DMABUF Developments

Sumit Semwal <sumit.semwal@linaro.org>
John Stultz <john.stultz@linaro.org>

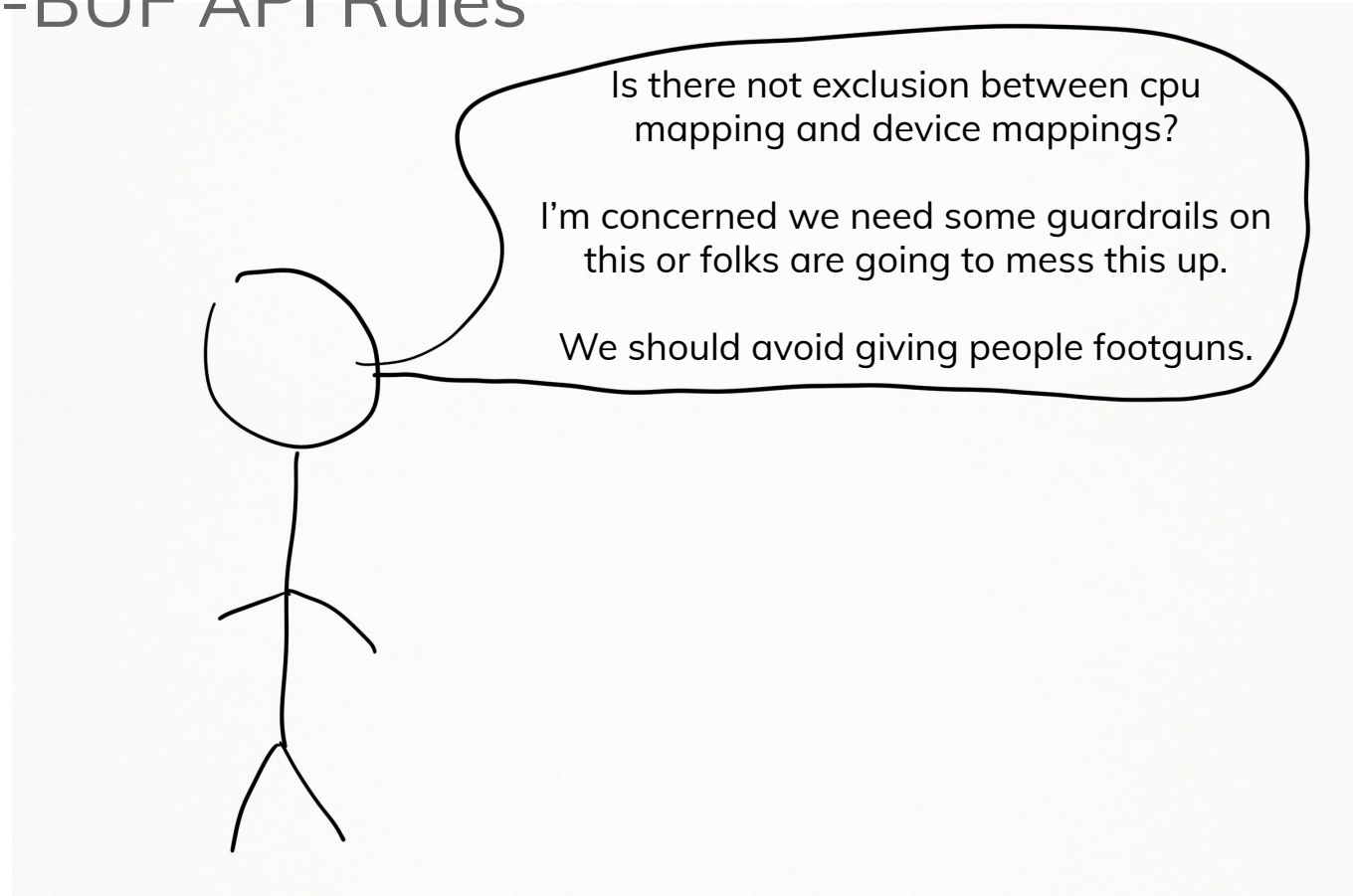
DMA-BUF Heaps (ION Destaging)

- ION Issues:
  - Trying to do too much in one interface (constraint solving, cache handling, etc)
  - Too flexible/poorly specified interface
  - Tried to handle all the cache management so vendors couldn't get it wrong
  - Vendor extensions often modified that shared code, making them incompatible and impossible to upstream
  - Cleanups/fixes for upstreaming have broken ABI and desired(even if “incorrect”) cache behavior

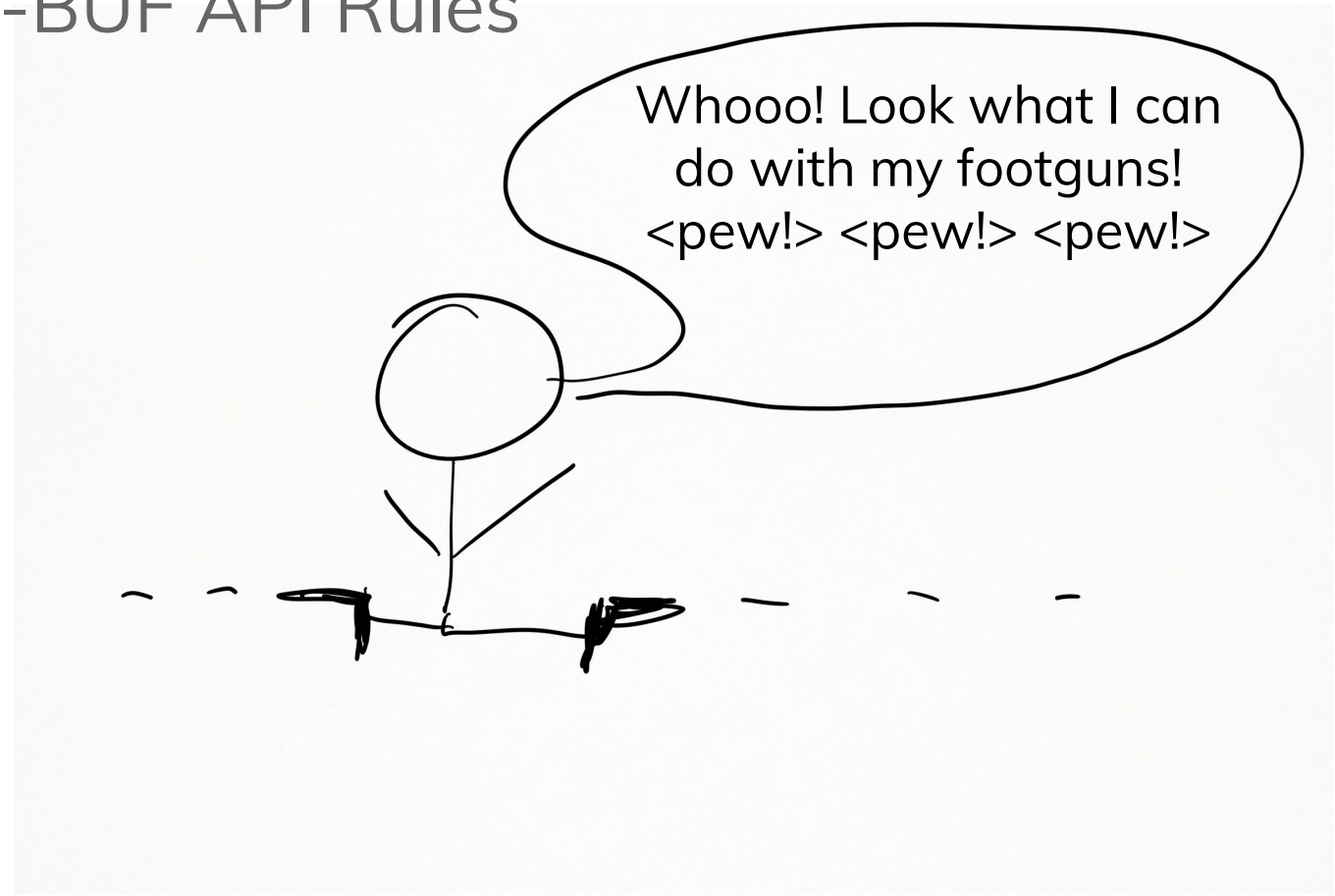
- DMA BUF Heaps
  - Focused on one feature from ION: Userland dma-buf allocation interface for various “types” of memory
  - Each heap driver is its own dmabuf exporter, so no need to modify core code for custom heap
  - Each heap driver gets its own chardev, no multiplexing heaps through one interface
  - Helper functions to avoid duplication where possible
  - Not planning on solving every crazy use case (ex: secure heaps that need per-allocation ids). You can still write your own dmabuf exporter driver!

- Vendors: Please move your ION code over to DMA-BUF Heaps once its upstream!

DMA-BUF API Rules



DMA-BUF API Rules



DMA-BUF API Rules

Topic: Need for clearer DMA-BUF API rules

- DMA-BUF Design vs actual DMA-BUF usage
  - Generic attach(), delayed-alloc, map() pattern never really realized
  - cpu_begin/end_access() and device_map/unmap() as exclusive usage barriers
    - Now cpu_begin/end_access and fence_wait/signal()?
- Implicit signaling vs explicit
  - How do we cleanly/clearly support both
- Use cases exist for multiple device maps used in parallel (manual cache cleaning & signaling w/ fences)
- Do we want to enforce exclusion between mmap/vmap/kmap and map_for_device?
  - Concern: GL has use cases for one buffer, accessed at the same time by device and cpu “carefully”
  - https://www.khronos.org/opengl/wiki/Buffer_Object_Streaming#Buffer_update
- How do we sanely address Christoph’s feedback? Without limiting some of these usage models?
- Some rules would help, as otherwise some optimizations are limited due to lack of ability to assert correctness (see next slide)

DMA-BUF Cache Management Optimizations

Topic: Ideas for reducing overhead of dma_map/unmap_sg() on every device_map/unmap()

- Cache flushing overhead of dma_map_sg() is significant (map/unmap * every device * every frame)
- Issue cropped up after 4.12 ION DMA API fixes made for correctness
  - Vendors using their own hacks to avoid (revert ION to 4.9, “uncached” buffers)
- Many in-kernel dma-buf exporters cheat by dma_map_sg on attach and when direction changes
  - Made generic w/ cache_sgt_mapping flag by Christian König
- Often buffer is never touched by the CPU
  - Can we safely dma_map_sg on attach and not touch it until begin_cpu_access()?
  - Allow to do lazy flushing when ownership/usage changes
  - Issue: To do this properly, seems to require exclusive device/cpu mapping?

Topic: Efficient partial cache invalidation on dma-bufs (Alistair)

- Caching handling routines for camera YUV, need to be able to flush 2d patterns in the data.
- Another example was when there is meta data in random places that needs to be flushed.
- Proposal: Range flushes
  - dma_clean_range()? dma_flush_range()?
  - Need clear articulation of the need to the community *and* an upstream user of the code.

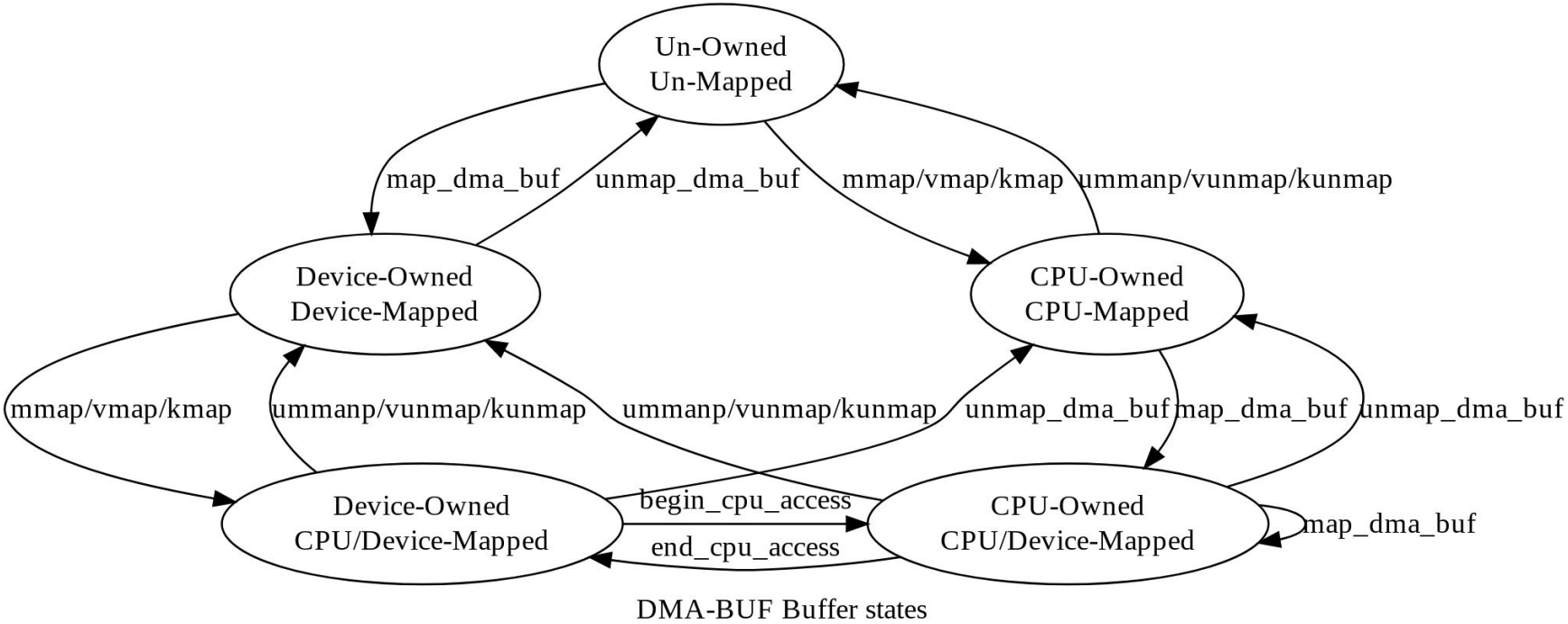
Kernel Graphics Buffers

Topic: New kernel graphic buffer abstraction built on top of dma-buf (Marissa/Alistair)

- Single handle to image buffer + meta-data
- Provides standardized pixel format info and meta-data (similar to gralloc native handle structure)
- Would allow for smarter partial (2d region) cache invalidations
- Question: Are GEM buffers the ‘right’ answer here? How else can we do this?

Backup Slides

DMA-BUF Ownership Statemachine (by Andrew Davis)



DMA-BUF Buffer states

https://dreampuf.github.io/GraphvizOnline/

Linaro