ION - Large pages for devices

ARM

John Einar Reitan <john.reitan@arm.com>

Android/Mobile Microconference - LPC 2016

Motivation

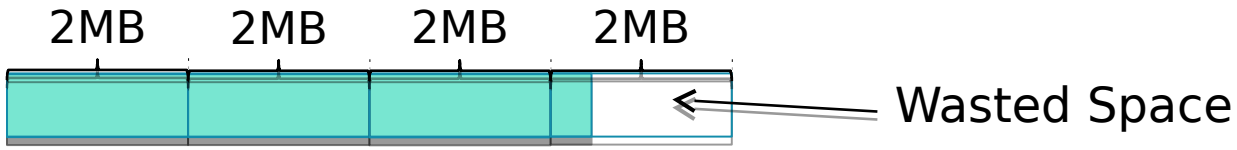
- ARM Display + IOMMU need 2MB pages when rotating
- Native page size 4kB
- 64kB pages investigated on Android, more or less dismissed
- Assumption to be tested: 2MB page size only achievable through use of reserved memory (carveout)
- Undesirable: Sized according to worst-case requirement but unallocated memory wasted

Compound Page ION heap

- New ION heap
- Any page order supported, but 2MB selected based on environment
- Designed to replace carveouts
- Holds a pool of pre-allocated and zeroed 2MB pages
  - Gralloc requests normally fulfilled immediately using 2MB pages in pool
  - Pool replenished asynchronously
  - Hides time spent doing page migration etc in async thread – can even run on LITTLE cores
- Registers as a shrinker
- Optionally collects usage data

Reducing Internal Fragmentation

- Caused by unused memory at the end of the last 2MB page.
  - E.g. 2560x1440 RGBA buffer is 14.065MB. 8x2MB pages, but only 64kB of last 2MB page used:



- Display maps buffers using 2MB page size
  - So can’t pack multiple “tails” – has to be mapped as virtually contiguous
- But a typical UI has a mixture of a few large surfaces and many small(<2MB) ones.
  - Can pack multiple such buffers into after a “tail”

Nexus 10 Background

- WQXGA 2560x1600 Display
  - World leading at release
  - ~Mid-range 2017 phone
- 2GB RAM
  - ~Mid-range 2016 phone
- Real consumer device
  - Lots of IO devices
  - Lots of drivers potentially allocating pinned memory

Nexus 10 Original Issues

- Pinned 4KB Pages was a problem
  - After ~4 days use, unable to find enough contiguous pages for fork()
    - Needed 16kB kmalloc
  - Traced to both ARM GPU and display subsystem pinning 4KB pages
  - Mitigated by adding a carveout for the GPU (384MB)
- IOMMU
  - Discovered display corruption when enabled
  - Traced down to IOMMU TLB misses
  - 2MB pages solved the problem
  - 256MB carveout set aside, exposed as ION chunk allocator

Dogfooding on the Nexus 10

- Nexus 10 with latest standard image
  - Including Google apps: Gmail, Maps, Chrome, YouTube, etc.
- Display chunk allocator replaced with the new heap
  - Used for all ION allocations (GPU and Hwcomposer surfaces, Camera and Video Decoder buffers, etc)
  - …extra 256MB for Linux
- GPU carveout disabled
  - Wanted to try to re-produce fork() issue++
  - …extra 384MB for Linux
- Used in day-to-day activities
  - Youtube, Chrome, Slack, Google+, Gmail, etc...

Scripted testing

- GPU benchmarks
- Open/close tabs in Chrome
- Youtube playback

Observations

- Dogfooding felt responsive, more capable than stock (more tabs etc), but would after ~2 weeks not wake up from sleep...
- Scripted tests could run for ~1-2 weeks, then faulted
- Both issues debugged to be GPU page pinning…

Example stats after 2 week run

- Pool current status
  - 21 free pages in pool
    - Applications had had freed back to the pool after the allocation fault
  - 10 partial pages in use
    - 1MB unused in total across the partials
  - Depleted 8 times
- Performance
  - Max. time spent fulfilling a gralloc allocation: 22ms
  - Max time spent allocating a single 2MB page: 49 ms
- 2MB page acquisition failures
  - Refill failures (non-fatal): 296
  - gralloc visible failures: 21
- Shrinker
  - Called 23 times
  - 184MB reclaimed in total

Usage over 2 weeks

- Allocation Totals
  - Total alloc requests fulfilled: 207,444
  - Current live allocs: 39
  - Current live bytes requested: 173 MB
  - Current live bytes used: 178 MB (5MB, 3% wasted)
- Distribution:
  - Packed: Live: 31, accumulated: 200,145
  - 2-page: Live: 0, accumulated: 7,004
  - 3-page: Live: 0, accumulated: 42
  - 4-page: Live: 1, accumulated: 1
  - 8-page: Live: 6, accumulated: 290
  - 15-page: Live: 1, accumulated: 4

What’s next?

Short term:

- Improve GPU driver, other drivers
  - Page migration support
  - Avoid causing fragmentation in the first place
- Extend ION system heap to support order 9

```
-static const unsigned int orders[] = {8, 4, 0};
+static const unsigned int orders[] = {9, 8, 4, 0};
```

- Find a way to specify a lowest order allowed for an allocation
- Upstream this heap?
- Or no new features for ION in mainline?

What’s next?

Longer term:

- Order constraint support in the new Unix Device Memory Allocator
  - Started/discussed at XDC 2016
  - http://theinnocuous.com/udma.pdf
  - https://github.com/cubanismo/allocator

Thank you

Any questions?

ARM