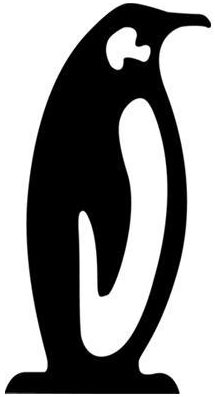


東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

BPF Based Telemetry, Metrics, and Accounting

on Android

東京 2025
LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

What type of things do we collect?

- Shared memory use (dmabuf)
  - Global (originally sysfs, now BPF)
  - Per-application (procfs)
- Amount of network use
  - Per-application (netd w/BPF)
- Amount of CPU time
  - Per-application, per-state (uid_sys_stats OOT driver)
- Amount of file I/O
  - Per application (BPF program)
- Wakeup source stats
  - Originally debugfs, then sysfs, soon BPF?)

Existing data sources and challenges

- debugfs
  - Not mounted in user builds for security
  - No stable API

- tracefs
  - Good, but...
  - Occasional hesitation adding tracepoints upstream
    - Result: Out of tree tracepoint patches (maintenance)
  - Tracepoints become part of kernel UAPI
    - Commit to an interface - modifications potentially challenging
  - Frequent event parsing
    - BPF enables always-on event-driven telemetry

      vs

      periodic polling / parsing (`trace_pipe`)
    - Ok for continuous monitoring, not ideal for rare reads that can be computed on-demand

Existing data sources and challenges

- procfs
  - Hesitance to add new files upstream
    - Result: out of tree patches (maintenance)
  - Part of stable kernel UAPI
    - Commit to an interface - modifications potentially challenging

Existing data sources and challenges

- **sysfs**
  - **Single value per file** = lots of files

```
raven:/sys/kernel/dmabuf/buffers # ls
1     1569  1656  205   212   2224  2241  2260  2278  2295  25    2575  26    2716  2939  32    510   88
10    1576  1657  2050  213   2225  2242  2261  2279  230   251   2578  260   2717  2940  33    676   89
1000  1577  1862  2051  214   2226  2243  2262  228   235   2517  2579  2600  2718  3002  34    677   9
105   158   1863  2052  215   2227  2244  2263  2280  2358  2518  258   2601  2719  3003  358   679   90
106   159   197   2053  216   2228  2245  2264  2281  2359  252   2586  2604  2720  3004  359   680   92
107   1606  198   2054  217   2229  2246  2265  2282  236   2521  2587  2605  277   3005  36    755   976
108   1607  1985  2055  218   223   2247  2266  2283  2360  2522  2588  2606  278   3006  37    756   977
109   1642  1986  206   219   2230  2248  2267  2284  2361  255   2589  2607  279   3007  38    773   988
11    1643  1987  2060  220   2231  2249  2268  2285  2384  256   259   2609  280   3008  39    774   989
110   1644  199   2061  221   2232  2250  2269  2286  2385  2566  2590  261   281   3009  399   799   991
111   1645  200   2062  2214  2233  2251  227   2287  2386  2567  2591  2610  282   3010  40    8     992
112   1646  203   2063  2215  2234  2252  2270  2288  2387  2568  2592  262   283   3011  400   800   993
113   1647  204   2064  2216  2235  2253  2271  2289  2483  2569  2593  263   284   3012  401   801   994
114   1650  2044  2065  2217  2236  2254  2272  229   2484  257   2594  264   287   3013  41    802   995
12    1651  2045  207   222   2237  2255  2273  2290  2485  2570  2595  265   288   3014  451   803   996
13    1652  2046  208   2220  2238  2256  2274  2291  2495  2571  2596  266   2884  3015  452   804   999
1319  1653  2047  209   2221  2239  2257  2275  2292  2496  2572  2597  2713  2885  308   453   82
1320  1654  2048  210   2222  224   2258  2276  2293  2497  2573  2598  2714  2929  309   454   85
1568  1655  2049  211   2223  2240  2259  2277  2294  2498  2574  2599  2715  2930  31    509   86
raven:/sys/kernel/dmabuf/buffers # find | wc -l
1018
```

Existing data sources and challenges

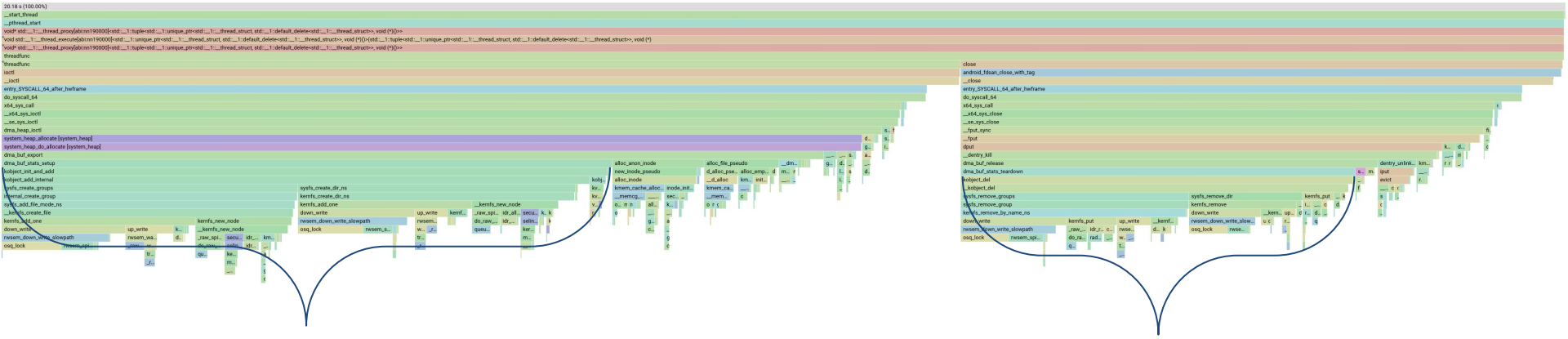
- sysfs
  - **Single value per file** = lots of files

```
raven:/sys/class/wakeup # ls
wakeup0    wakeup110  wakeup123  wakeup136  wakeup17  wakeup30  wakeup44  wakeup57  wakeup7   wakeup82  wakeup95
wakeup1    wakeup111  wakeup124  wakeup137  wakeup18  wakeup31  wakeup45  wakeup58  wakeup70  wakeup83  wakeup96
wakeup10   wakeup112  wakeup125  wakeup138  wakeup19  wakeup32  wakeup46  wakeup59  wakeup71  wakeup84  wakeup97
wakeup100  wakeup113  wakeup126  wakeup139  wakeup2   wakeup33  wakeup47  wakeup6   wakeup72  wakeup85  wakeup98
wakeup101  wakeup114  wakeup127  wakeup14   wakeup20  wakeup34  wakeup48  wakeup60  wakeup73  wakeup86  wakeup99
wakeup102  wakeup115  wakeup128  wakeup140  wakeup21  wakeup35  wakeup49  wakeup61  wakeup74  wakeup87
wakeup103  wakeup116  wakeup129  wakeup141  wakeup22  wakeup36  wakeup5   wakeup62  wakeup75  wakeup88
wakeup104  wakeup117  wakeup13   wakeup142  wakeup23  wakeup37  wakeup50  wakeup63  wakeup76  wakeup89
wakeup105  wakeup118  wakeup130  wakeup143  wakeup24  wakeup38  wakeup51  wakeup64  wakeup77  wakeup9
wakeup106  wakeup119  wakeup131  wakeup144  wakeup25  wakeup39  wakeup52  wakeup65  wakeup78  wakeup90
wakeup107  wakeup12   wakeup132  wakeup145  wakeup26  wakeup40  wakeup53  wakeup66  wakeup79  wakeup91
wakeup108  wakeup120  wakeup133  wakeup146  wakeup27  wakeup41  wakeup54  wakeup67  wakeup8   wakeup92
wakeup109  wakeup121  wakeup134  wakeup15   wakeup28  wakeup42  wakeup55  wakeup68  wakeup80  wakeup93
wakeup11   wakeup122  wakeup135  wakeup16   wakeup29  wakeup43  wakeup56  wakeup69  wakeup81  wakeup94
raven:/sys/class/wakeup # find -L -maxdepth 2 2>/dev/null | wc -l
1837
```

Existing data sources and challenges

- sysfs
  - Single value per file = lots of files
    - **Slow to read (foreach file: open/read/close)**
    - ***Very* non-atomic view**
      - "it is now very easy for the global state of all kernel wakesources to change in the middle of trying to read them"
      - "wakesources and their stats can be created and destroyed while trying to walk the symbolic tree of stats nodes"
    - **sysfs/kernfs lock contention: slow creation / deletion / allocations**
      - "dma_buf_stats_setup() is responsible for 39% of single-page buffer creation duration, or 74% of single-page dma_buf_export() duration when stressing dmabuf allocations and frees."

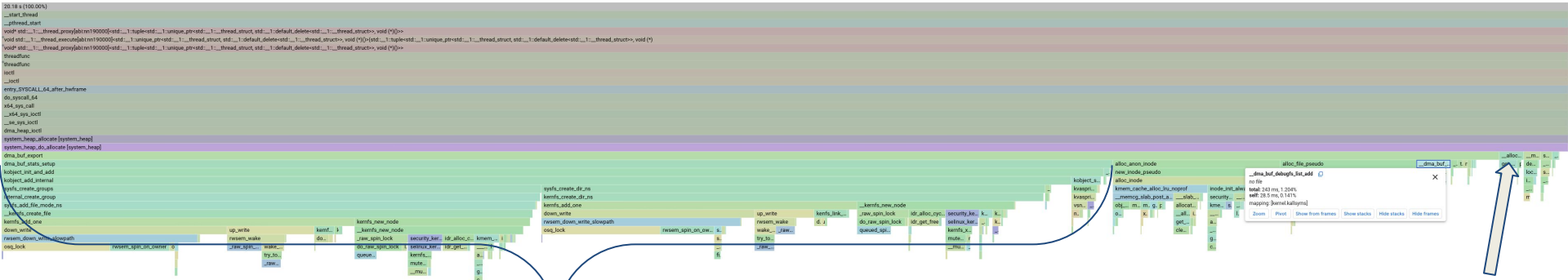
Dmabuf allocation + free w/CONFIG_DMABUF_SYSFS_STATS



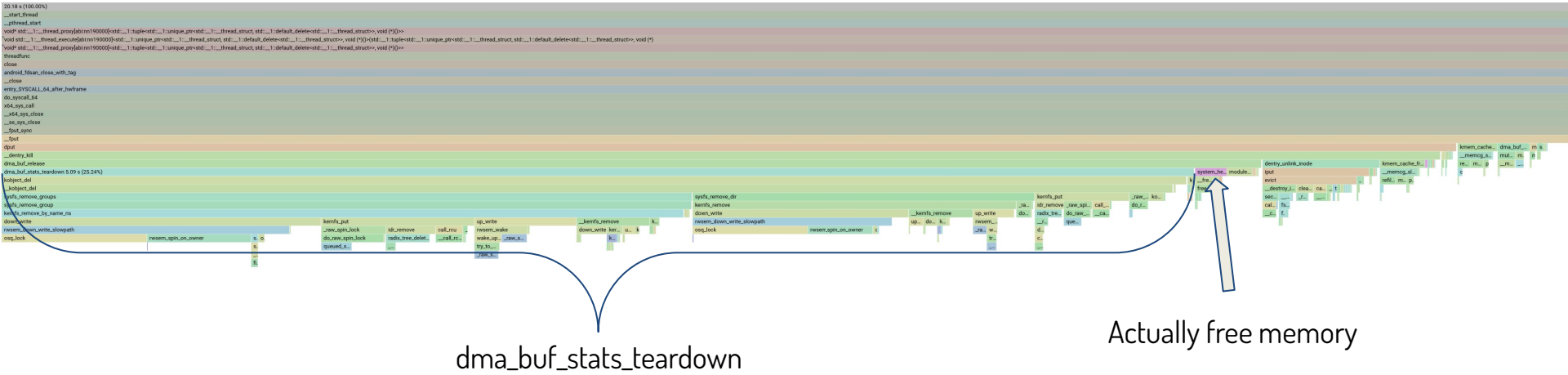
dma_buf_stats_setup

dma_buf_stats_teardown

Dmabuf allocation w/CONFIG_DMABUF_SYSFS_STATS



Dmabuf free w/CONFIG_DMABUF_SYSFS_STATS



Switching from sysfs to BPF

- Define our own UAPI
  - We can change it more easily
  - text (BPF iterator) or binary (BPF map)

- Low or 0 continuous overhead
  - BPF iterators can be cheaper than BPF programs attached to tracepoints

- Need BPF CO-RE (Compile Once - Run Everywhere)
  - Available in 26Q1

- For dmabuf: https://lore.kernel.org/all/20250522230429.941193-1-tjmercier@google.com/
  - Disable CONFIG_DMABUF_SYSFS_STATS (beginning android17-6.18)
- For wakeup sources: https://lore.kernel.org/all/20251204025003.3162056-1-wusamuel@google.com/

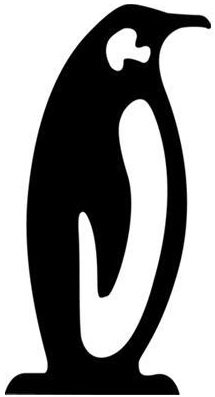
BPF oopsies

- Lack of kernel support = boot time regression
  - Full symbol search
- BPF iterator: multiple reads() for “large” output
  - In general: requires unlocking = state can change
  - Currently limited to PAGE_SIZE << 3 bytes

Future BPF work

- Put all of `dmabuf_dump` in BPF?
  - Sould be possible using existing *traditional* task-file and task-vma iterators
  - Single program with open-coded iterator would require open-coded task-file iterator support which was NAKed
- Lock Contention
  - Using `contention_begin` from 5.19
  - Requires symbolizing dynamically allocated locks (e.g. `mm->mmap_lock`)
- Wakeup sources
  - Ideally add a CONFIG_ to allow disabling sysfs file generation
- Detailed metrics on memory reclaim
  - Latency (including direct reclaim, by app-state / oom_adj)

Discussion



東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025