Passthrough/Headless GPU Gets A Head

Tina Zhang tina.zhang@intel.com

Vivek Kasireddy vivek.kasireddy@intel.com

Sep. 15th 2021

Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. No product or component can be absolutely secure.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

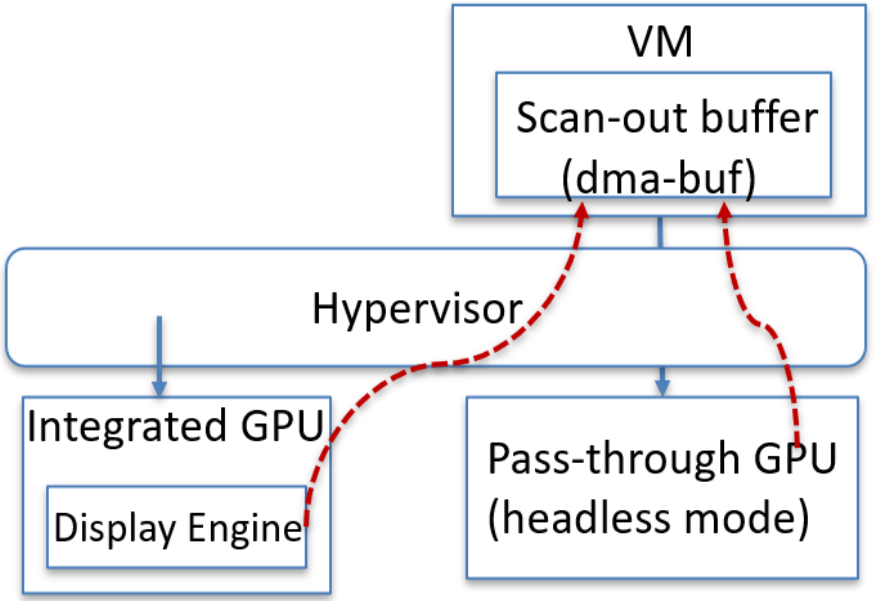
*Other names and brands may be claimed as the property of others.

Agenda

- Motivation & Background
- Architecture
- Implementation
- Summary

Motivation & Background

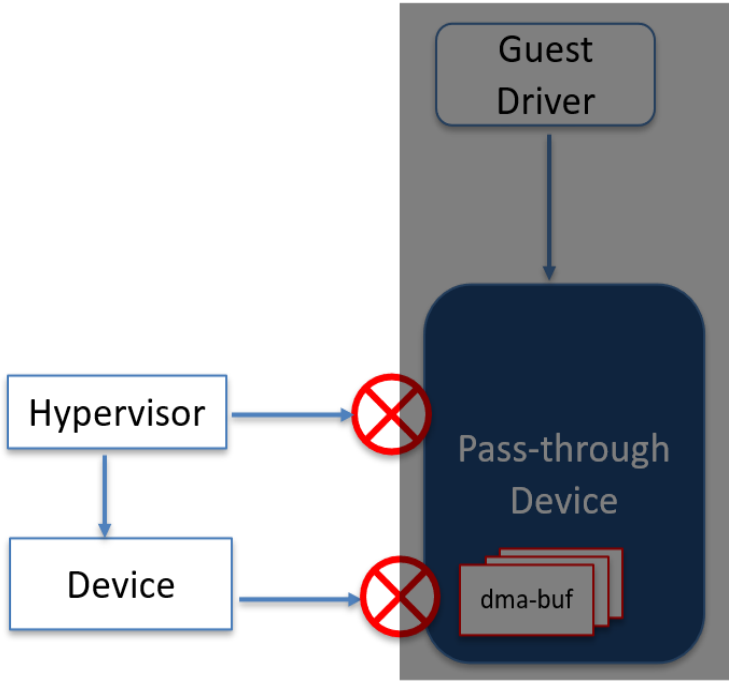
- Multi-GPU platform
  - More than one GPU is provided
  - Power & performance benefits
- A multi-GPU use case in client virtualization field
  - Pass-through GPU:
    - Best GPU performance within VM
  - Integrated GPU for host display and render
    - Power-saving
  - Need scan-out buffer sharing mechanism between iGPU and pass-through GPU
    - **cross-domain dma-buf sharing**



Share guest scan-out buffer between pass-through GPU & integrated GPU

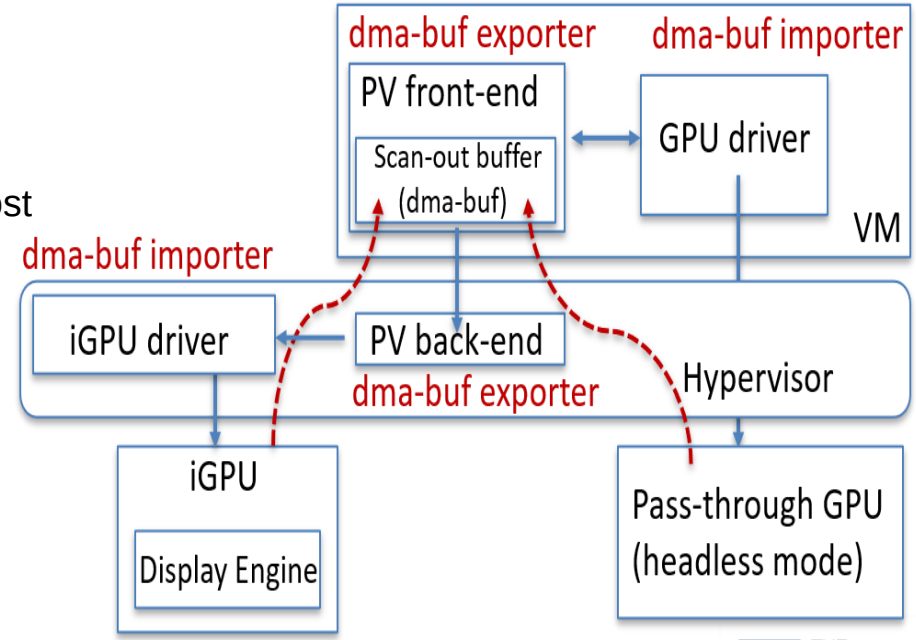
Cross-Domain DMA-BUF Sharing

- Sharing dma-buf owned by a pass-through device might not be feasible
  - Hypervisor has no visibility of dma-buf resource of a pass-through device
  - The backing storage of a pass-through device's dma-buf may be its private local memory which may not be accessible to other devices
- Proposal: para-virtualization based dma-buf exporter

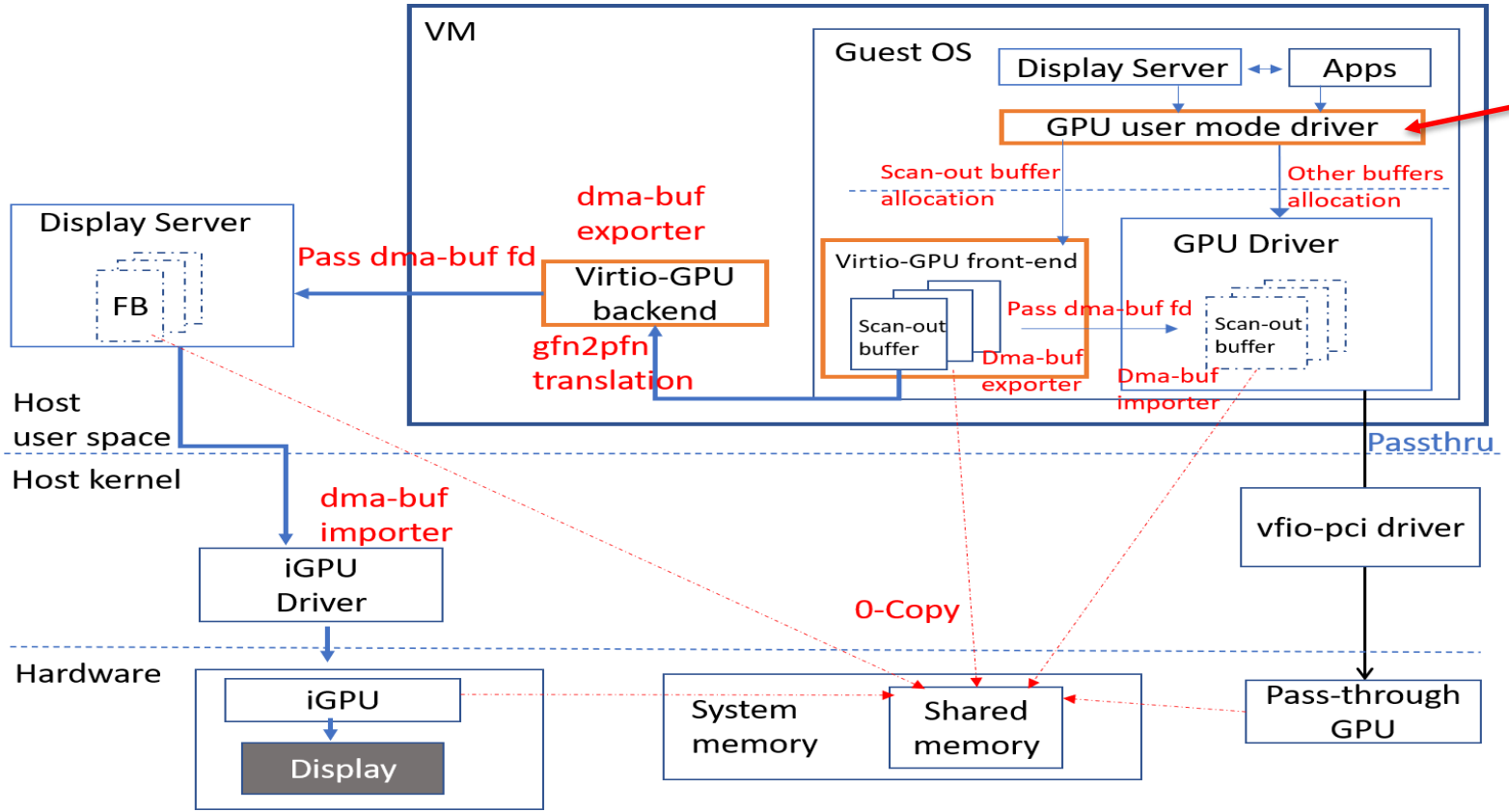


PV Based DMA-BUF Exporter

- VIRTIO-based cross-domain dma-buf sharing mechanism
  - Front-end: page-backed dma-buf owner and exposer in guest
  - Back-end: dma-buf owner and exposer on host
  - Equipped with a buffer producer-consumer synchronization mechanism
- Vdma-buf was proposed
  - RFC patch-set: https://lwn.net/Articles/846810/
- Choose virtio-gpu at last
  - Has cross-domain dma-buf sharing support
  - Supported by Linux user space graphic stack



Architecture Picture



From guest p.o.v:

- Multi-GPU case, if virtio-gpu 2d/3d is enabled;
- Virtio-gpu display + pass-through GPU render-only, if virtio-gpu 2d is enabled.

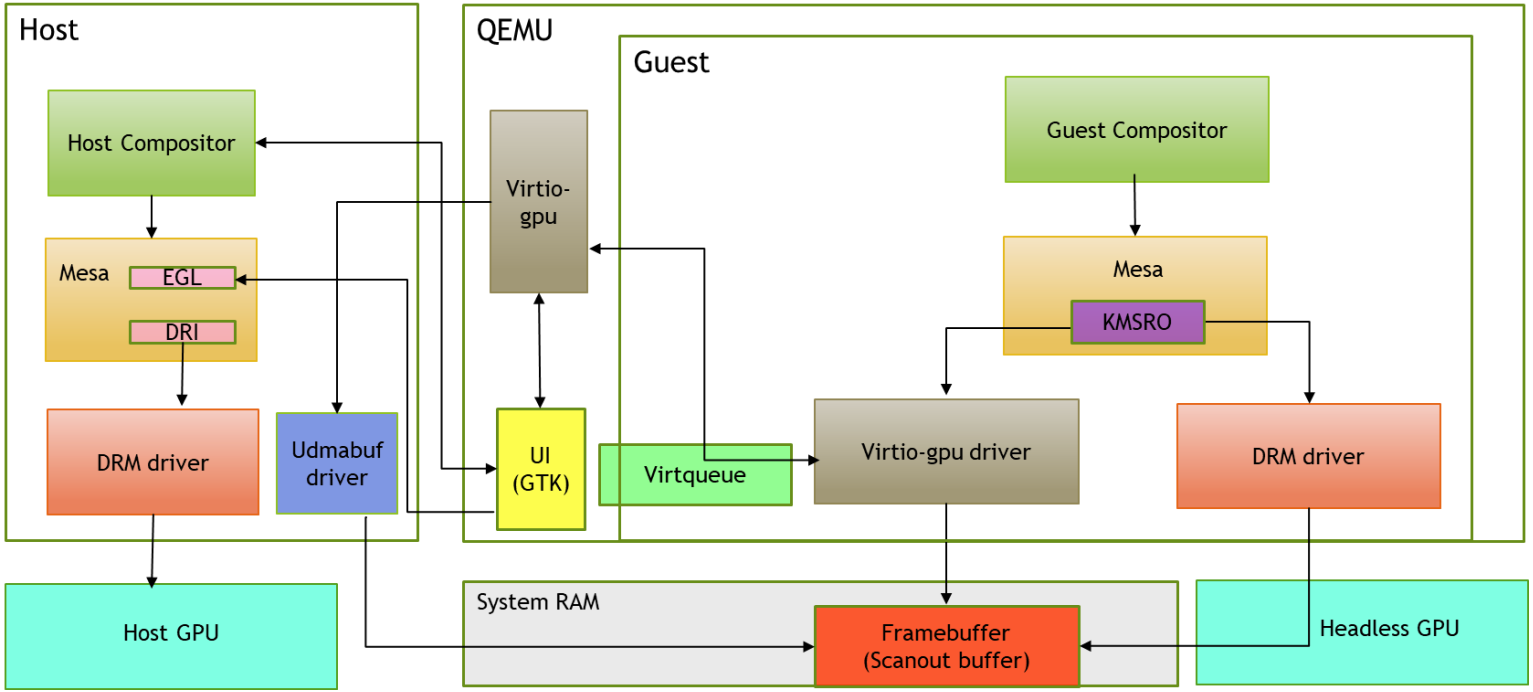
Virtio-gpu: Performance challenges

- More headless GPUs coming up: Intel XeHP SDV, SRIOV VFs, etc.
- Noticed that there was one CPU copy and multiple GPU copies done for transferring the Guest framebuffer data to the Host.
- Solution to eliminate the CPU copy was already available: Blob resources.
- Virtio-gpu and Qemu UI maintainer (Gerd Hoffmann) created initial patches to implement this feature more than a year ago.
- Refactored and augmented these initial patches and added few more and finally got them merged.

Virtio-gpu: Blob resources

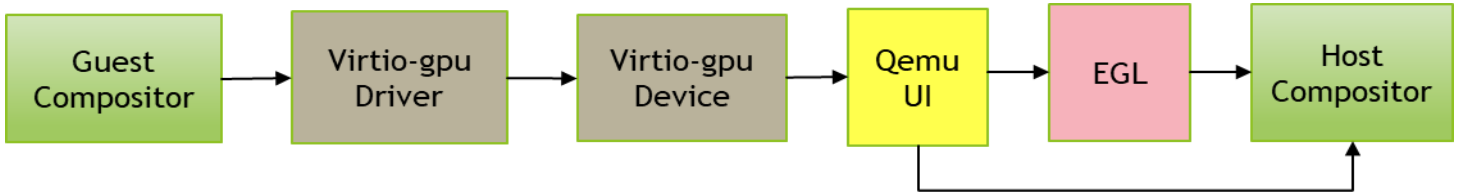
- It is based on Linux dma-buf buffer sharing framework.
- Before this feature, the resource data was (mem) copied from the IOV to a shadow buffer (pixman) and then a texture was created.
- We now associate a dma-buf fd with the resource (FB) by passing the IOV entries to the Udmabuf driver and getting an fd in return.
- A texture can be created directly using the fd thereby eliminating the need for a CPU copy.
- The Udmabuf driver was also augmented to work if Qemu's memory backend was backed up by memfd + Hugepages.

Virtio-gpu: Blob resources



Virtio-gpu: Blob resource synchronization

- Synchronization (i.e., prevent Host and Guest from accessing a buffer at the same time) was never a problem before this feature.
- And guests would render frames at a rate faster than what the Host can consume -- wasting GPU cycles.
- Fix both these issues by introducing sync objects.
- A file descriptor (fd) is extracted from a sync object and can be added to Qemu's main event loop.
- The Guest would be blocked -- from rendering new frames -- until the relevant fd is signaled.



Virtio-gpu: Blob resource synchronization

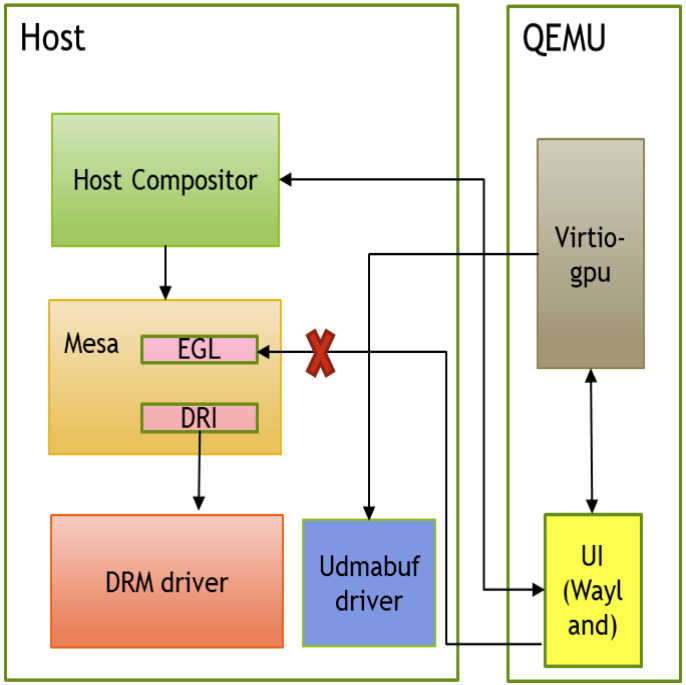
- Sync objects and fds are created using EGL APIs:

  EGLSyncKHR sync = eglCreateSyncKHR();

  int fd = eglDupNativeFenceFDANDROID(…, sync);
- Qemu UI backends such as GTK (default) and SDL that use EGL for presenting Guest frames can make use of these sync objects.
- Once an fd associated with a sync object is signaled, it means the Host is done using the buffer and it can be reused by the Guest.
- This also ensures that the Guest rendering rate is not higher than the monitor refresh rate.
- Most of the patches associated with synchronization are already reviewed and ready to be merged.

Qemu UI: New Wayland UI backend

- Qemu currently supports multiple UI backends: GTK, SDL, Cocoa, Spice, etc.
- All these backends make a copy of the Guest framebuffer -- using either the CPU or the GPU -- before presenting it.
- This can lead to performance issues if multiple Qemu instances are running and presenting Guest frames simultaneously.
- Using Blob resources would eliminate the need for a CPU copy but a GPU copy (aka Blit) cannot be eliminated if EGL is used.
- Only way to eliminate the Blit is to submit the Guest framebuffer directly to the Host display server/compositor.
- A Wayland UI backend makes this possible.



Qemu UI: New Wayland UI backend

- With this backend, it is possible to have the Guest framebuffer be placed directly on a hardware plane (zero copy).
- If a hardware plane is not available, then the Host compositor would inevitably Blit the Guest framebuffer onto its scanout buffer.
- In addition to limiting the (GPU) copies to a maximum of one, this backend is very lightweight and has a smaller footprint.
- Drawbacks include no window decorations (menus, etc.) and its efficacy is limited to iGPUs and Guests that do double-buffer rendering.
- Patches are posted but a non-trivial overhaul (dbus) of Qemu UI modules is underway after which this can be reviewed.

Summary

- Dma-buf exported by passthrough device may not be accessible to other devices controlled in other domains.
- Para-virtualization based solution provides an efficient and generic way for sharing cross-domain dma-buf.
- Using Blob resources would eliminate the need for CPU copies.
- Wayland UI backend would limit the GPU copies to a max of one.

Acknowledgements

- Gerd Hoffmann (Virtio-gpu and Qemu UI maintainer)
- Daniel Vetter (DRM subsystem maintainer)

KVM FORUM