Exporting virtual memory as dmabuf

Nikhil Devshatwar
Texas Instruments, India

Texas Instruments

About author

- Embedded Linux developer @Texas Instruments
  - Video subsystem
  - Camera drivers
  - Base port support
- Contributions
  - V4L2 drivers
  - Device tree compilers

Texas Instruments

Outline

- dmabuf support in kernel drivers
- Problems with Legacy drivers
- Memory sharing constraints
- Virtual memory export – concept
- New use cases
- Memory sharing over client/server
- Security concerns

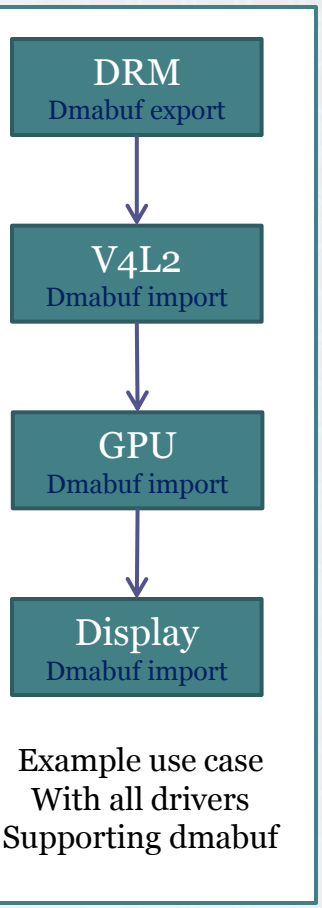
Texas Instruments

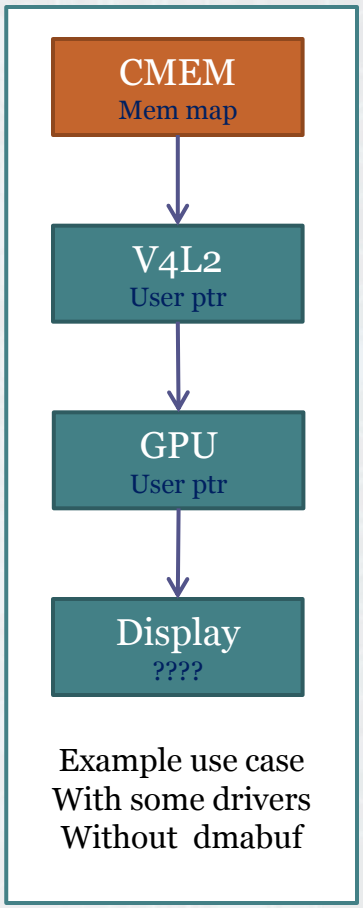
Dmabuf support in Linux kernel

- Generic mechanism for buffer sharing
- Most embedded drivers support dmabuf import
- Some legacy drivers lack support for dmabuf
  - Buffers have to be allocated using the same driver
  - Map the buffers for use in application
  - Cannot share it with other dmabuf importers
  - E.g. CMEM driver for manging CMA area
- Sharing of buffers is a concern with legacy drivers

Texas Instruments

Problems with legacy drivers





- Some drivers support only mmap
- Some support only dmabuf

- Creating use cases with heterogeneous mix of drivers
- Incompatibility at use case level

- Buffer copy to realize such use cases

Texas Instruments

Sharing memory – typical flow

- For sharing memory using dmabuf
  - First allocate memory from the exporter
  - Export it as dmabuf
  - Import it via the user/importer
  - Generate content in the imported memory
- For sharing memory using shared memory
  - First allocate shared memory
  - Map the shared memory in different context
  - Generate content into it

Texas Instruments

Dependencies on content generation

- Many content generator libraries allocate their own memory
  - Gstreamer plugins like videotestsrc, appsrc
  - Software implemented algorithms
  - 3rd party libraries
- What if we could export the memory after it was allocated?
  - Share it as a regular dmabuf with other drivers

Texas Instruments

Virtual Memory Exporter

- ABI - Simple character driver with few ioctls:
  - Ioctl to export memory regions as dmabuf
    - DBUFIOC_VMEM_EXPORT
  - Ioctl for cache sync operations
    - DBUFIOC_VMEM_SYNC
- Impl^{n} – Software page walk
  - Find our vaddr => pfn mapping
  - Implement map/unmap/kmap to dma_map_sg
  - Lock pages to avoid swapping
  - Export any vaddr, even user space memory
  - Export memory mapped by other drivers

Texas Instruments

Features & Limitations

- Multi context support
  - Each context is managed separately
  - Removing all exports upon device closure
  - Export overlapping regions as different dmabufs
    - Works with both single/multi planar buffers
- Only page aligned addresses
  - Most dmabuf importers don’t respect SGT offset
  - Avoid exporting non page aligned addresses
- Support contiguous & scatterlist buffers both
  - No limitation on the buffers being contiguous
  - Imported may reject the import if incompatible

Texas Instruments

New use cases

- Some of the DRM displays can work with SGT
  - Use malloc[ed] buffer to display
- Integrate drivers w/o dmabuf support
  - Map the driver memory, export is as dmabuf
- Integrate software algorithms
  - Let the library allocate memory
  - Export it as dmabuf, share with GPU, display
- Use CMA drivers with zero copy
  - Allocate memory using CMA drivers
  - Map them to get vaddr, export as dmabuf

Texas Instruments

Memory sharing – client/server

- Compositor systems (e.g. weston, X11) are client/server based
  - Clients talk to servers via sockets only
  - Buffers are shared using dmabuf or shm
- Components allocating own memory for buffers
  - E.g. Gstreamer plugins, custom shaders, textures
  - Sharing these buffers involves copy to shm
- Export as dmabuf and share across process
  - Using socket's fd passing mechanism
  - Vmem from one process can be mapped to other
- Shared memory redefined!
  - No need to pre allocate the shmem regions
  - Export whenever something needs to be shared

Texas Instruments

Security concerns

- Security checks in place
  - Page walk takes care of access overflow & segfault
  - Restrict export to only certain types of VMAs (Only data segment)
- Are we creating any security holes?
- Should we restrict sharing of certain mem areas?
- What happens if the owner of virt mem unmaps/frees it?
  - Is this application's fault or kernel bug?

Thank you

Nikhil Devshatwar
nikhil.nd@ti.com
nikhildevshatwar@gmail.com