed 1-2 days of data to see the regression in our charts.
Trying Things

- strace not the best tool for this job.
- Use eBPF!
  - Great for tracing and observability.
  - Meta is a founding member of the eBPF Foundation!
- Found Brendan Gregg’s page for looking at memory leaks and growth (right).
  - Started experimenting with BCC.

https://brendangregg.com/FlameGraphs/memoryflamegraphs.html
Count Calls to `malloc()`

- **BCC’s stackcount.py**
  - Used to count events and their stack traces.
- Stacks/allocations were similar between systemd-journald 247 and 248.

```
# /usr/share/bcc/tools/stackcount -U c:malloc -p 800
...cut for brevity...

b'read_one_line_file'

b'get_process_comm'

b'client_context_read_basic'

b'client_contextreally_refresh'

b'client_contextmaybe_refresh'

b'stdout_stream_log'

b'stdout_stream_line'

b'stdout_stream_found'

b'stdout_stream_scan'

b'stdout_stream_process'

b'source_dispatch'

b'sd_event_dispatch'

b'sd_event_run'

b'main'

b'__libc_start_main'

b'[unknown]'

1508
```
Memory Leak?

- **BCC’s memleak.py**  
  - Used to trace outstanding allocations.

- Stacks/allocations were similar between systemd-journald 247 and 248.

- All allocations were eventually deallocated; no leak.
What We Know So Far

• No leak!
  – Confirmed by 3 tools.
• Initial bisect did not find the blame commit.
• Allocations were similar between systemd-journald 247 and 248.
  – No extra calls to malloc() and related functions.
• Allocations tend to start from client_context_read_basic().
  – But systemd’s core functions do the allocations (e.g. read_full_virtual_file()).
• Used this information to change bisect strategy.
Another Set of Suspicious Commits

$ git log v247..v248 --oneline --no-merges src/basic/fileio*

...many lines omitted for brevity...
2ac67221bb basic/fileio: fix reading of not-too-small virtual files
f1a8a66c35 basic/fileio: use malloc_usable_size() to use all allocated memory
a9899ff358 basic/fileio: optimize buffer sizes in read_full_virtual_file()
ca79564309 basic/fileio: simplify calculation of buffer size in...
c5384931b7 fileio: add missing overflow checks to read_full_virtual_file()
b235b03138 fileio: don't use realloc() in read_full_virtual_file()
Bisect (Again)

- Another 1-2 days of data to see the regression in our charts.
Bisect Prevails

• Summary of the commit
  – Instead of allocating 4K and using `realloc()` to expand the buffer, we start with 4MB and `realloc()` to decrease the buffer.
  – Everything is freed properly and memory is returned to libc; so what’s the problem?

https://github.com/systemd/systemd/commit/2ac67221bb6270f0fbe7cbd0076653832cd49de2
Root Cause

Explained
Allocations in systemd-journald 247
Allocations in systemd-journald 247
Allocations in systemd-journald 247

 realloc(ptr, 8192)
Allocations in systemd-journald 247

Illustration of the Heap

```c
realloc(ptr, 8192)
```
Allocations in systemd-journald 247

Illustration of the Heap

realloc(ptr, 16384)
Allocations in systemd-journald 247
Allocations in systemd-journald 247

Illustration of the Heap

realloc(ptr, 32768)
Allocations in systemd-journald 247

Illustration of the Heap

 realloc(ptr, 32768)
Allocations in systemd-journald 248
Allocations in systemd-journald 248

Illustration of the Heap

realloc(ptr, 4096)
Allocations in systemd-journald 248

 realloc(ptr, 4096)
Allocations in systemd-journald 248
Allocations in systemd-journald 248
Allocations in systemd-journald 248

realloc(ptr, 4096)
Allocations in systemd-journald 248

Illustration of the Heap

realloc(ptr, 4096)
Allocations in systemd-journald 248
Root Cause Explained

Allocations in systemd-journald 248

Illustration of the Heap

realloc(ptr, 4096)
Allocations in systemd-journald 248

Illustration of the Heap

realloc(ptr, 4096)
Allocations in systemd-journald 248

Illustration of the Heap

free(ptr)
Fix Merged

• Summary of the fix:
  – Partially revert back to previous behavior.
  – Allocate 4K and use realloc() to expand the buffer as needed.

• Meta was the first to notice and fix it!

https://github.com/systemd/systemd/commit/5aa55d841249f057fd69e50cf12a52e9781a6ce
Hindsight
Rate of Change of Anonymous Memory

systemd 247

systemd 248
mtrace()

- Part of glibc.
- Function call:
  - Insert at the beginning of the program to record memory allocation and deallocations.
  - Records the data to a text file.
- Command line tool:
  - Uses the text file and binary to tell you about unfreed memory.
Massif: a heap profiler

timeout 20m /usr/local/bin/valgrind --tool=massif --time-unit=B /usr/lib/systemd/systemd-journald

systemd-journald 247

systemd-journald 248
Another Way?

- Instead of `realloc()`, do `malloc()` and `memcpy()`.
- The reallocation would be copied to create less fragmentation.
- More `malloc()` calls, more copies.

Memory Usage (MB) Over 24 Hours
Takeaways

• Invest in monitoring, logging, and visualizations!
• Usable stack traces are a blessing.
  – Needs frame pointers.
• Always be willing to learn and pick up new tools!
  – eBPF is amazing: BCC, bpftrace, etc.
  – Valgrind is more than memcheck: massif, callgrind, helgrind, etc.
Questions?

THANK YOU FOR YOUR TIME

Anita Zhang
github.com/anitazha
twitter.com/the_anitazha
Extra Slides
$ cat /etc/systemd/journald.conf

[Journal]
ForwardToSyslog = true
RuntimeMaxUse = 10M
Storage = volatile