Why are GPUs (not) fast

Lucas Stach – l.stach@pengutronix.de

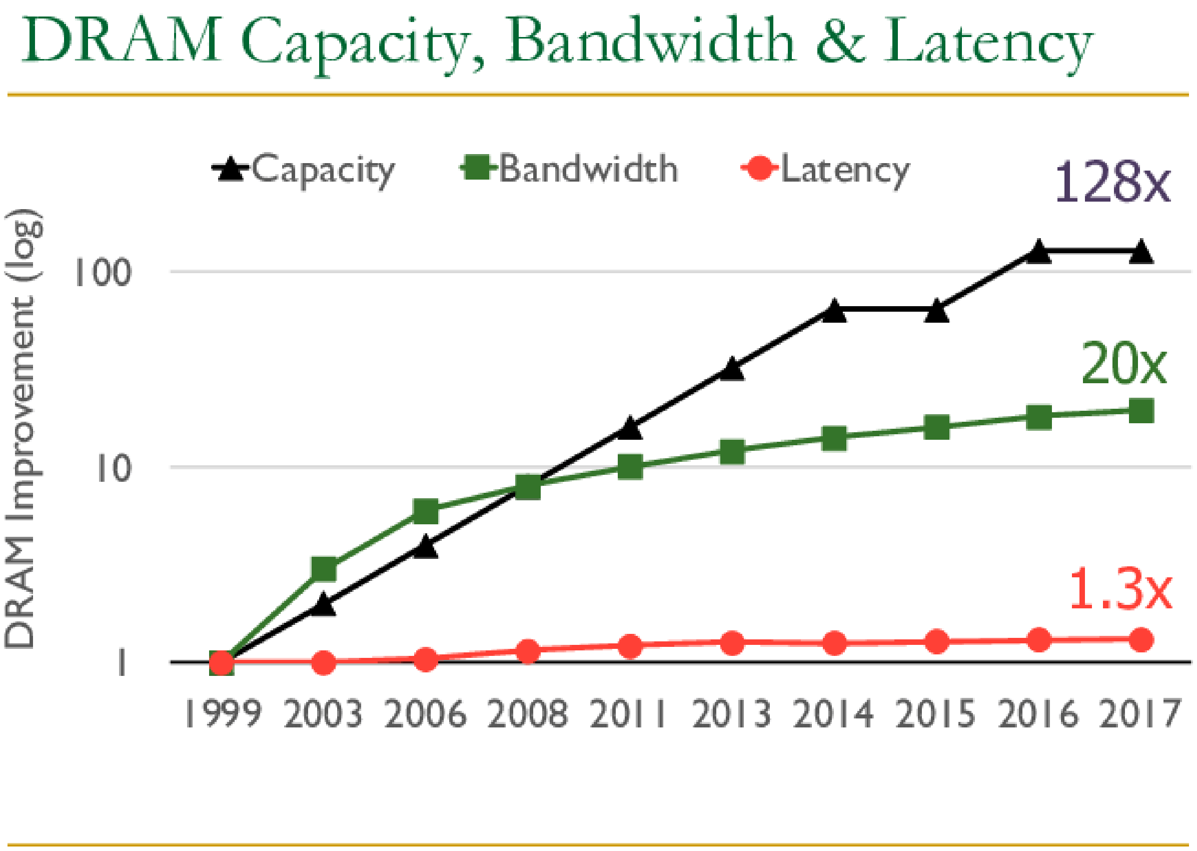
graphics stack developer @ Pengutronix

Magic?

Graphics Processing Units are magical go fast devices, right?

Well, no...

Deep down



DRAM Capacity, Bandwidth & Latency

Example:

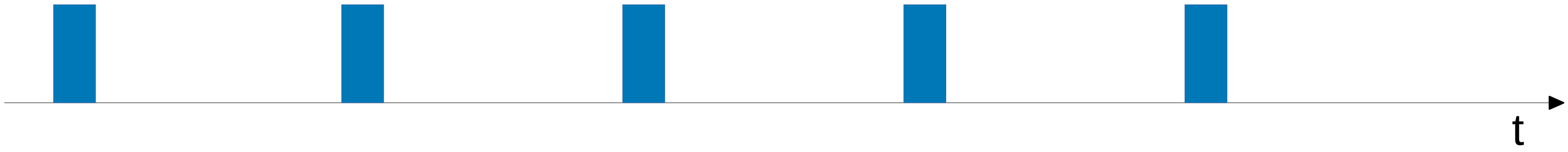
DDR4 at 4000 MT/s
32 Bit Bus
20 ns access time

16 GB/s
bandwidth*delay = 320 Bytes

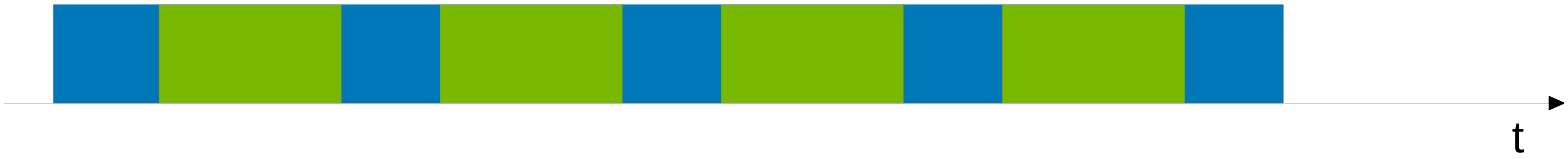
Mutlu, Onur & Ghose, Saugata & Gómez-Luna, Juan & Ausavarungnirun, Rachata. (2020). A Modern Primer on Processing in Memory.

Deep down (the memory lane)

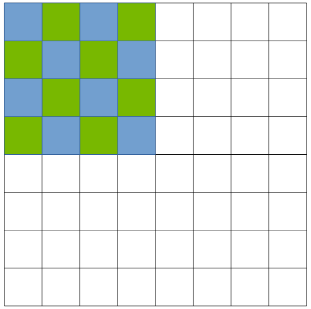
How to avoid the memory bus looking like this?



Caches



Throughput over latency

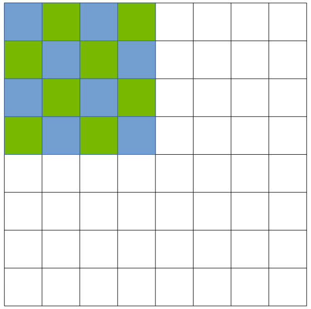


Filling a 2D grid of pixels

- Inherently parallel problem
- Latency matters only at the grid level

Spend HW resources on more, but less sophisticated execution engines

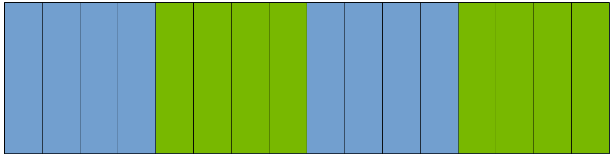
Throughput over latency



SIMT – single instruction multiple threads

- Multiple threads share one execution engine

shared register file



If threads use more registers, lower number of thread can be in flight

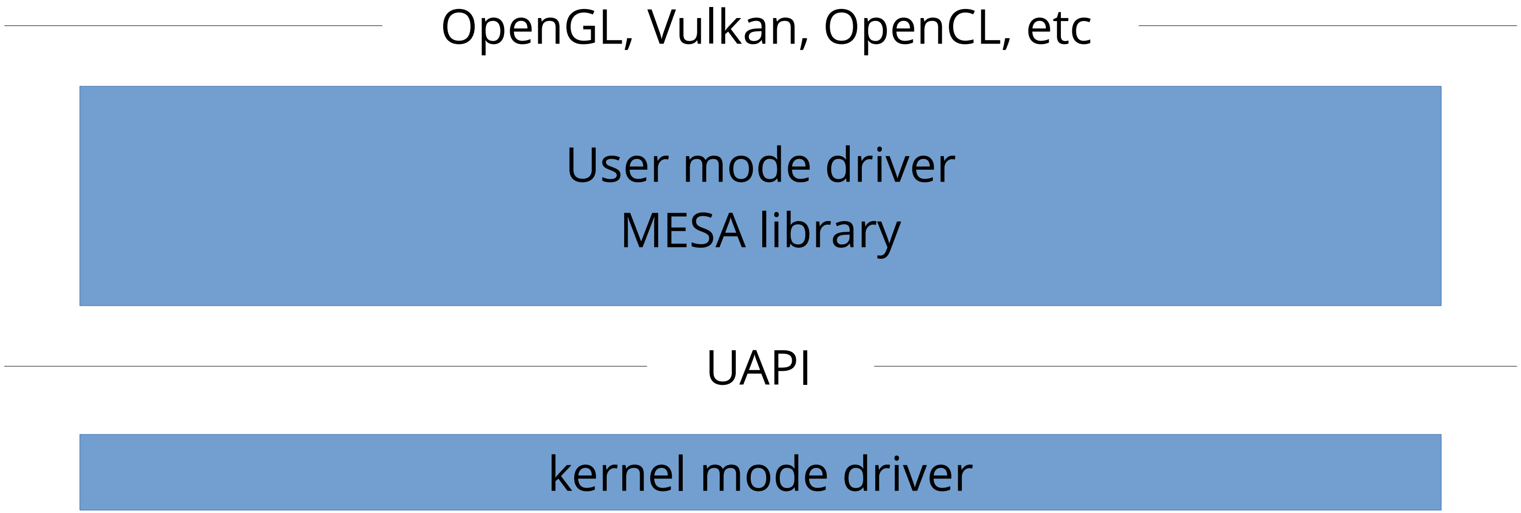
- Less opportunities to hide memory latency

GPU hardware

- Optimized for (ridiculously) parallel workloads
- Memory latency hiding
- Breaks down if problem isn’t parallelizable or individual strands are too complex

GPU drivers

Split between kernel and user mode



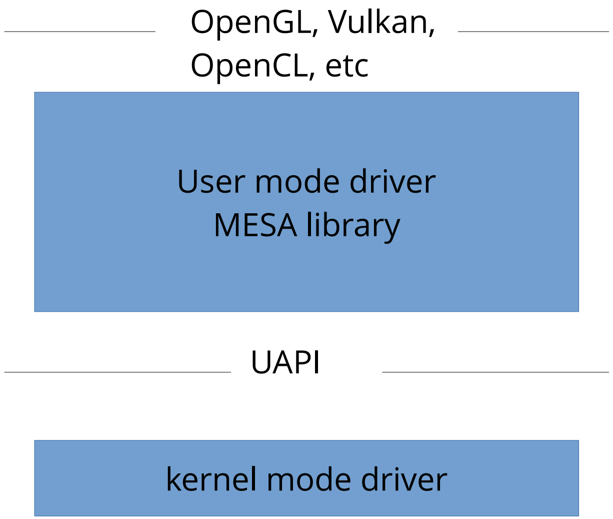
OpenGL, Vulkan, OpenCL, etc

User mode driver
MESA library

UAPI

kernel mode driver

GPU drivers



OpenGL, Vulkan, OpenCL, etc

User mode driver
MESA library

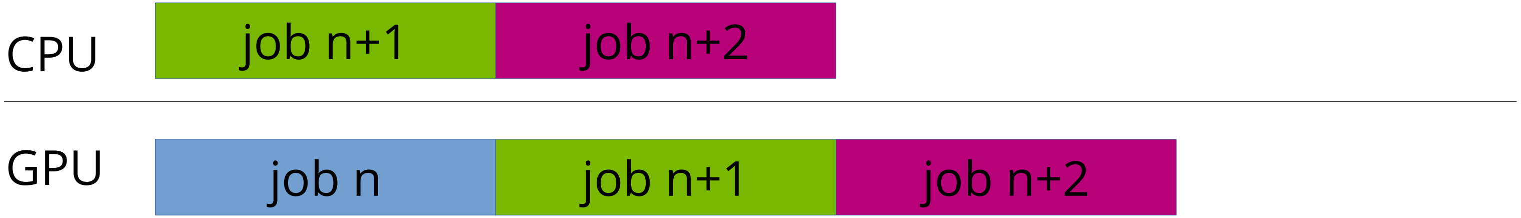
UAPI

kernel mode driver

- (Relatively) expensive submissions to hardware
- User mode driver amortizes cost via batching
- Reducing execution latency by forcing job submission (glFlush, vkQueueSubmit) increases cost (driver overhead)

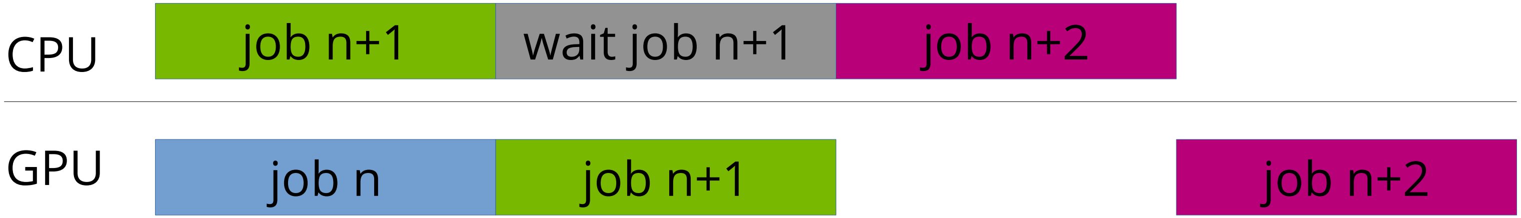
GPU drivers

GPU drivers optimize for throughput by allowing the CPU to get ahead of the GPU (pipelining)



GPU drivers

Synchronous waits for results (job finish, pixel data readback, etc) will create a pipeline bubble

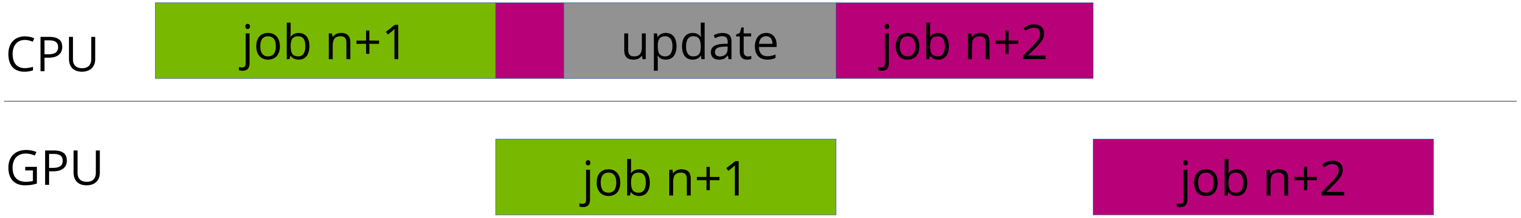


Whenever possible extend pipelining into application by using asynchronous interfaces

GPU drivers

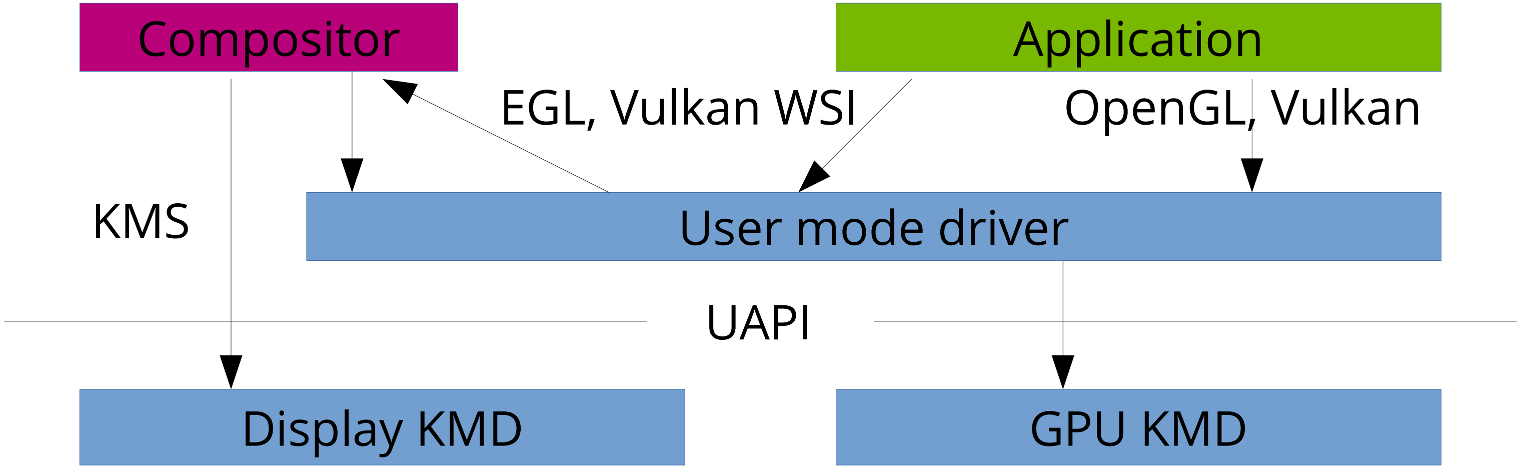
Updates of shared data can introduce pipeline bubble

Example: change texture data used by consecutive GPU jobs

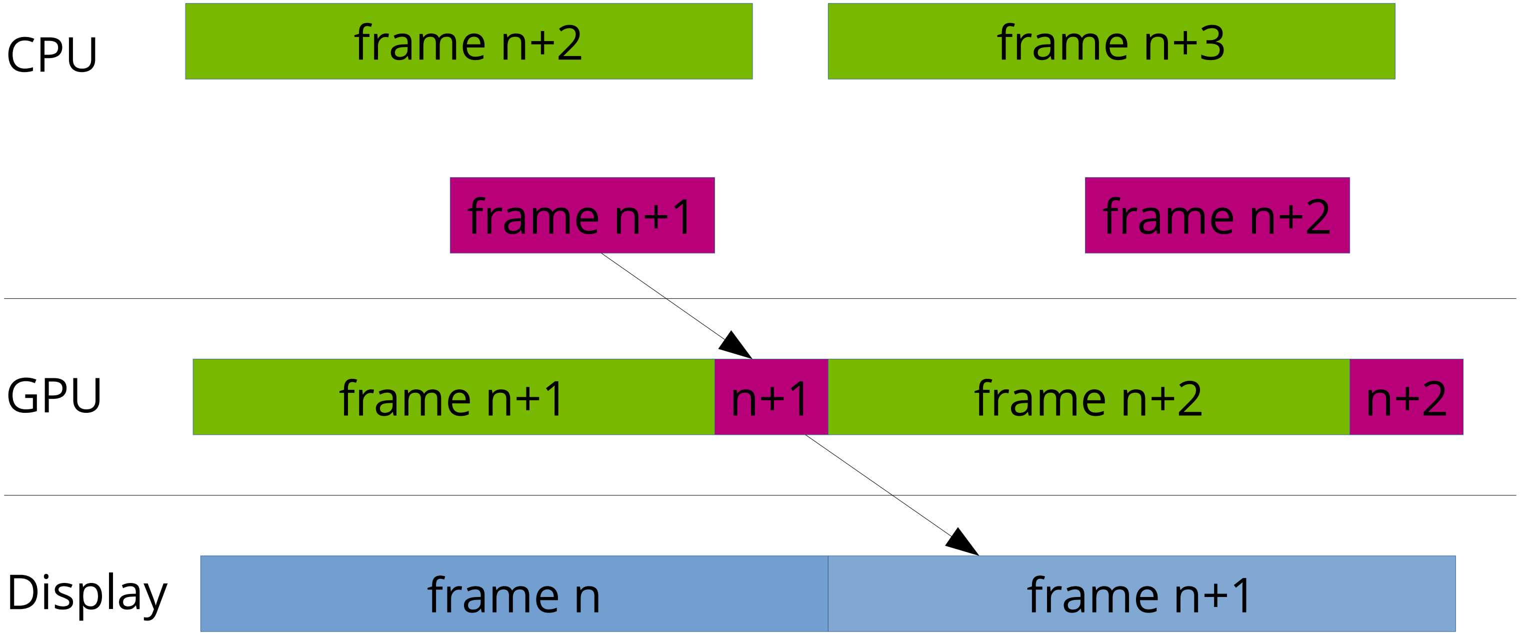


Display composition

How to get pictures on the screen



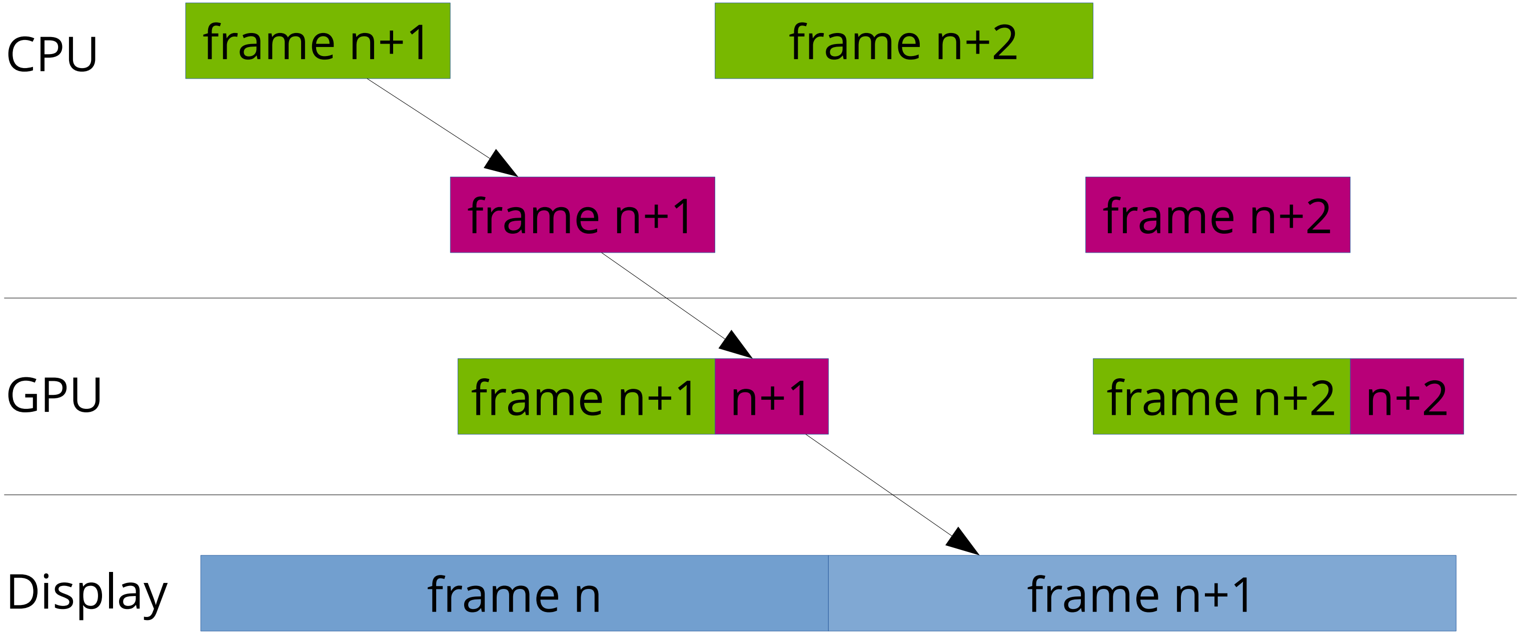
Display pipelining



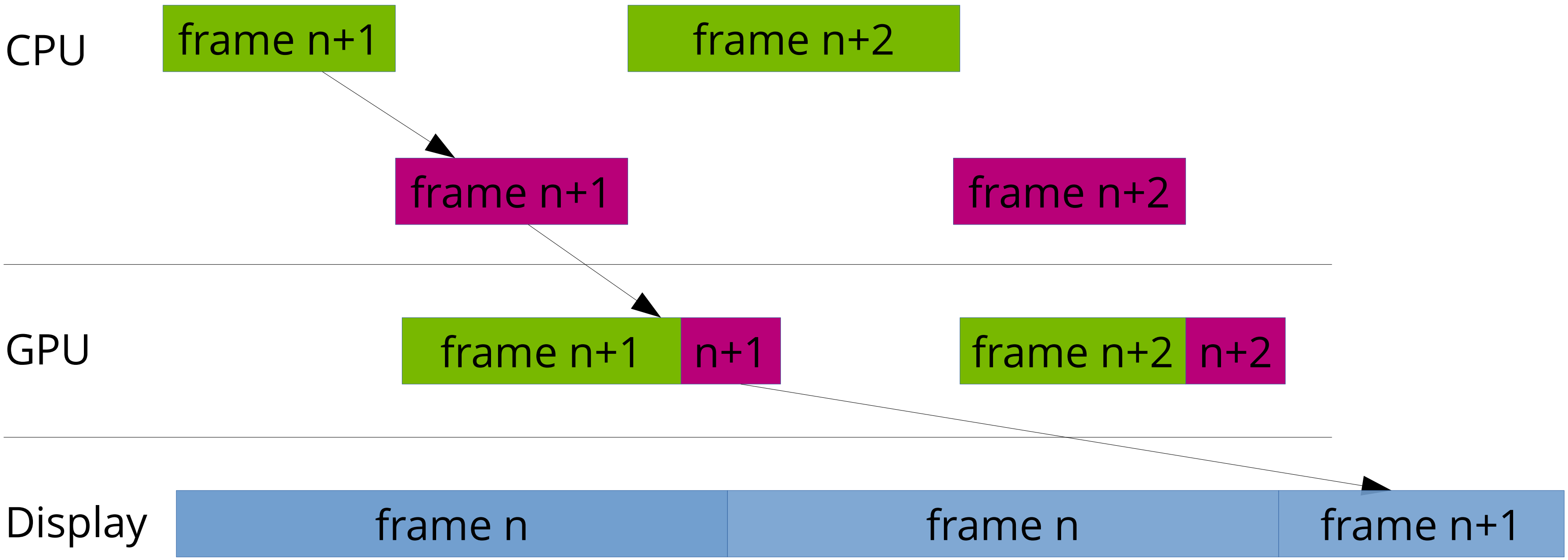
Display pipelining

- Pipelining keeps hardware busy
- Sacrifices latency (a lot) to gain throughput

Display latency reduction



Display latency reduction (failed)



GPU driver

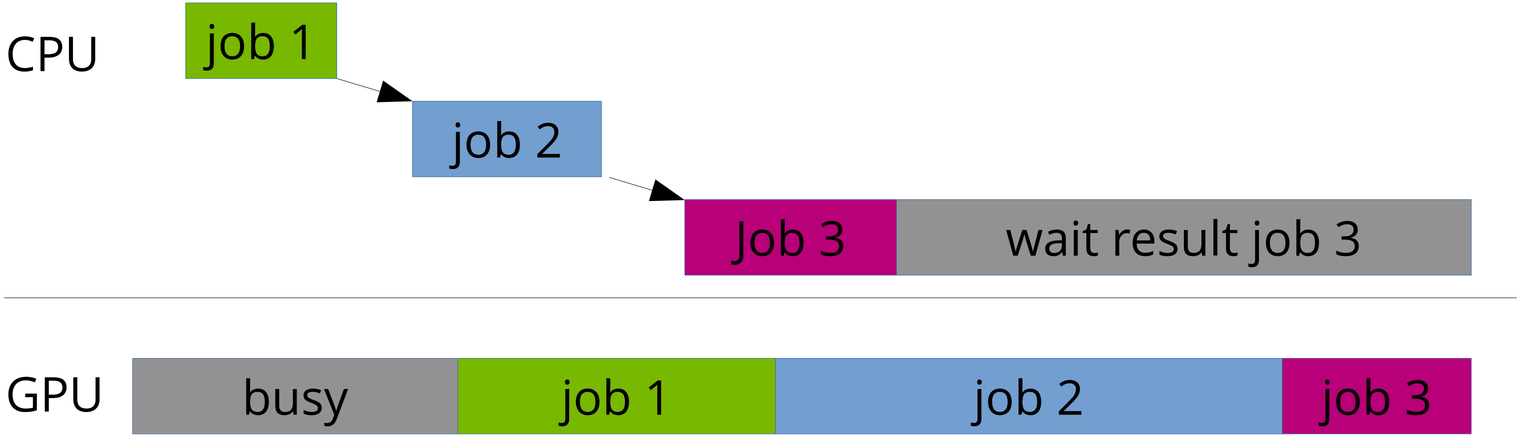
- Tuned to optimize throughput
- Latency reduction is possible to some degree
- Low latency at good hardware utilization rates is (really) hard

Bonus: fences

- Fences keep track of committed work

- Eventual completion guarantees

Bonus: fences


CPU
job 1
job 2
Job 3
wait result job 3
GPU
busy
job 1
job 2
job 3