Exporting virtual memory as dmabuf

Nikhil Devshatwar
Texas Instruments, India

About author

- Embedded Linux developer
- Texas Instruments (Bangalore, India)
- Key work areas
  - Video subsystem drivers
  - Camera drivers
  - Base port support
  - Linux & RTOS integration
- Contributions
  - V4L2 drivers
  - Device tree compilers

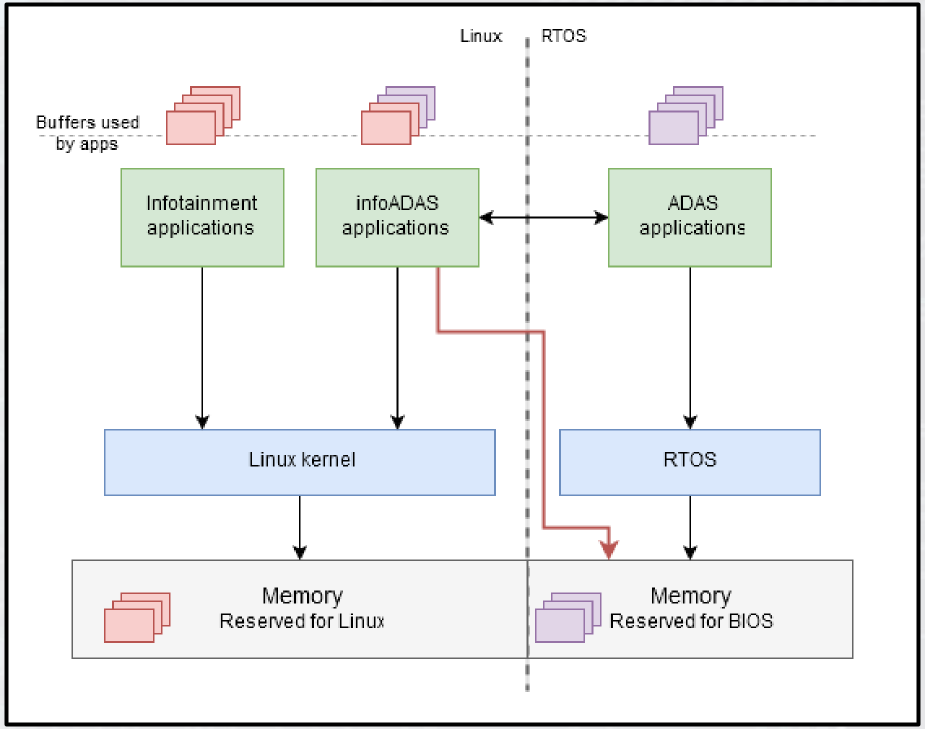
Outline

- Problems with RTOS integration with Linux
- Solution with dmabuf
- Generic solution: vmemexp
- Implementation details
- Memory sharing constraints
- Memory sharing over client/server
- vmemexp Use cases
- Security concerns

Note: Only embedded hardware and use cases considered

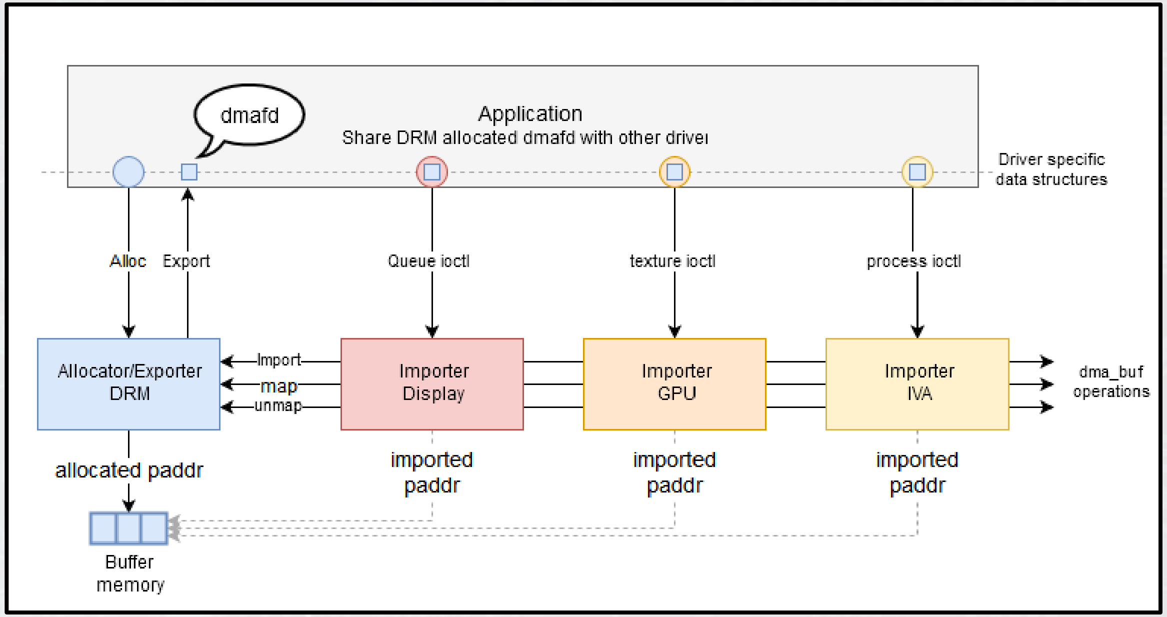
Integrating RTOS with Linux

- Dedicated memory carve outs
- Linux app accesses RTOS buffers
- For using HLOS features
  - Access RTOS memory
  - Share with Linux drivers
- Typical usecases
  - GPU & Display
  - Encode & File save

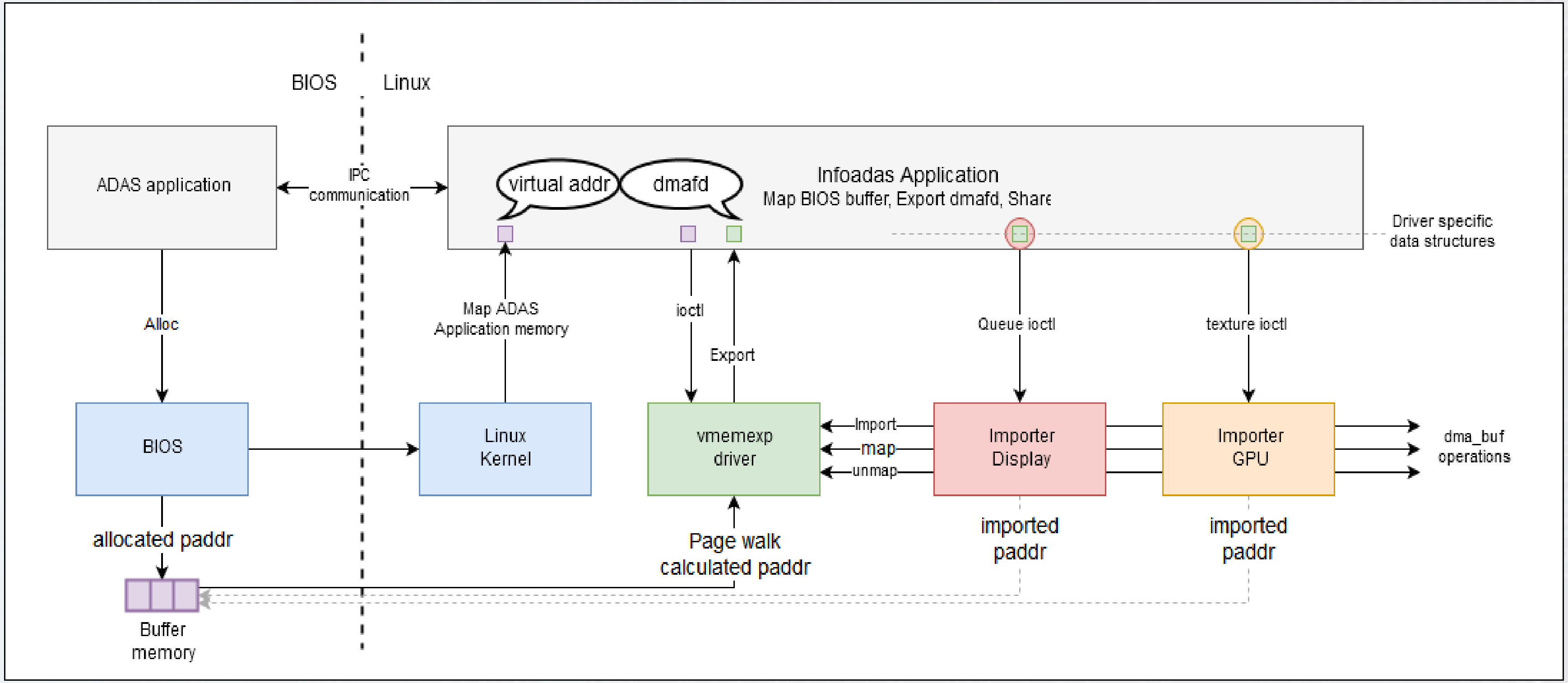


A short note on dma_buf

- Share buffers across drivers
  - Using anon file descriptor
- Allocator knows paddr
  - Implements dma_buf_ops
  - Export buffers as dmafd
- Application passes dmafd
  - Drivers import dma_buf
  - Access buffer using ops



Soln: Map and export as dma_buf



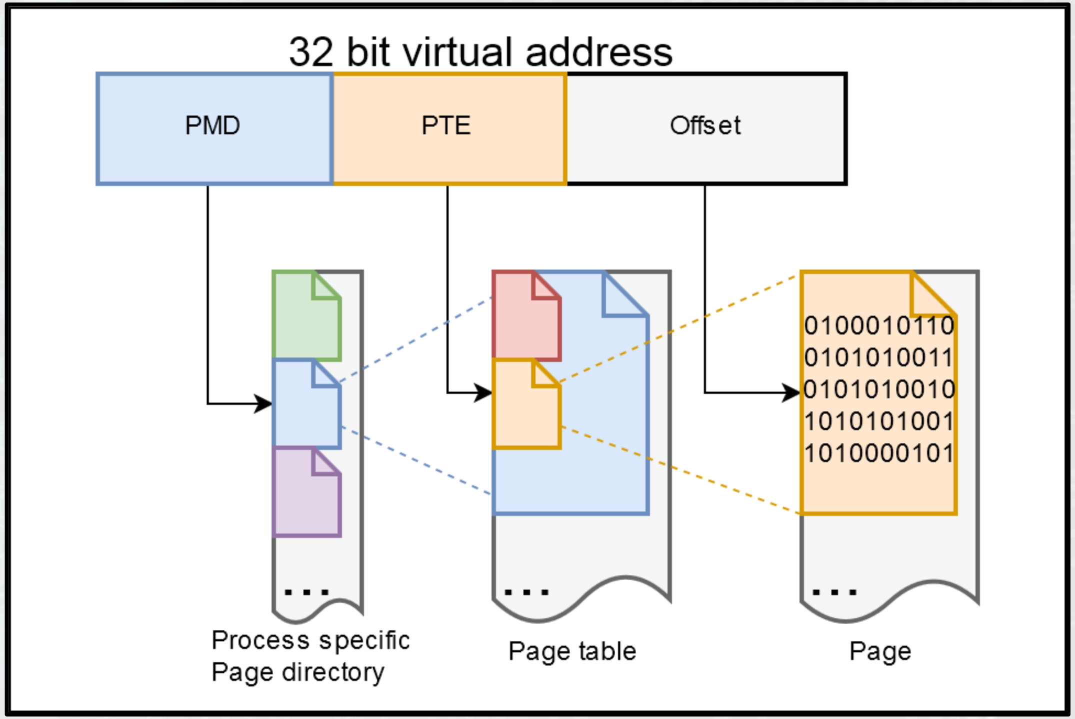
Generic soln: Virtual memory export

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

vmemexp: Page table walk

- Get paddr for vaddr
- Decode PGD, PMD, PTE
- Example on 32bit ARM processor ===>
- Security checks
  - Page attributes
  - Seg fault checks

PGD = Page global directory
PMD = Page middle directory
PTE = Page table entry

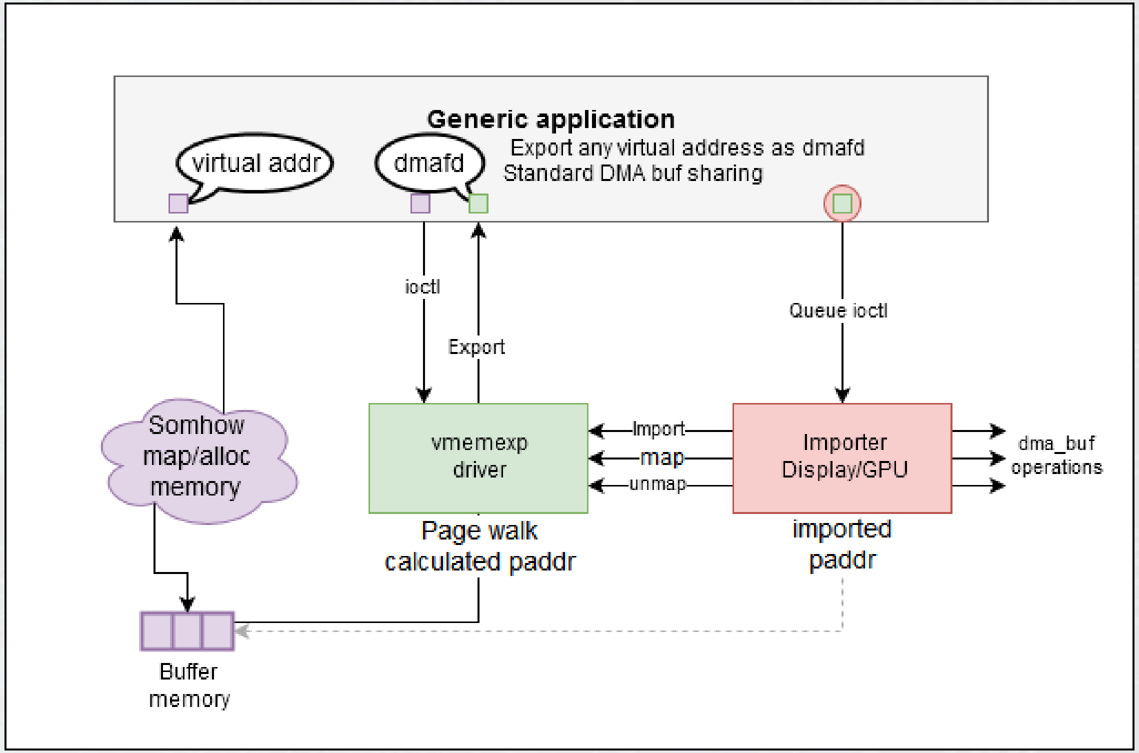


Features & Limitations

- Supports both contiguous and Scatter gather buffers
- Page aligned addresses recommended
  - Some importers do not respect SGT offset while importing
- Assumes GPU doesn’t have separate memory
  - Typical embedded GPUs work with system memory
  - If not, dmabuf import won’t be supported by DRM
- Assumes GPU has MMU
  - For importing non-contiguous buffers (like Malloc)
- Depending on Hardware support for scatter-gather usecases vary
  - If the hardware needs contiguous buffer, no point in exporting a SGT

What can be exported?

- va = mmap(mem)
- va = mmap(drv_hdl)
- va = mmap(file)
- va = malloc(size)
- dmafd = ioctl(vmem, va)
- drmSubmit(dmabuf)
- GL_EXT_DMABUF, dmafd



Memory sharing w/ & w/o vmemexp

- Allocate from dmabuf exporter
  - Export using same driver

- Gstreamer usecase
- Alloc before creating content
  - First alloc from DRM, share with src
  - Generate content in imported memory
  - Content generator should be aware of external allocation

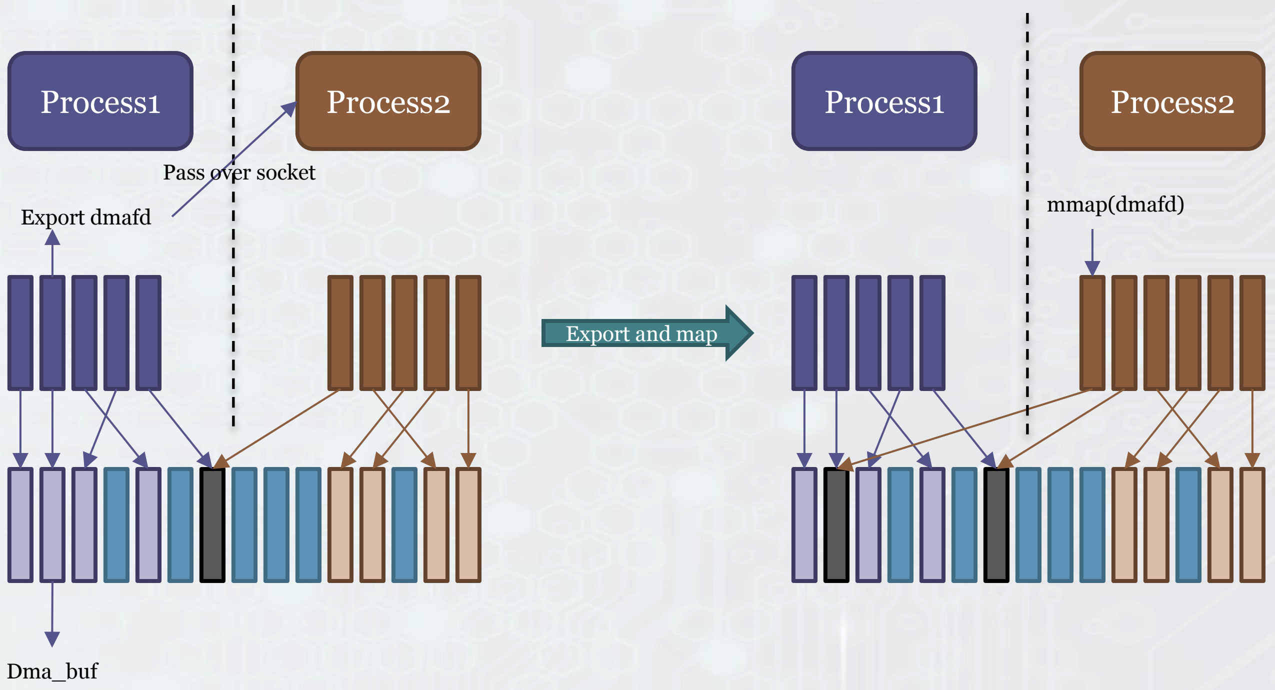
- Allocate from any driver
  - Map and export via vmemexp

- Gstreamer usecase
- Export content from src
  - Allocate at the source
  - Generate content in allocated memory
  - Export the shared memory and share with DRM
  - No dependency on external allocation

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

Memory sharing example



▨	Process1 memory
▨	Process2 memory
▨	Free memory
■	Shared memory

vmemexp use cases

- Integration of RTOS applications with Linux
- Enable dmabuf export for drivers which don’t support
  - E.g. CMA drivers like CMEM drivers (zero-copy)
- Share memory mapped from other drivers
- Export user space memory (malloc)
  - If Display supports SGT, display a malloced buffer
- Share a process’ memory to other process
  - Using fd passing for dmabuf
- Easy integration of software components
  - Overcome content generation dependencies

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)
- Added page reference to avoid swapping out
- Trigger page faults to make sure all pages are in
- Are we creating any security holes?
- Should we restrict sharing of certain mem areas?
- Handle races between unmap and close(fd)

References

- Informational ADAS: http://www.ti.com/general/docs/video/watch.tsp?entryid=4274413412001
- Linux kernel DMA buf documentation: https://www.kernel.org/doc/Documentation/dma-buf-sharing.txt
- DMA buf presentation: http://elinux.org/images/a/a8/DMA_Buffer_Sharing-_An_Introduction.pdf
- Page table information: https://en.wikipedia.org/wiki/Page_table
- Linux shared memory manual: http://man7.org/linux/man-pages/man7/shm_overview.7.html
- Videobuf2_dma_sg: http://lxr.free-electrons.com/source/drivers/media/v4l2-core/videobuf2-dma-sg.c

Thank you

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