Android, CMA and Nexus Low Memory Killer Application Notes

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

Revision History

Revision	Date	Change Description
v0.1	05-05-2019	Sam Irvine - Initial release

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

Common Customer Issues

Android Low Memory Killer (LMK) engages because of zone balance issues

In Broadcom Android O based systems, LMk can engage on a per-zone basis. This can cause LMK to engage even when the system has plenty of free memory to satisfy paging requests.

Symptoms

```
cat /proc/zoneinfo
```

If you notice ZONE_DMA/ZONE_NORMAL lingering near its low watermark while ZONE_HIGHMEM has lots of free pages, this is likely the culprit.

Solution
Update the vmalloc argument to tune ZONE_DMA and ZONE_HIGHMEM to approximately equal sizes. Known “good” values:

cypress	vmalloc=506m

Nexus LMK engages and kills a background application even with free system memory

Symptoms
Background applications are killed even when the system shows plenty of free memory. Logcat will show the following line when this occurs:

3824:06-22 18:15:33.721 3566 3592 I nxserver:
oom-kill[27]:16725::mmu:178MB::nxmem:0MB::gfxo:0MB::score:1

Where 16725 is the PID of the process being killed.

Discussion
Some Broadcom platforms use CMA memory regions to share memory between the drivers/HW buffers and Android userland processes. This partitioning causes problems in some use cases. When the driver cannot allocate memory from CMA, the NEXUS LMK within nxserver process

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

will engage to kill off processes that uses large amounts of CMA memory, this could occur periodically and on demand.

The Nexus LMK periodically queries the nx_ashmem driver and puts together a list of processes, their CMA usage and OOM_ADJ score. When the process goes into the background, it's OOM_ADJ score is raised. If a background process in this list uses more CMA than the limit defined in ro.nx.lmk.bg, it will be killed to free up CMA memory in the order of PIDs with highest OOM_ADJ score first.

Another route to trigger Nexus LMK is by demand. This occurs when gralloc fails to allocate CMA memory for graphics, or when omx fails to create video decoder. In either case, gralloc or omx will request nxserver via HIDL interface to perform LMK. Once this has been completed, another attempt to allocate memory will be made.

Finally, the Nexus LMK could also be triggered when too many graphical applications are running in the system. When this occurs, the user will fail to open new applications due to failure to create additional graphics handle, this will directly impact user experience. To deal with this problem, Nexus LMK periodically monitors applications holding graphics client handle and kills background applications with the highest OOM_ADJ score when a default limit is exceeded. By default this limit is set 14. This default is chosen because each graphics client handle occupy 1MB of CMA-1 space and this region is currently set to 16Mb

Solution
To further understand the CMA usage trends of the application, the ro.nx.lmk.bg can be temporarily set to a very large number (300mb) to prevent Nexus LMK from killing the app, then change #define NX_CLIENT_USAGE_LOG to 1 in the file vendor/broadcom/bcm_platformnxif/nxserver/treble/nxserver.cpp. This will allow nxserver to periodically print out CMA usage in logcat so the developer can understand the maximum CMA usage of the application as it is being stressed.

Once the CMA usage has been profiled, the application developer should be sure to release graphics memory and deallocated video decoders when it goes into the background so that the CMA usage will stay below the limit set in ro.nx.lmk.bg.

Common Customer Questions

Which heaps are backed by CMA and which are backed by BMEM?

- In p-based systems, /proc/brcm/core displays this information.
- In o-base systems

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

- Main, secure (CRR): BMEM
- Picture Buffer heaps: CMA
- GFX (graphics): CMA

Why is a CMA allocation failing when there is plenty of free CMA?

The CMA allocator is subject to internal fragmentation. This fragmentation is grossly visible via the /sys/kernel/debug/cma/cma-X/bitmap entry. Each bit represents a free/allocated block 2^order_per_bit * PAGE_SIZE bytes in size (see /sys/kernel/debug/cma/cma-X/order_per_bit). For this system, order_per_bit == 0, so each bit represents a page free/allocated.

There is also fragmentation at a lower layer that is invisible to the CMA subsystem. If a page is allocated to a Linux address space, the address space may reject requests requests to move pages. This type of pinning is not visible to the system until the actual move to allocate a CMA block is made, and pages are actively migrated away. Some return paths from this pinning do not return -EAGAIN, and thus the CMA system does not know to retry its operation and aborts. The broadcom DTU driver requires 2MB continuous regions of free memory. A single 4k page can thus render an entire 2MB region unusable.

System Level Tracking Memory

dumpsys meminfo

- Free Ram
  - cached pss: pss from ‘cached’ OOM adjustment category
  - cached kernel: /proc/meminfo:Buffers + /proc/meminfo:SReclaimable + /proc/meminfo:Cached - /proc/meminfo:Mapped
    - /proc/meminfo:Buffers - Buffer cache
    - /proc/meminfo:SReclaimable - Shrinkable slab cache
    - /proc/meminfo:Cached - Active(file)+Inactive(file)
      - Evictable read only code pages
    - /proc/meminfo:mapped
      - Why doesn’t Mapped == Active(file)+Inactive(file) ?
        - When a process is terminated it doesn’t need to remove entries from the file cache. Linux will reclaim that memory from the LRU lists if the files remain inactive.
    - Question: I see kernel cached go up considerably and free memory reduced. What can be done to reduce kernel file caching and free up memory?
      - Nothing. Memory in the file cache is considered ‘free’ since kswapd will reclaim inactive entries on demand as zone memory

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

reaches the low watermark. You don't need to do anything to free up this cached memory; Linux does it for you on demand.

- Used RAM:
  - Used pss = totalPss - cached pss from above.
  - kernel = /proc/meminfo:SUnreclaim + /proc/meminfo:KernelStack + /proc/meminfo:PageTables + /proc/meminfo:Shmem + /proc/meminfo:VmallocUsed
    - /proc/meminfo:SUnreclaim - Unreclaimable slab entries (see /proc/slabinfo for detailed breakdown)
    - /proc/meminfo:KernelStack - Memory dedicated to the kernel stack(s). Note that each thread of a process has a kernel stack at 8k (32bit) or 16k (64bit). On android `ps -AT | wc` * 8K should equal KernelStack
    - /proc/meminfo:PageTables - Memory used for page directories and page table entries
    - /proc/meminfo:Shmem - Shared memory (&pipes?)
    - /proc/meminfo:VmallocUsed - Kernel calls to vmalloc
    - Note: Nexus makes calls to kmalloc and vmalloc for general purpose allocation. As such, general purpose NEXUS memory is accounted for here. This excludes large allocations by Nexus managed heaps.
  - Free: Actual free memory reported by Linux (`free -h`). "Free memory" is memory that resides in the buddy lists (/proc/pagetypeinfo)
- LostRam: RAM that cannot be accounted for.
  - Actually computed as Total - Accounted
  - CMA allocated memory is not accounted for by dumpsys meminfo'

Lower Level Tools

/proc/<pid>/smaps (also `pmap <pid>`)

- Information on a processes memory map. Used to compute pss by 'dumpsys meminfo'

/proc/meminfo

- Generalized Linux VM status information. Used by 'dumpsys meminfo' to compute various values.

/proc/pagetypeinfo

- Per node, per zone, per migrate type buddy list accounting
- Useful for examining statistics on 'alloc_pages' calls

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

/proc/zoneinfo

- Zone specific memory management numbers. Important for gauging VM pressure.

vmstat

- Effective gauge for VM pressure.
    - When bi rate is high, this indicates the file page cache turnover rate is high and the VM doesn't have enough memory to service the working set.

CMA /sys/kernel/debug entries

/sys/kernel/debug/cma/cma-X/count

Total count of CMA blocks. The total size of this CMA pool should be:

```
count * 2^order_per_bit * PAGE_SIZE
```

/sys/kernel/debug/cma/cma-X/used

The number of CMA blocks used. The total used CMA size should be:

```
Used * 2^order_per_bit * PAGE_SIZE
```

/sys/kernel/debug/cma/cma-X/buffers

A listing of CMA allocation chunks

/sys/kernel/debug/cma/cma-X/order_per_bit

The size of a CMA accounting block; 2^order_per_bit * PAGE_SIZE is the minimum allocation granularity for CMA

/sys/kernel/debug/cma/cma-X/bitmap

A bitmap indicating free / allocated blocks from this CMA pool. This is useful for grossly depicting CMA memory fragmentation.

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.

Legacy Style Operator Applications

The legacy app model for operators assumes an operator's application owns as much memory as it needs or that it can use a fixed carve-out of memory and leave the remaining memory to other applications. These assumptions are not true in Android. Android prioritizes applications and service memory footprints according to its own model [1,2]. An application placed in the background is subject to eviction per Android's priority model.

Additionally, Android recommends using CMA over fixed, unshared carve outs [3]. Broadcom has adopted this recommendation and shares driver memory with applications using the CMA carveout. However, the Android LMK cannot differentiate between driver and user memory. It also has no mechanism to track the CMA pool size or status. As such, a Nexus LMK implementation applies an LMK like policy to background applications that are using CMA memory. If the foreground activity requires CMA memory and cannot allocate any, the Nexus LMK will engage to kill a background task and free memory for CMA usage.

Recommendations

Separate the application into UI and services with discrete responsibility

By separating the application into multiple components, Android's LMK can selectively kill portions of the application and leave longer running critical components alive.

Implement long running, non-graphics components as services

Services receive higher priority when the Android LMK engages to kill a process and free up memory. Since they do not use graphics, they are unlikely to have large regions of CMA allocated to them, except possibly in the case of a service that owns a video decoder.

References

1. Android lifecycle
2. Lowram Process States
3. Android Low Ram CMA

Broadcom Proprietary and Confidential. © 2019 Broadcom. All rights reserved.