Google

videobuf2 in 2024

Current state, limitations and path forward

Media Summit 2024

Tomasz Figa <tfiga@chromium.org>

Sep 2024

01
Current state
aka recap of recent changes

Removed the fixed 32 buffer limit

Kudos to Benjamin (Collabora) and everyone involved.

- No more statically sized array of vb2_buffer
- No direct access to buffer array or size - only via helpers
- vb2_buffer * used instead of index across vb2 helpers
- Default is still 32 buffers - easy override via vb2_queue struct field

ML archive:
https://lore.kernel.org/linux-media/20231109163008.179152-1-benjamin.gaignard@collabora.com/

Added buffer removal

Kudos to Benjamin (Collabora) and everyone involved.

- New VIDIOC_REMOVE_BUFS ioctl deletes a given range of buffers (indexes) from a buffer queue
- Drivers can opt-in by setting the .vidioc_remove_bufs op to vb2_ioctl_remove_bufs()
  - Code must not hold vb2_buffer pointers outside of driver's buffer ownership
- The buffers must be in the DEQUEUED state

ML archive:
https://lore.kernel.org/linux-media/20240314153226.197445-1-benjamin.gaignard@collabora.com/

Optimized DMA-buf plane mapping

Kudos to Yunke (Google, ChromeOS) and everyone involved.

- Re-use mem_priv between planes if they import the same DMA-buf
- Eliminates unnecessary DMA mapping operations and waste of IOVA space
- Note: Only works within one buffer
    - If another vb2_buffer has the same DMA-buf imported to it, it will have its own duplicated DMA mapping

ML archive:
https://lore.kernel.org/linux-media/20240814020643.2229637-1-yunkec@chromium.org/

02

Limitations

aka room for improvement

Drivers that need CPU access to buffers

- Sharing buffers between CPU and DMA requires proper memory coherence via for example
  - CPU cache maintenance (invalidate, clean)
  - coherent memory (uncached/write-combine mapping)
- What does the framework provide today?

Buf_type / allocator	**vb2-dma-sg**	**vb2-dma-contig**	**vb2-vmalloc**
MMAP	cache maint. in buf_prepare/_finish	coherent memory OR cache maint. in buf_prepare/_finish	only CPU access
USERPTR	cache maint. in buf_prepare/_finish	cache maint. in buf_prepare/_finish	only CPU access
DMABUF	**nothing**	**nothing**	**nothing**

Drivers that need CPU access to buffers, cont'd

- DMA-buf provides dma_buf_begin_cpu_access() and dma_buf_end_cpu_access() to transfer the ownership to/away from the CPU
- ~30 drivers [1] don't do any CPU cache synchronization for imported DMA-buf despite accessing them from the CPU

Idea:

- Add vb2_{begin/end}_cpu_access() helpers, carefully audit each driver and add calls to those around the CPU accesses
  - Note: Would only affect DMABUF mode, so low potential for regression (and probably already broken)
  - Tomasz will work on an RFC

[1] https://lore.kernel.org/linux-media/CAAFQd5DfbqOkZzPfCNRMGeMgv2NfM6WENWXeLUNsuMgkzeBQKw@mail.gmail.com/

Page allocation in vb2-dma-sg

- vb2-dma-sg has its own allocation of pages for MMAP buffers
  - vb2_dma_sg_alloc_compacted()
- Limitations:
  - alloc_pages() does not account for DMA constraints (e.g. DMA mask)
    - dma_alloc_pages() does
  - not using __GFP_NORETRY, which will trigger costly reclaim if there are no big order free pages at hand
- DMA mapping already has similar code backing dma_alloc_coherent() and dma_alloc_noncontiguous()

Idea:

- Extend dma_alloc_noncontiguous() to accept maximum number of sg_table segments (suggested by Christoph Hellwig [1])
  - Tomasz will collaborate with DMA maintainers to see if this could happen
- Side effect: The same allocation function could be used for both sg and contig
  - Could we merge them into a single vb2-dma?

[1] https://lore.kernel.org/linux-media/20230926094616.GA14877@lst.de/

DMABUF buffer mapping lifetime

- VB2 attaches and maps a DMA-buf when the buffer at the given index is being prepared
  - The mapping for the buffer index is kept until a different DMA-buf is queued or the index is removed (e.g. via REQBUFS or REMOVE_BUFS)
- Problematic scenarios:
  - Video codecs can use dequeued CAPTURE buffers as reference frames, but they could be unmapped if the application queues a different DMA-buf at the same buffer index
  - Wasteful repetitive maps and unmaps if the application doesn't strictly match buffer indexes and DMA-bufs

Idea:

- Decouple physical buffer storage from the position in the queue
  - Sounds much easier than done. Tomasz is still thinking about the exact proposal

Duplicate DMA mappings

- Queuing the same DMA-buf at different indexes of the same vb2 queue or to different vb2 queues of the same device leads to duplicate DMA mappings being created
- Example: A complex ISP device with multiple processing blocks where the same output is used as input of multiple other processing blocks
- Example 2: Application keeps a pool of buffers around to avoid allocation cost, but they don't end up always used for the same video nodes
- Example 3: The same DMA-buf is queued with different offsets at different V4L2 indexes
    - Very convenient to use a ring buffer-like pattern to deal with compressed data, as the frame size varies heavily

Ideas:

- Could possibly be handled at the DMA API level by looking up and refcounting existing mappings with matching attributes

03
Other improvements

Can locking be simplified?

- Videobuf2 locking patterns tend to be perceived as overly complex by developers
  - mmap_lock (mutex)
  - done_lock (spinlock)
  - vb2_queue->lock (mutex)
  - media_request->lock (spinlock)
  - media_request_(un)lock_for_update (helpers)
  - wait_prepare/_finish (helpers)
  - V4L2 video_dev lock (mutex)
- History of timing bugs detected via fuzzing
- How do we make sure that the locking is understandable and remains correct going forward?
- Request for ideas!

Idea: Remove .wait_prepare/_finish

- Could we enforce the locking in the framework and remove the custom unlocking callbacks?
- First step: Drop the custom callbacks for most drivers (patches by Hans [1])
- What are the use cases for custom locking?

[1] https://lore.kernel.org/linux-media/cover.1725285495.git.hverkuil-cisco@xs4all.nl/

Idea: More documentation

- Create a dedicated htmldoc page for vb2 locking
- Are there any useful annotation macros that could be added to the code to document what’s guarded by which locks?

Misc clean-up ideas

- Simplify USERPTR code
  - After removing support for PFNMAP the frame_vector abstraction is not useful anymore.
- Reducing boilerplate
  - Returning all buffers when stopping streaming
    - We ultimately only need the driver to just make sure the buffers are not in-use anymore.
    - Would need to keep around a list of buffers owned by the driver.
  - Queue setup
    - All drivers implement pretty much the same thing, but with their own little bugs all around.
- Anything else?

Thank you!