Optimus and cross-device synchronization support

What's done and what needs to be done?

Maarten Lankhorst

Canonical Ltd

September 21, 2012

Summary

1. Architecture
   - The parts involved
   - Synchronization
2. Demo
3. Questions

The primitives

- DMA-BUF
- X.org server
- individual DDX drivers
- xrandr

DMA-BUF

- All kernel video drivers have to be modified to support DMA-BUF
- DMA-BUF objects can be imported and exported as fd's.
- No synchronization is done yet, expect tearing or worse.

X.org server, DDX and xrandr

- Platform bus support is added for supporting dynamic gpu hotplugging
- xrandr 1.4 is used to configure gpu slaves and connect them to masters
- Each ddx needs to add support for platform bus, and also:
  - code for being a output slave (xf86-video-modesetting, intel)
    - USB Displaylink devices that have no hardware acceleration on their own, or intel when a display is connected to nouveau.
  - code for being a offload slave (xf86-video-ati/nouveau)
    - Optimus devices where the slave has more powerful hardware than the master.
  - code for being a offload/output master (xf86-video-ati/intel/nouveau)
    - Required to support offload/output slaves. Offload master is untested for ati and nouveau.
- code for switching between muxed GPUs (WIP)
  - Robustness extension

The problem

- Multiple devices involved
- Arbitrary number of buffers shared in an arbitrary order between those devices
- Preferably no deadlock on either cpu, or because gpu devices waiting on each other for buffer use completion

Example deadlock

- 2 devices, devA and devB.
- devB imports bufA, devA imports bufB.
- Both want to use bufA and bufB, but want to reserve them in opposite order.
- Deadlock! Both hold a buffer and wait for the other.
- This can happen on the cpu if you're lucky, or on both gpu's if unlucky.

TTM Style reservations

- Literally pick up the code from TTM that manages reservations.
- Use reservation_ticket for reservation multiple objects.
- Use fence for cross-device synchronization primitive.

Fence API (WIP!)

- Work in progress, not even the name is finalized.
- Dumbest possible primitive for synchronization
- Signaled upon completion, software and hardware waiters can be waiting on completion.
- Hardware fences might unblock other hardware.
- Object might have a single exclusive or multiple shared fences.

Reservation API (WIP!)

- Work in progress, not even the name is finalized.
- lockdep annotations have been added, will pick up most common errors.
- reservation_ticket for performing annotating multi-object reservations, passed to object_reserve.
- reservation_object is a primitive that is used for synchronization, and also contains pointers to fences.
- Eviction support is still a TODO!

Fence api rules

- When holding a reservation on a obj, the fence members can be read and written.
- Any fence calls must be made after reserving and before unreserving.
- Only one new fence needs to be allocated for all reservation buffers held.
- BUF_MAX_SHARED_FENCE shared slots, 1 exclusive slot.
- For a new shared fence, wait on the last exclusive fence before starting.
- If you request exclusive access access, you should wait on all previous shared fences before starting or if there are none, wait on the last exclusive fence.

Summary

1 Architecture

2 Demo

3 Questions

Summary

1 Architecture

2 Demo

3 Questions