第三届 eBPF开发者大会
www.ebpftravel.com

内存可观测：eBPF技术在Android系统内存剖析及调优实践

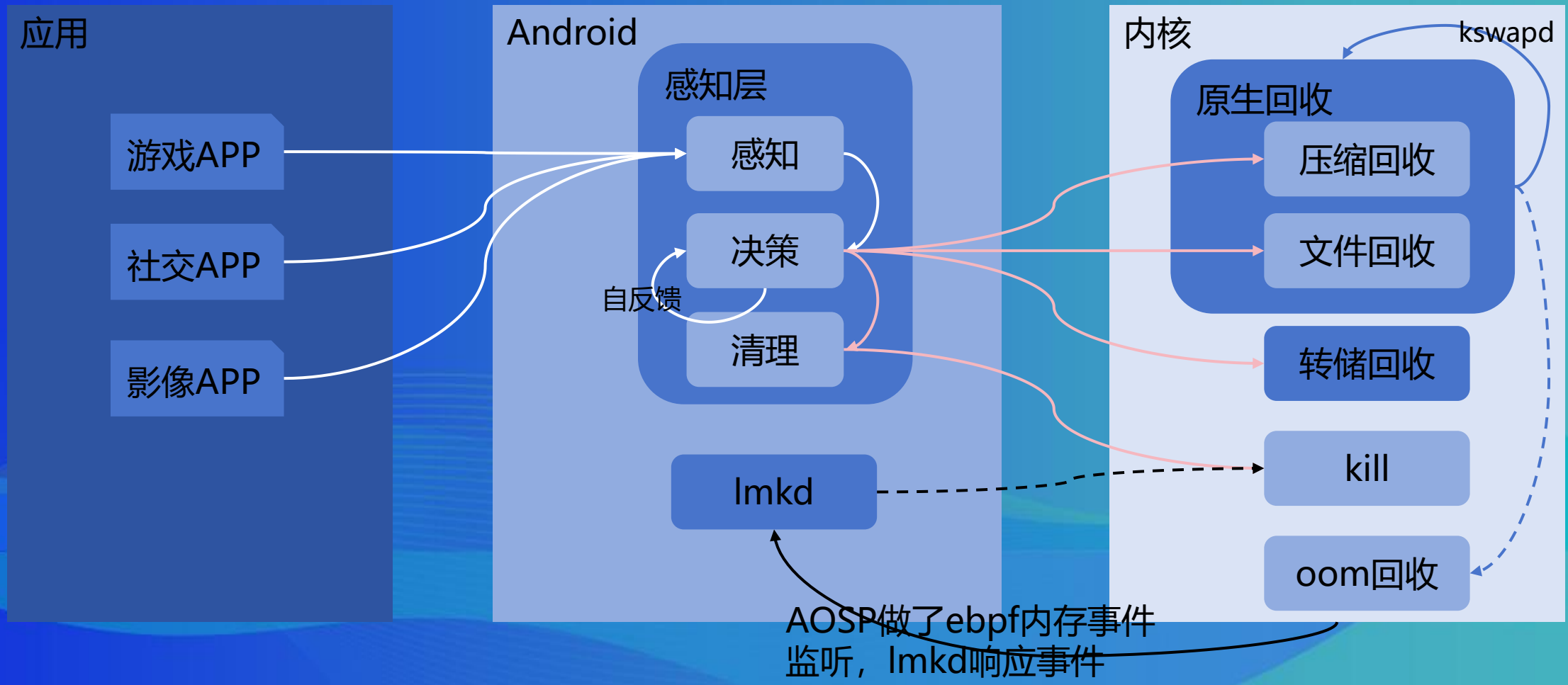
荣耀终端 | OS Kernel Lab
伊鹏翔 林琨力 王鑫

中国·西安

作者介绍

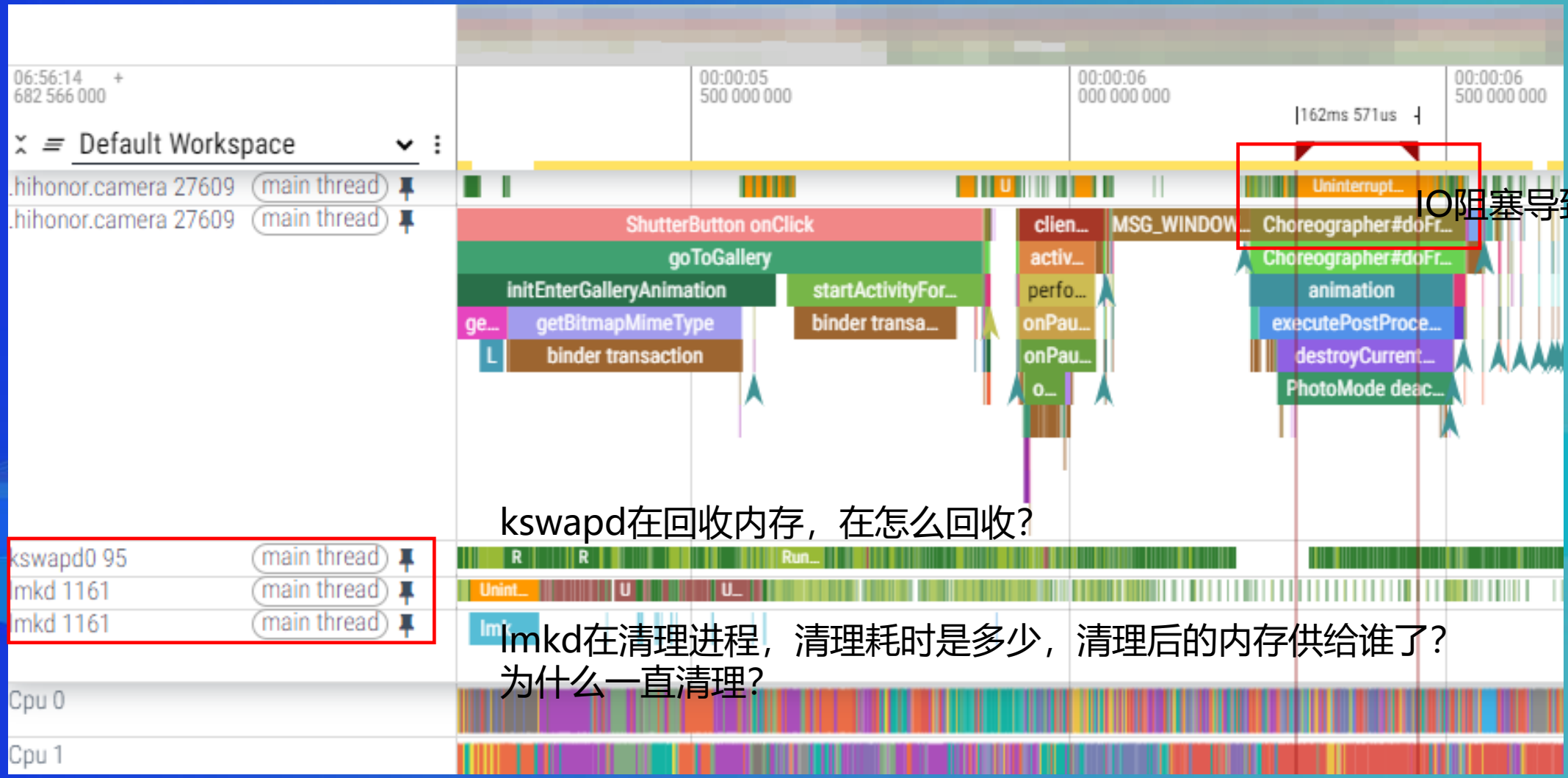
- 伊鹏翔：荣耀内核开发专家（内存方向），负责内存方案开发及演进，主要在内存及性能方面工作。
- 林琨力：荣耀内核工程师，中科院博士，主导开发多个内存优化方案如：dmabuf压缩等方案。
- 王鑫：荣耀内核工程师，负责内存性能优化工作。

终端设备内存性能现状



内存管理基于事件触发，触发后决策清理/回收/转储等策略。而性能事件产生后，我们只能看到结果，无法评估策略的合理性。

问题：性能问题发生后只有内存事件，怎么性能调优？



性能问题采集到离散内存数据

dumpsys meminfo

```
289
290 Total PSS by OOM adjustment:
291     2,136,268K: Native                                                  (   944,883K in swap)
292         691,003K: vendor.honor.hardware.camera.dme-V1-ndk-service (pid 4255)  (    10,304K in swap)
293         391,082K: vendor.qti.camera.provider-service_64 (pid 1994)          (   265,540K in swap)
294         302,895K: surfaceflinger (pid 2044)                                 (    41,428K in swap)
295          63,528K: aptouch_daemon (pid 2197)                                 (    52,376K in swap)
296          28,451K: vendor.qti.hardware.display.composer-service (pid 1997)   (    17,764K in swap)
297          24,890K: dubaid (pid 2653)                                         (    15,120K in swap)
298          24,719K: media.hwcodec (pid 2005)                                  (    23,852K in swap)
299          24,243K: android.hardware.bluetooth@aidl-service-qti (pid 1898)    (    23,552K in swap)
300          20,965K: audiohalservice.qti (pid 1911)                            (    20,940K in swap)
301          19,569K: audioserver (pid 2030)                                    (    17,744K in swap)
302          18,473K: mediaserver (pid 2665)                                    (    15,652K in swap)
303          16,566K: vendor.honor.hardware.aoservice-service (pid 1927)        (    16,328K in swap)
304          15,363K: qcrilNrd (pid 2616)                                       (    12,272K in swap)
305          15,076K: hiview (pid 4770)                                         (     8,500K in swap)
306          14,327K: hiview (pid 2834)                                         (     5,652K in swap)
307          12,495K: vendor.honor.hardware.biometrics.hwfacerecognize.aidlservice (pid 2628) (    12,148K in
```

/proc/vmstat

```
037
038 Number of blocks type     Unmovable      Movable      Reclaimable          CMA      HighAtomic      Isolate
039 Node 0, zone   Normal         1563         1022           35           260            4            0
040 nr_free_pages 104887
041 nr_zone_inactive_anon 207291
042 nr_zone_active_anon 151740
043 nr_zone_inactive_file 269286
044 nr_zone_active_file 117924
045 nr_zone_unevictable 24026
046 nr_zone_write_pending 3077
047 nr_mlock 23682
048 nr_bounce 0
049 nr_zspages 404668
050 nr_free_cma 4034
051 nr_inactive_anon 207291
052 nr_active_anon 151740
053 nr_inactive_file 269286
054 nr_active_file 117924
055 nr_unevictable 24026
056 nr_slab_reclaimable 64892
```

/proc/meminfo

```
607 MemTotal:       11477792 kB
608 MemFree:          421112 kB
609 MemAvailable:    3871004 kB
610 Buffers:            1628 kB
611 Cached:          1682556 kB
612 SwapCached:        12388 kB
613 Active:          1077428 kB
614 Inactive:        1906328 kB
615 Active(anon):     606168 kB
616 Inactive(anon):   829296 kB
617 Active(file):     471260 kB
618 Inactive(file):  1077032 kB
619 Unevictable:       96104 kB
620 Mlocked:           94728 kB
621 SwapTotal:      12582908 kB
622 SwapFree:        7388212 kB
623 Dirty:             12128 kB
624 Writeback:             0 kB
625 AnonPages:       1393760 kB
626 Mapped:           714456 kB
627 Shmem:             43020 kB
628 KReclaimable:    2230944 kB
629 Slab:             978864 kB
630 SReclaimable:     259520 kB
631 SUnreclaim:       719344 kB
632 KernelStack:       77056 kB
633 ShadowCallStack:       0 kB
634 PageTables:       239560 kB
635 SecPageTables:         0 kB
636 NFS_Unstable:          0 kB
637 Bounce:                0 kB
638 WritebackTmp:          0 kB
639 CommitLimit:     18321804 kB
640 Committed_AS:    172949784 kB
```

在性能故障时，如何将离散的内存数据变成连续数据，优化内存管理策略？

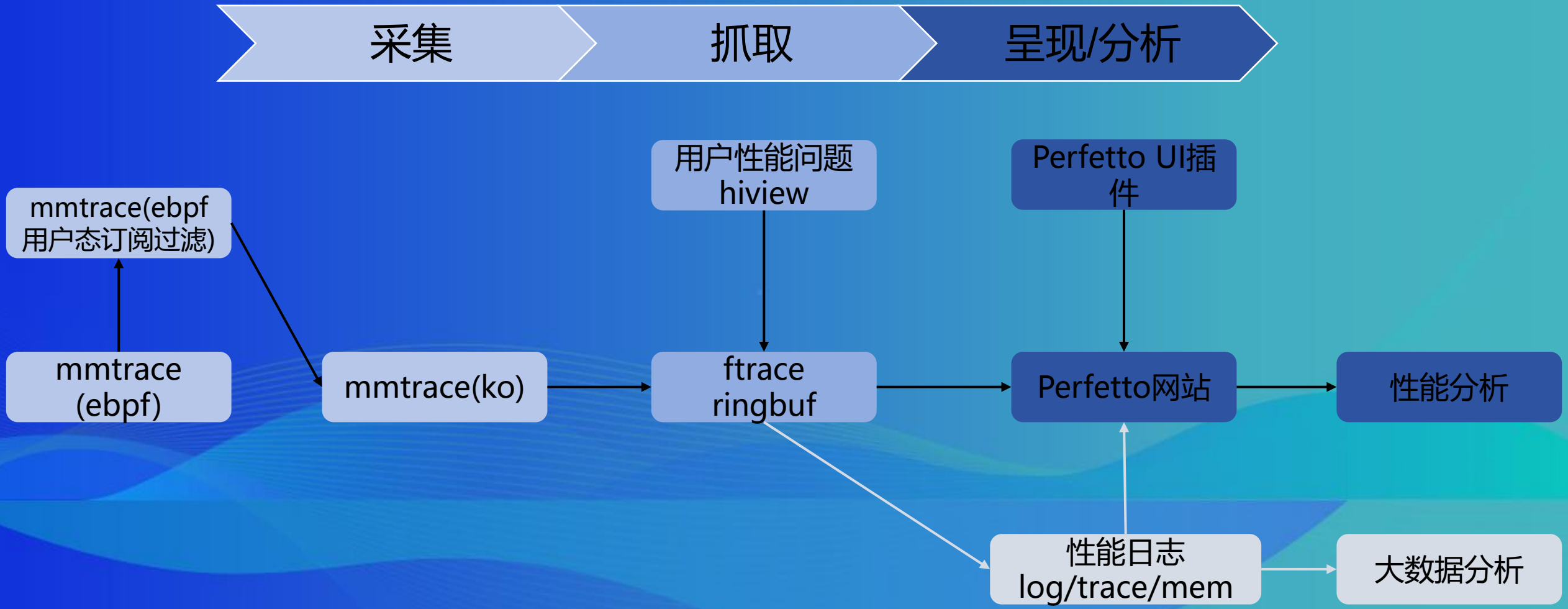
mmtrace主要工作

mmtrace实现了低开销的内存资源实时采集方案并能关联性能故障，极大方便了定位内存供给/回收机制，系统低内存高IO等问题。

【主要内容】
在本次分享中，我们将重点介绍：
1.通过eBPF等技术实现进程内存，整机内存，内存事件的抓取及聚合，单次采集小于1KB，大幅减少了采集量。
2.如何实现与性能故障时序关联，可视化内存资源与性能关系。
3.通过实际案例展示如何通过本项目对内存回收策略优化，实现性能调优。

【实践成效】
1. 事件触发，故障数据快速关联
2. 开销低，能部署到Android用户环境，实现规模商用

mmtrace设计思路



mmtrace采集：低开销连续采集

阈值（如：时间窗内超过x次、内存压力上报等）
类型（如：vip线程、render线程、cameraprovider等）
重点名单（如：systemui、launcher等）

内存tracepoint:
mm_vmscan_direct_reclaim_begin
mm_vmscan_direct_reclaim_end
mm_vmscan_kswapd_wake
kprobe & kretprobe:
dma_heap_buffer_alloc
kgsl_allocate_user
__alloc_pages 等

ko模块连续读取内核信息：
系统状态/关键进程状态/memcg状态
file/anon/dmabuf/kgsl/mlock等
读取时机：1.定时读取 2.ebpf触发
3.vendor_hook事件 4.主动触发

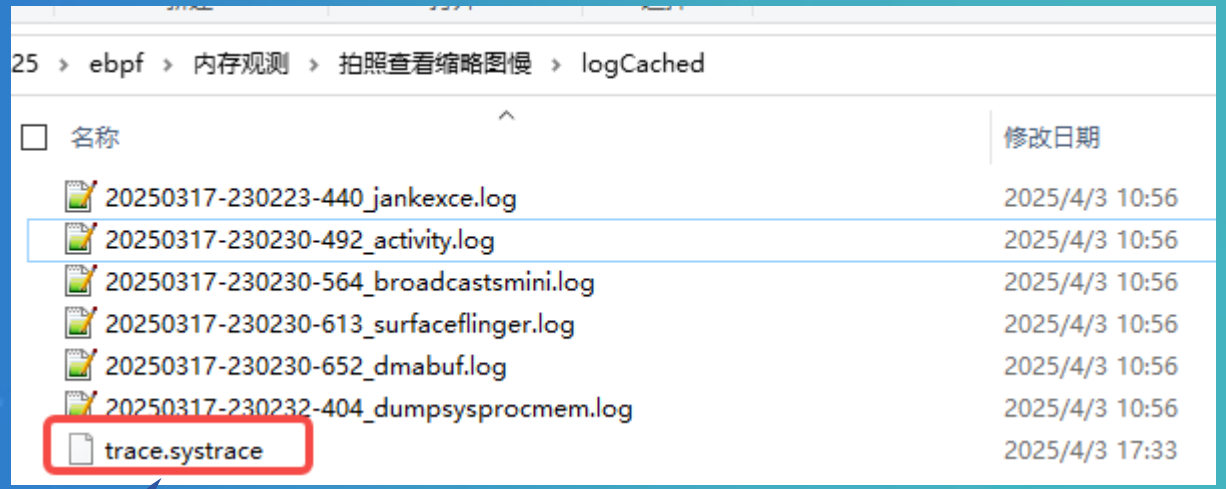
示例：

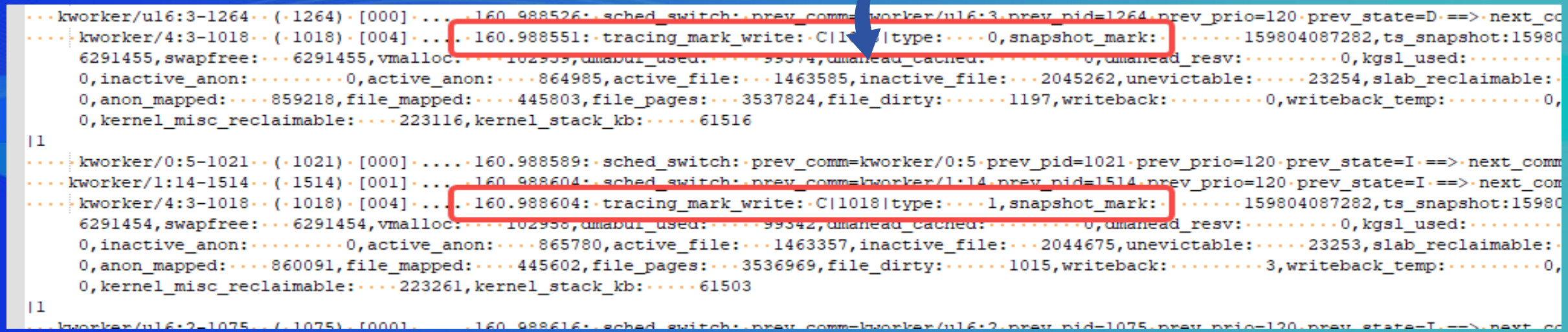
监测事件：用户使用某应用，产生了多次direct_reclaim

流量控制：应用的vip线程达到了x次direct_reclaim，往mmtrace写入事件

信息采集：mmtrace采集预先设定内存信息，如果多次发生会采集多次，也有定时采集。

mmtrace抓取：用户性能事件触发

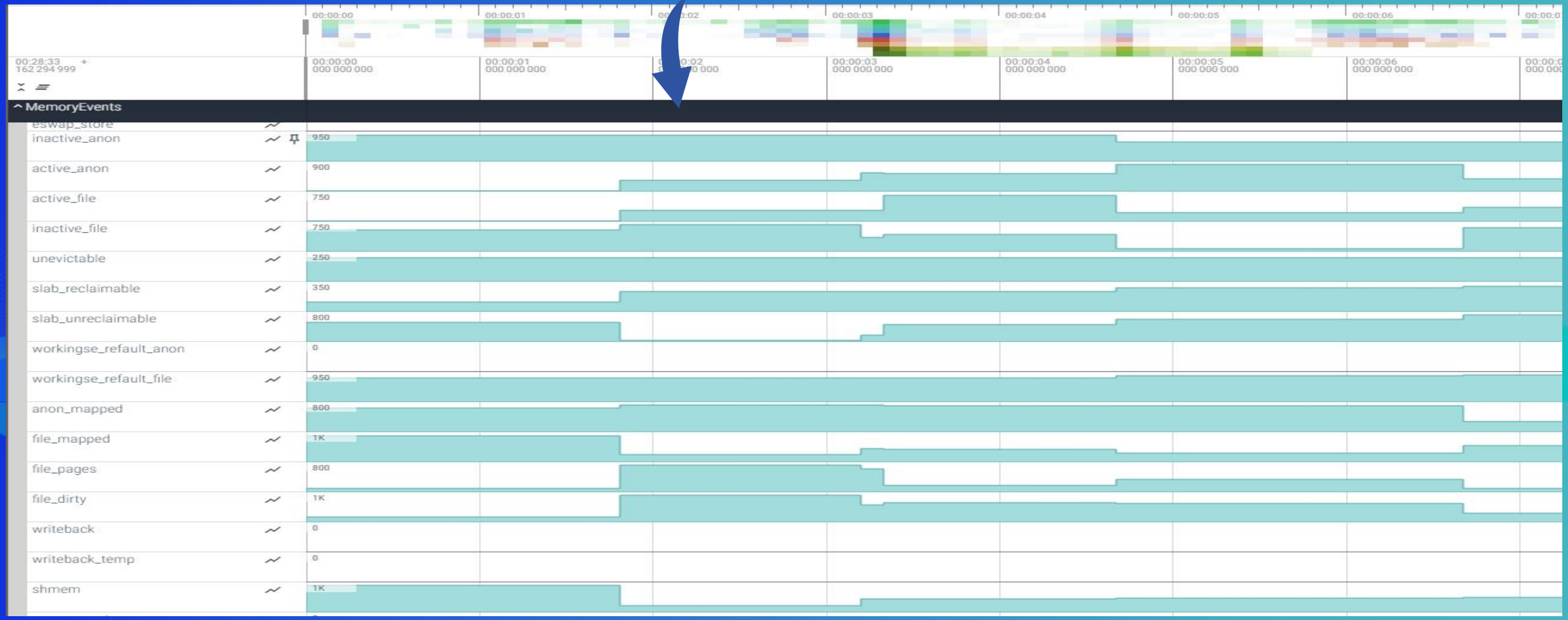




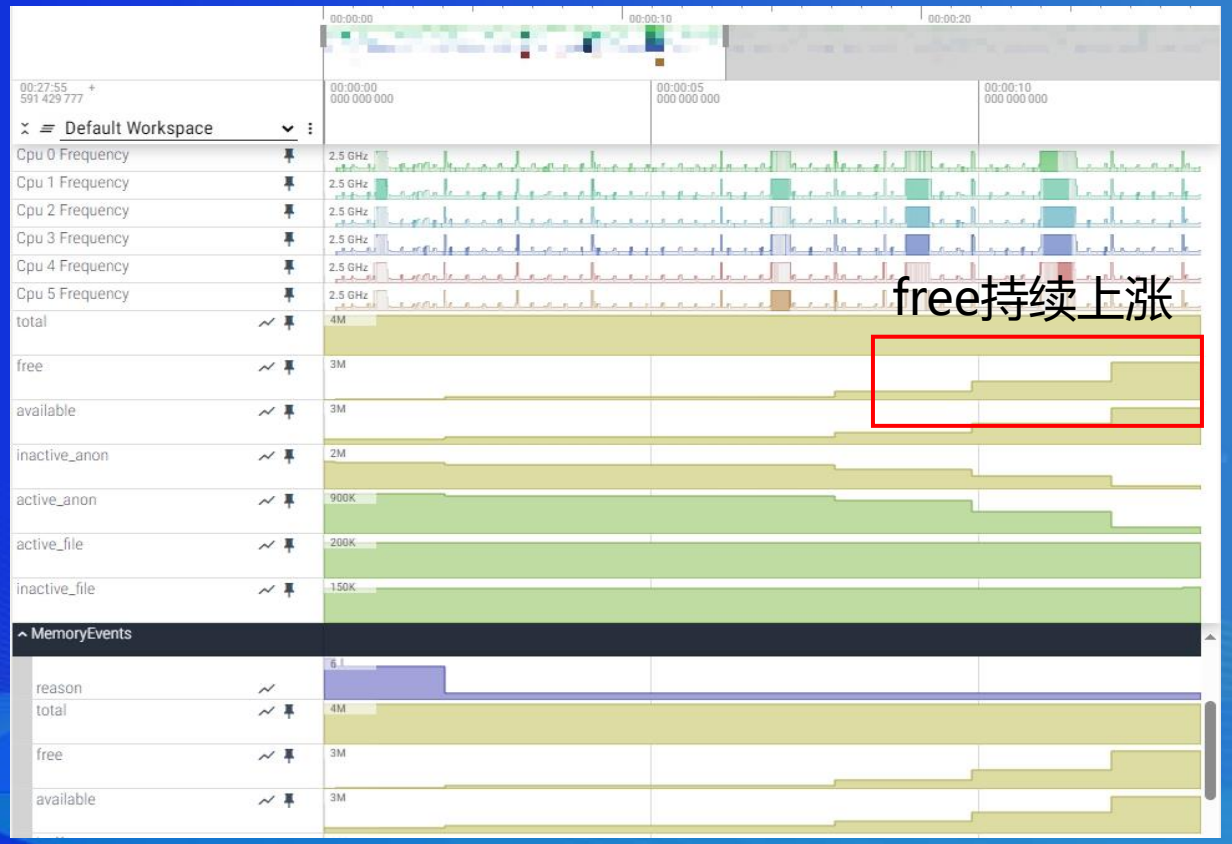
mmtrace呈现&分析：可视化分析问题

Perfetto UI插件

ftrace ringbuf → Perfetto网站 → 性能分析



mmtrace实践1：内存过度回收导致的性能问题

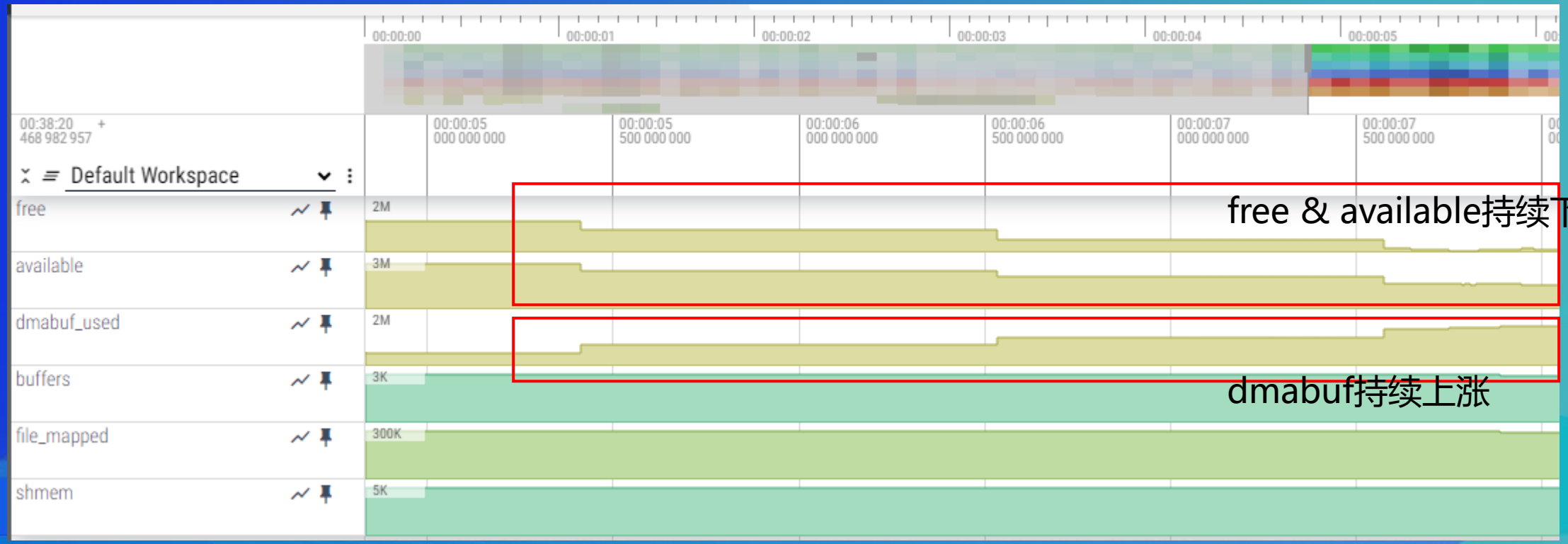



```c
/* don't abort memcg reclaim to ensure fairness */
if (!root_reclaim(sc) && !bypass)
    return false;

if (sc->nr_reclaimed >= max(sc->nr_to_reclaim, compact_gap(sc->order)))
    return true;

trace_android_vh_mglru_should_abort_scan_order(sc->order, &bypass);
/* check the order to exclude compaction-induced reclaim */
if ((!current_is_kswapd() || sc->order) && !bypass)
    return false;

mark = sysctl_numa_balancing_mode & NUMA_BALANCING_MEMORY_TIERING ?
       WMARK_PROMO : WMARK_HIGH;

for (i = 0; i <= sc->reclaim_idx; i++) {
    struct zone *zone = lruvec_pgdat(lruvec)->node_zones + i;
    unsigned long size = wmark_pages(zone, mark) + MIN_LRU_BATCH;

    if (managed_zone(zone) && !zone_watermark_ok(zone, 0, size, sc->reclaim_idx, 0))
```

内存持续被回收，free上涨，kswapd占用cpu资源导致性能卡顿。

修改链接：.../vmscan.c · Gerrit Code Review (googlesource.com)

mmtrace实践2：dmabuf异常占用导致的性能问题



前台应用滑动申请内存卡顿， perfetto显示free & available持续下降。
而dmabuf占用又持续升高，说明dmabuf内存有明显异常。再查看应用内存占用，可以快速定界到应用。

总结

mmtrace实现了一个低开销内存采集工具，适合部署在beta/商用手机。

通过mmtrace可以关联性能和内存问题，呈现出内存变化过程，变离散数据为连续数据，大幅提升性能开发人员的工作效率。

Thank you