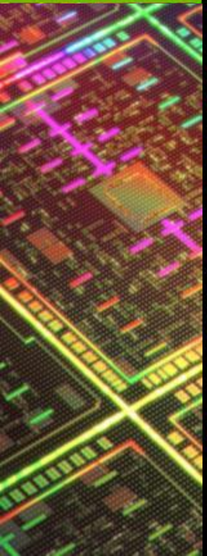
GPU TECHNOLOGY CONFERENCE

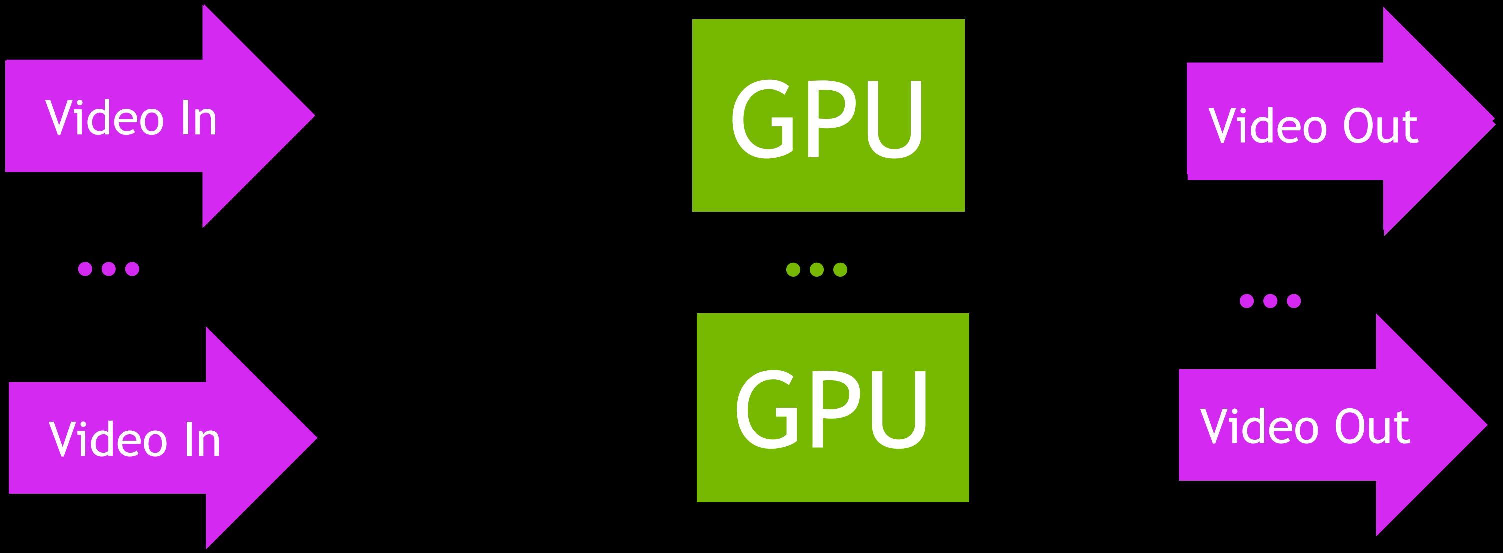
GPU Direct for Video

Agenda

- Video Processing Workflow
- Video I/O with GPU
- GPUDirect for Video
- Why Not Peer to Peer
- SDK Availability
- Conclusions



Video Processing Workflow



Video In

…

Video In

GPU

…

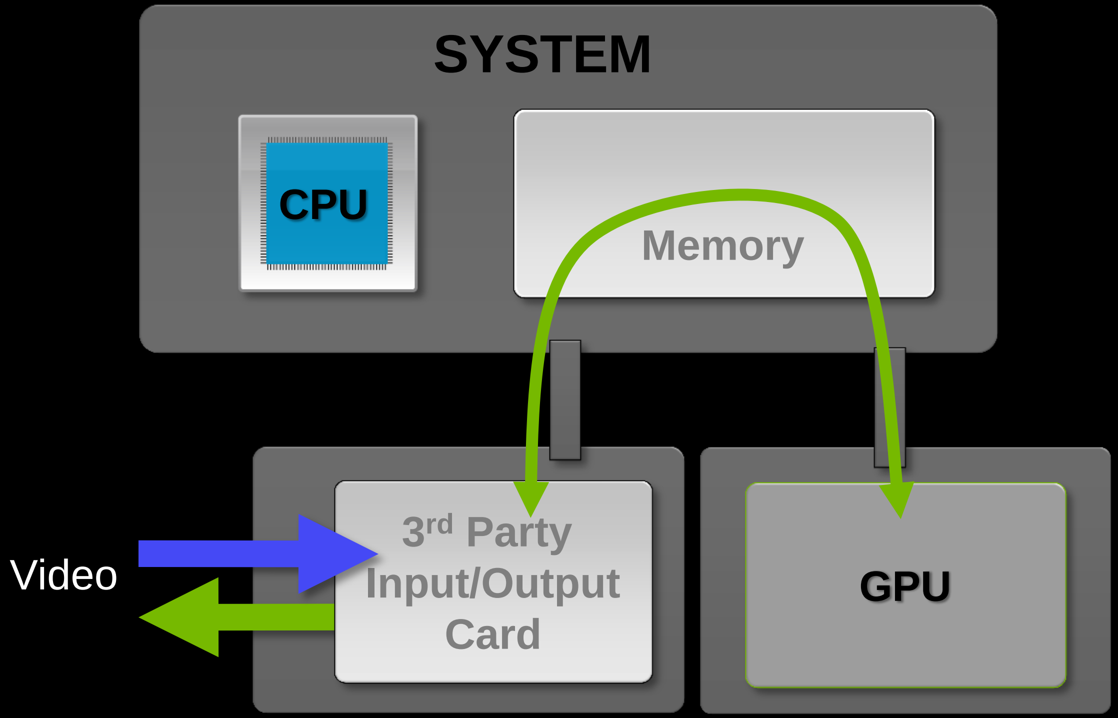
GPU

Video Out

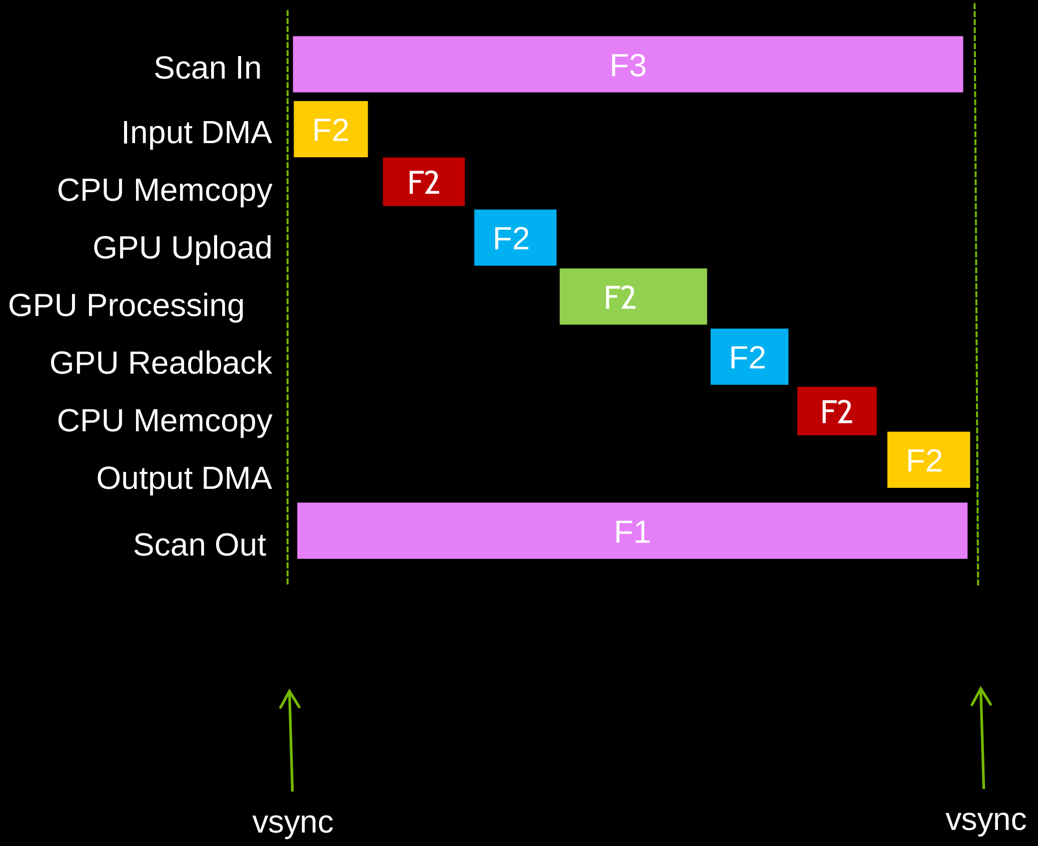
…

Video Out

Video I/O with GPU: transfer path



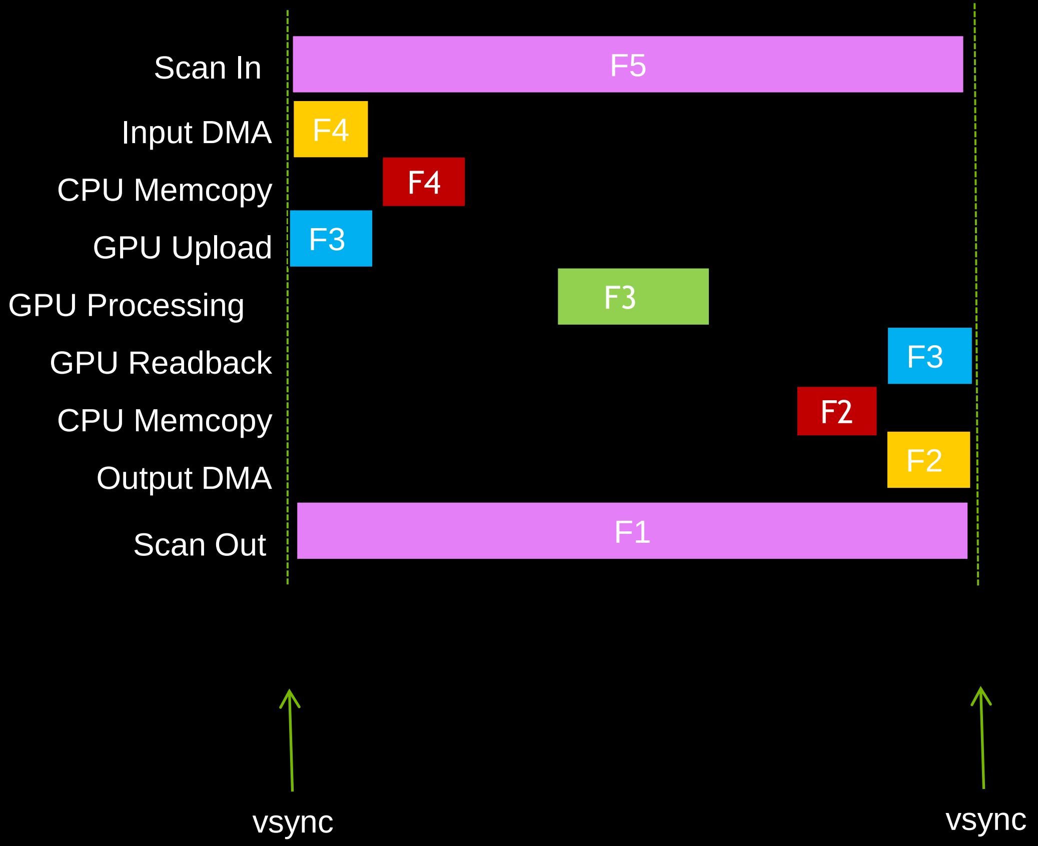
Video I/O with GPU: inefficiencies



2 frames of latency

- I/O <-> GPU through GPU unshareable system memory buffer
- GPU synchronous transfers

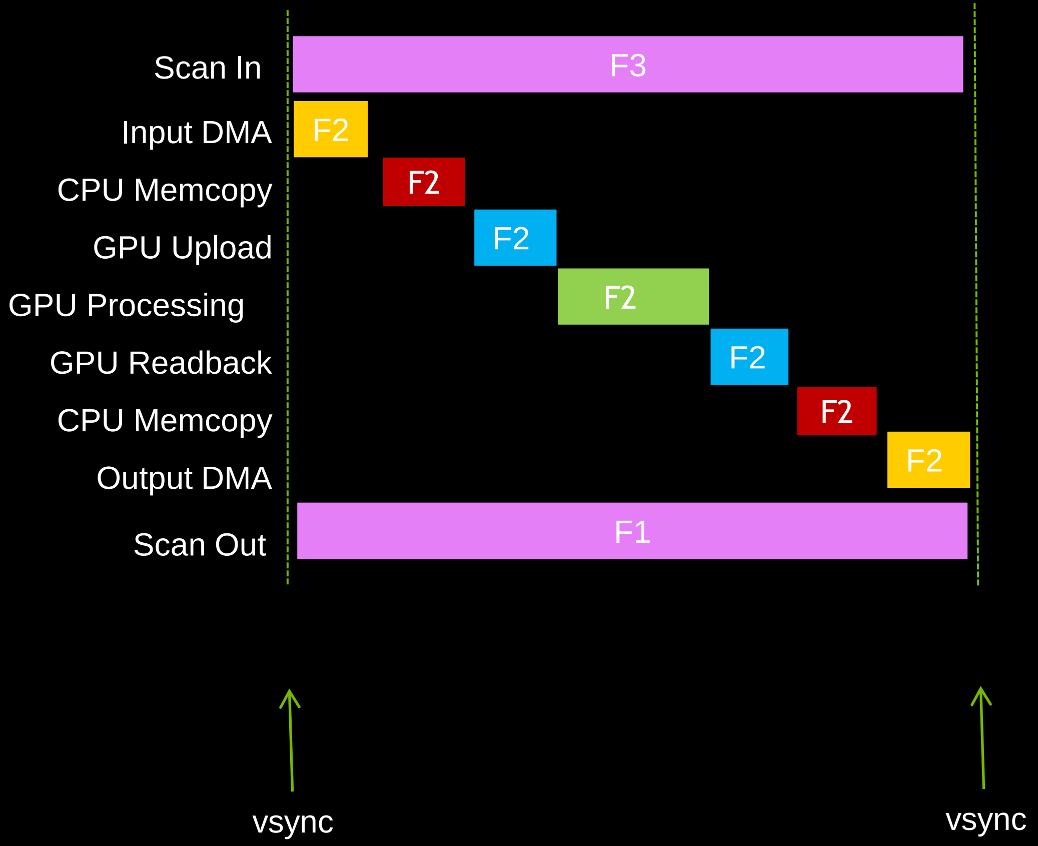
Video I/O with GPU: improvements



- Increasing latency

4 frames of latency!

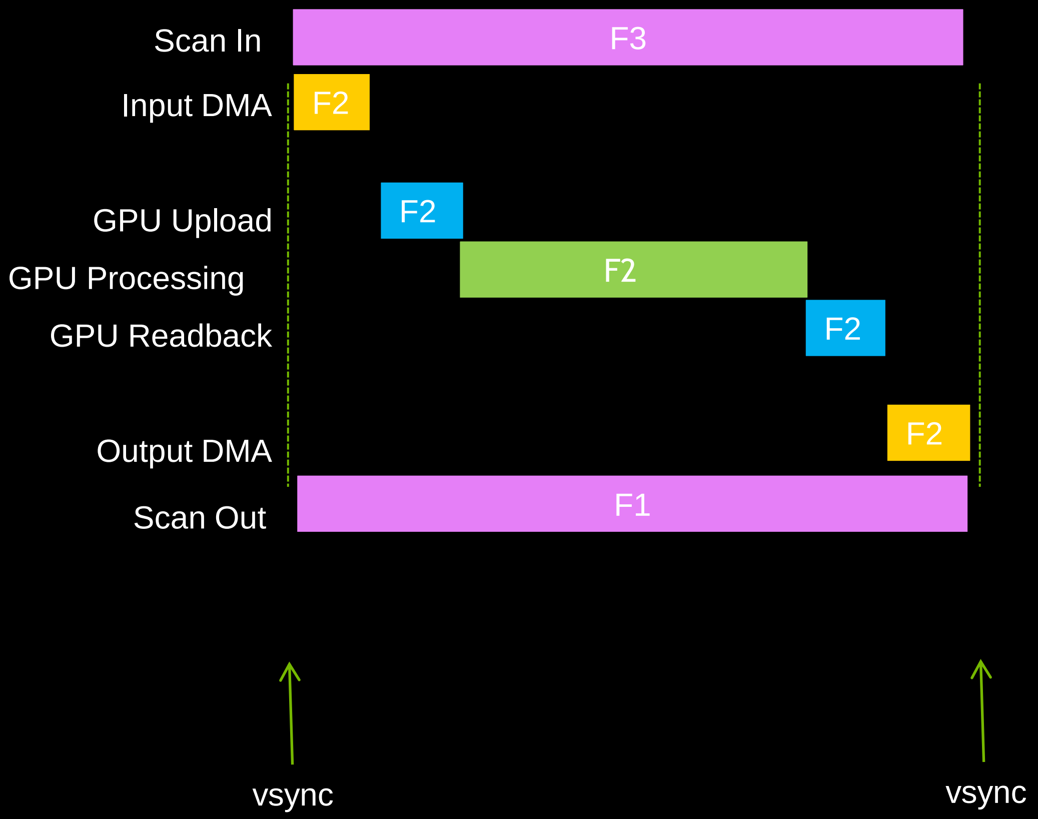
Video I/O with GPU: improvements



2 frames of latency

- I/O <-> GPU through GPU shareable system memory buffer

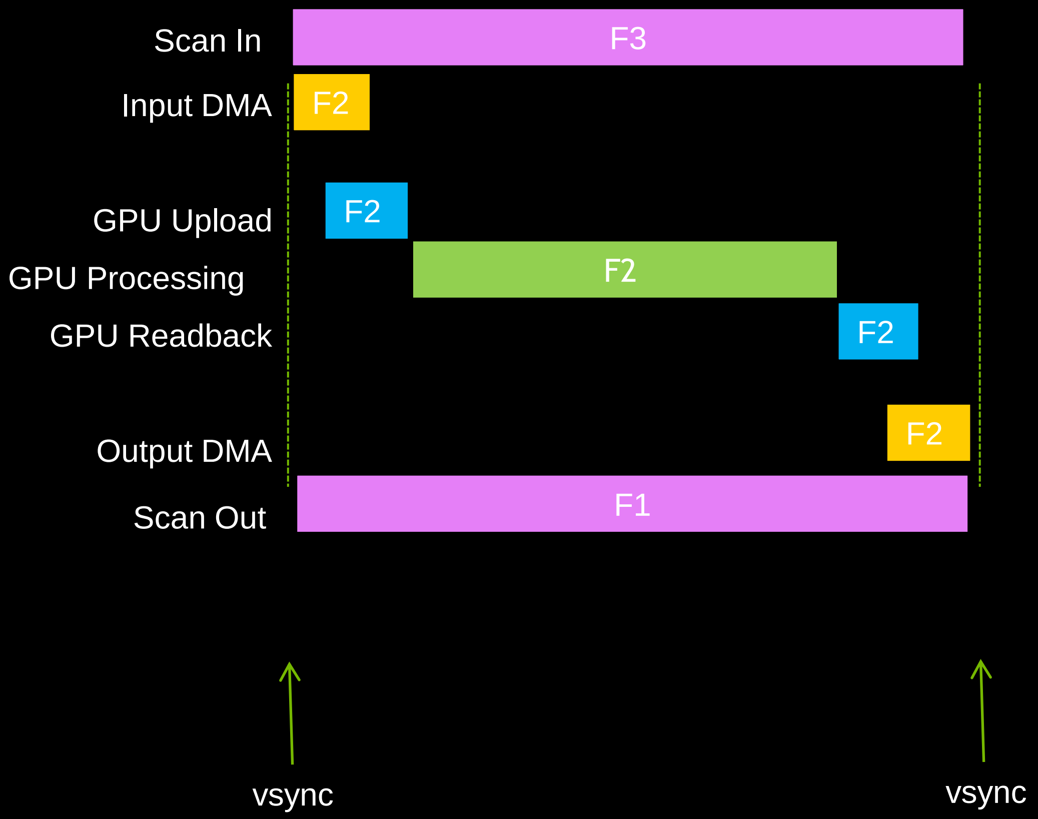
Video I/O with GPU: improvements



2 frames of latency

- Overlapping I/O and GPU DMAs by using:
  - sub-field chunks

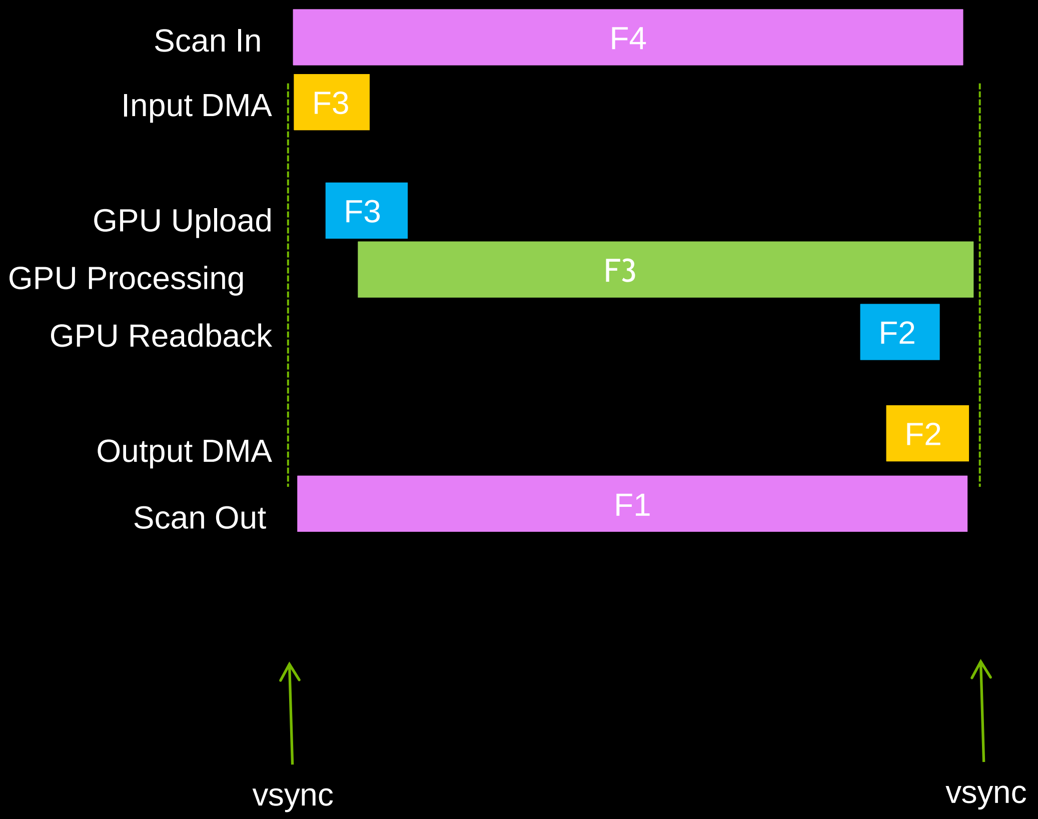
Video I/O with GPU: improvements



2 frames of latency

- Overlapping GPU DMAs and GPU processing by using:
  - sub-field chunks
  - GPU asynchronous transfers

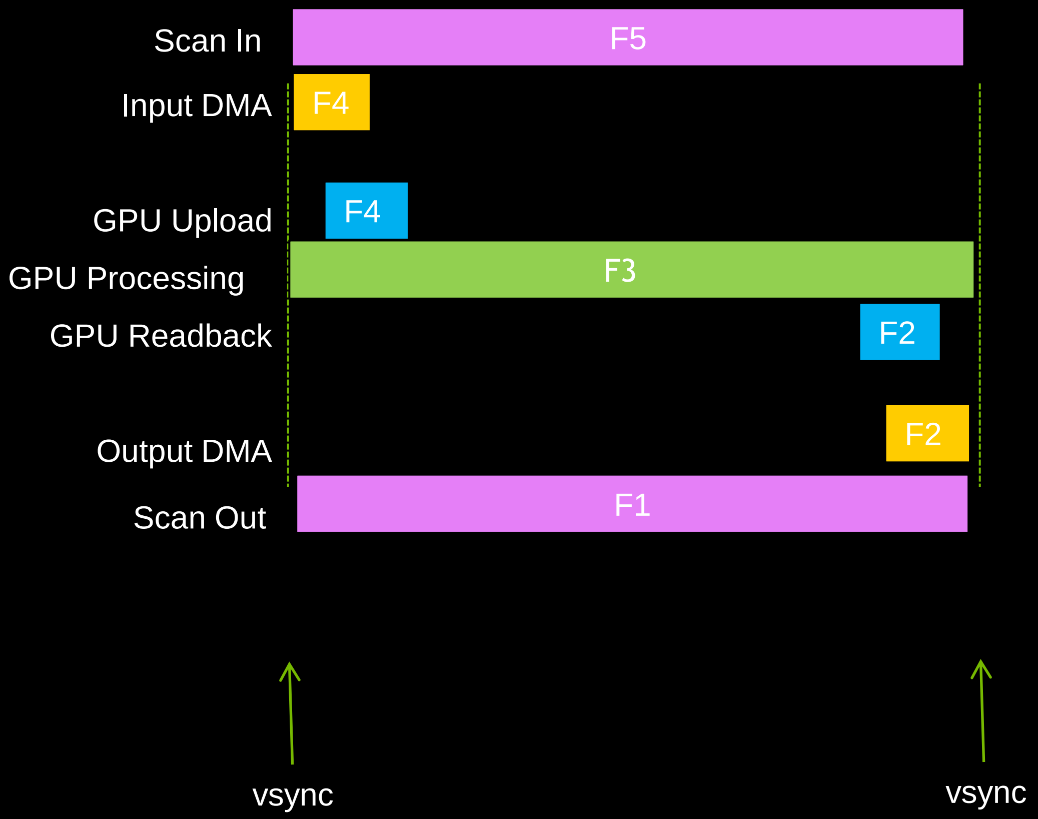
Video I/O with GPU: improvements



3 frames of latency

- Overlapping GPU DMAs and GPU processing by using:
  - GPU asynchronous transfers
  - Increasing latency

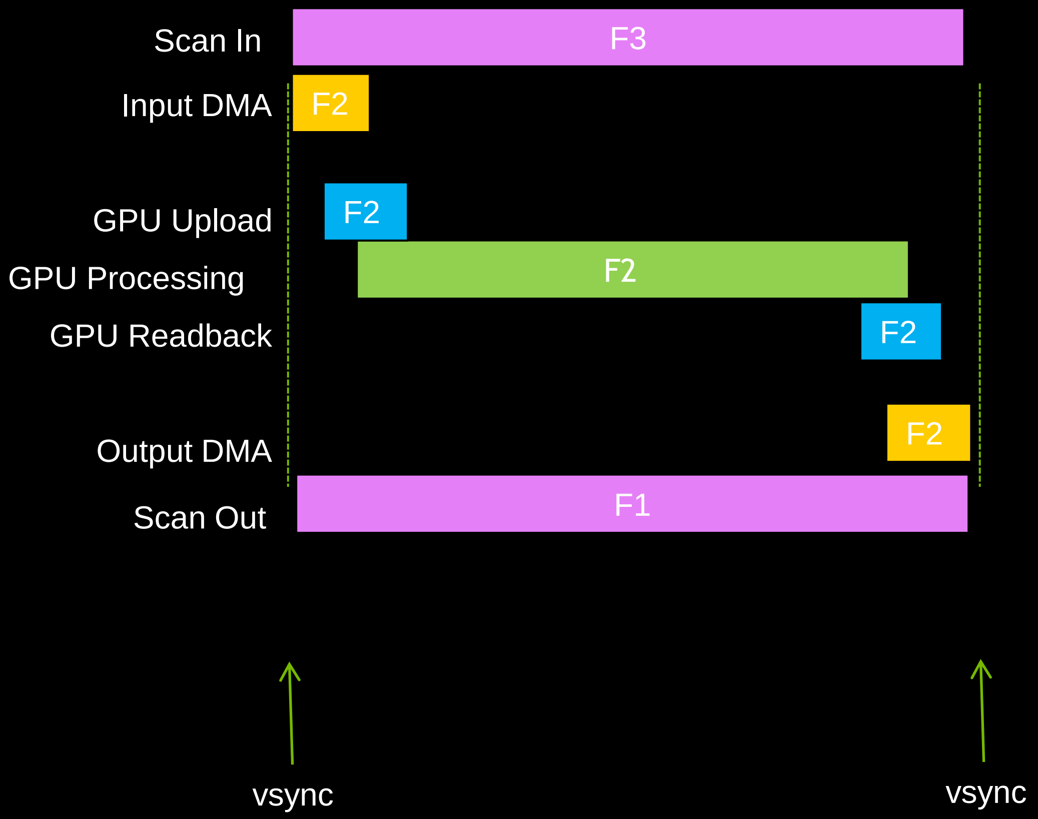
Video I/O with GPU: improvements



- Overlapping GPU DMAs and GPU processing by using:
  - GPU asynchronous transfers
  - Increasing latency even more

4 frames of latency = processing can occupy the entire frame!

GPUDirect for Video

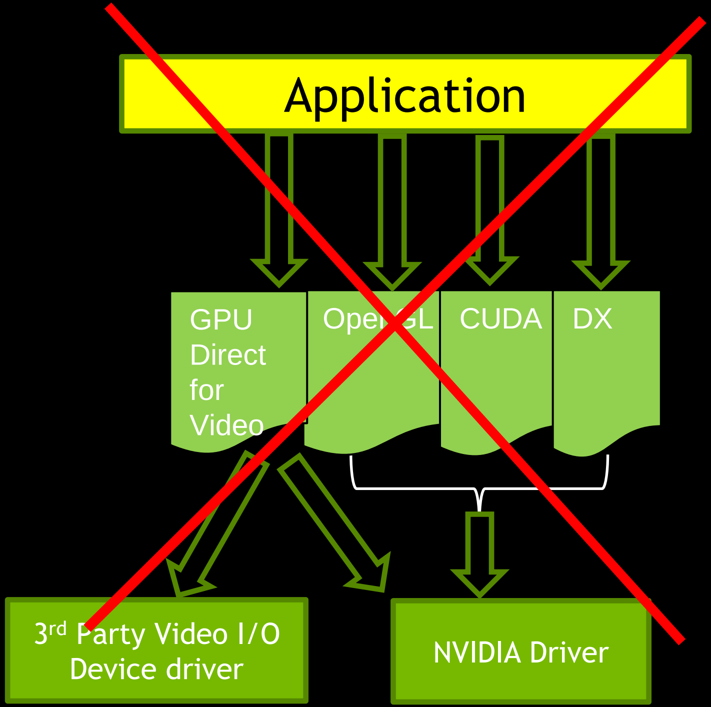


2 frames of latency!

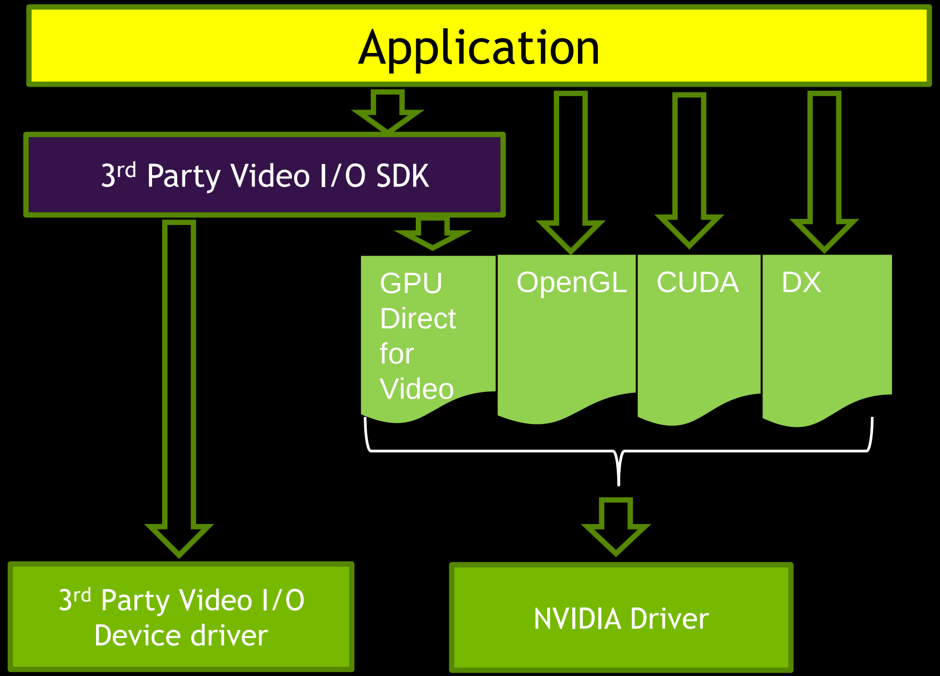
- Unified data transfer API for all Graphics and Compute APIs’ objects
  - Video oriented
  - Efficient synchronization
  - GPU shareable system memory
  - Sub-field transfers
  - GPU Asynchronous transfers

GPUDirect for Video: Application usage

- Not this



- But this:



GPUDirect for Video:Application Usage

Use the SDK Provided by Your Preferred Video I/O Vendor

Blackmagicdesign

AJA® VIDEO SYSTEMS

DELTACAST

DVS
A Rohde & Schwarz Company

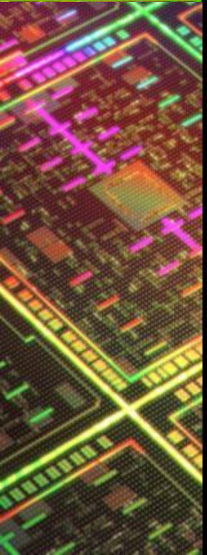
bluefish444

matrox
Graphics for Professionals

GPUDirect for Video: Application Usage

Video Capture to OpenGL Texture

```
main()
{
    …..
      GLuint glTex;
      glGenTextures(1, &glTex);   \\ Create OpenGL texture obect
      glBindTexture(GL_TEXTURE_2D, glTex);
      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, bufferWidth, bufferHeight, 0, 0, 0, 0);
      glBindTexture(GL_TEXTURE_2D, 0);
      EXTRegisterGPUTextureGL(glTexIn);  \\ Register texture with 3rd party Video I/O SDK
    while(!quit)
    {
        EXTBegin(glTexIn);  \\ Release texture from Video I/O SDK
        Render(glTexIn);  \\ Use the texture
        EXTEnd(glTexIn);  \\ Release texture back to Video I/O SDK
    }
    EXTUnregisterGPUTextureGL(glTexIn);  \\ Unregister texture with 3rd party Video I/O SDK
}
```



Results

Optimal transfer time for 4-component 8-bit 1080p video:

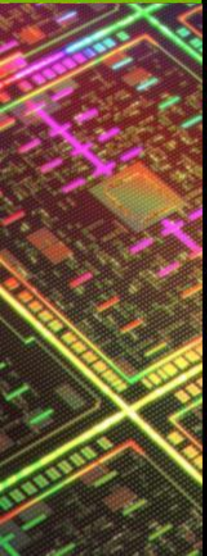
$$\textit{transfer time} \cong \frac{\textit{frame size}}{\textit{PCIE bandwidth}} * 2 \cong \frac{497664000 \textit{ bytes per second}}{6000000000^{*} \textit{ bytes per second}} * \frac{1}{60} * 2$$

$$\textit{transfer time} \cong 2.397 \textit{ msec}$$

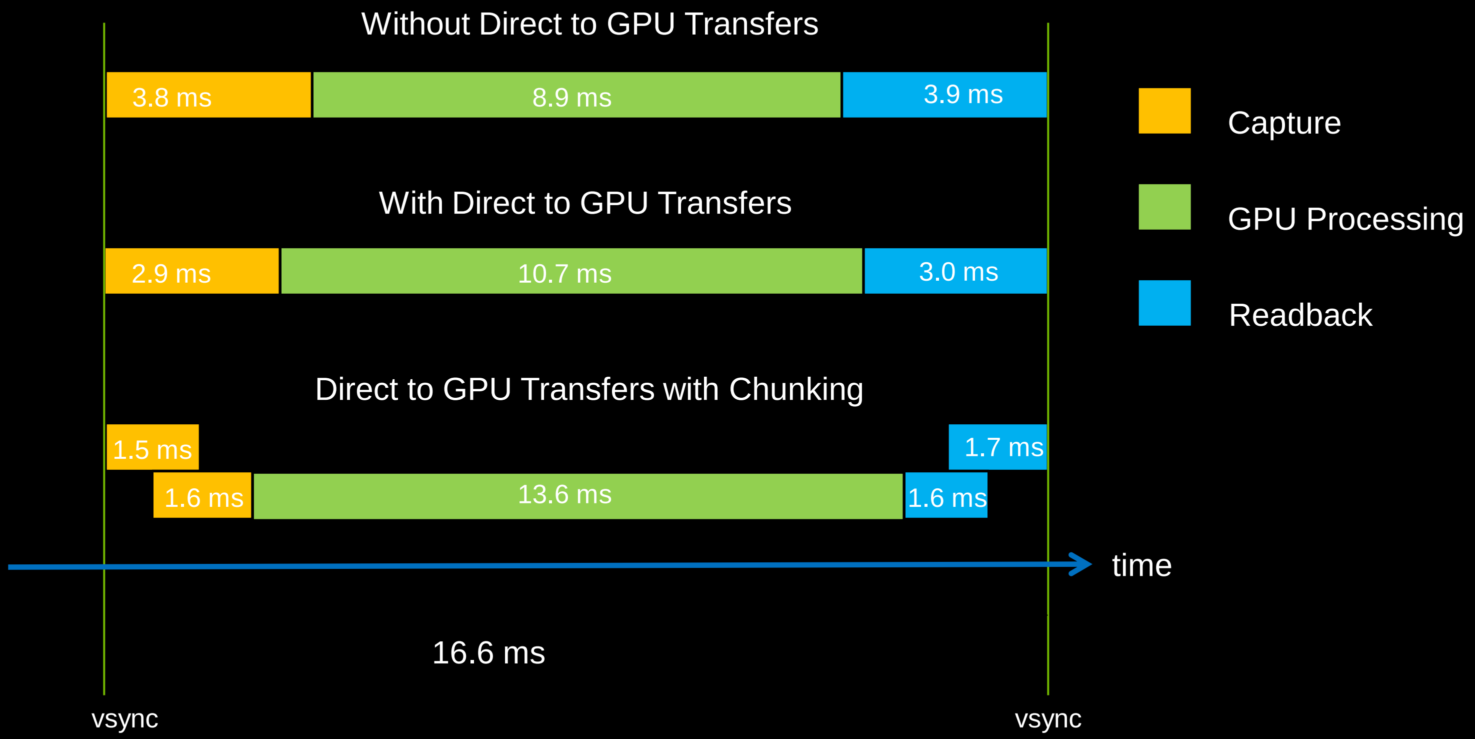
[*] Although Gen 2 PCI Express bandwidth is specified at 8.0GB / sec, the maximum achievable is ~6.0 GB/sec

Direct to GPU video transfer time for 4-component 8-bit 1080p video:

Capture Latency		Playout Latency	
ExtDev to SysMem	SysMem to GPU	GPU to SysMem	SysMem to ExtDev
1.4ms	1.5ms	1.5ms	1.5ms
2.9 ms		3.0ms	

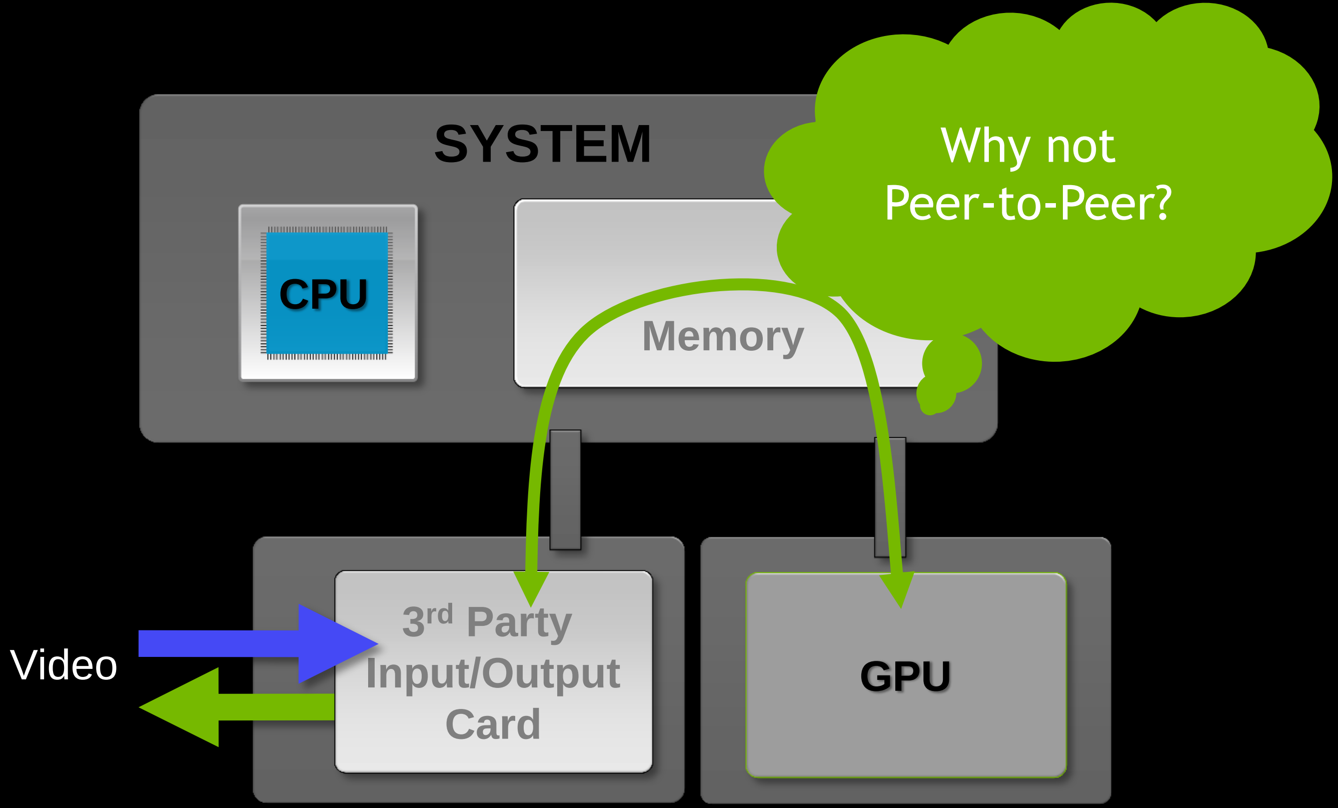


Results


Without Direct to GPU Transfers
3.8 ms
8.9 ms
3.9 ms
With Direct to GPU Transfers
2.9 ms
10.7 ms
3.0 ms
Direct to GPU Transfers with Chunking
1.5 ms
1.7 ms
1.6 ms
13.6 ms
1.6 ms
time
16.6 ms
vsync
vsync
Capture
GPU Processing
Readback

GPU TECHNOLOGY CONFERENCE

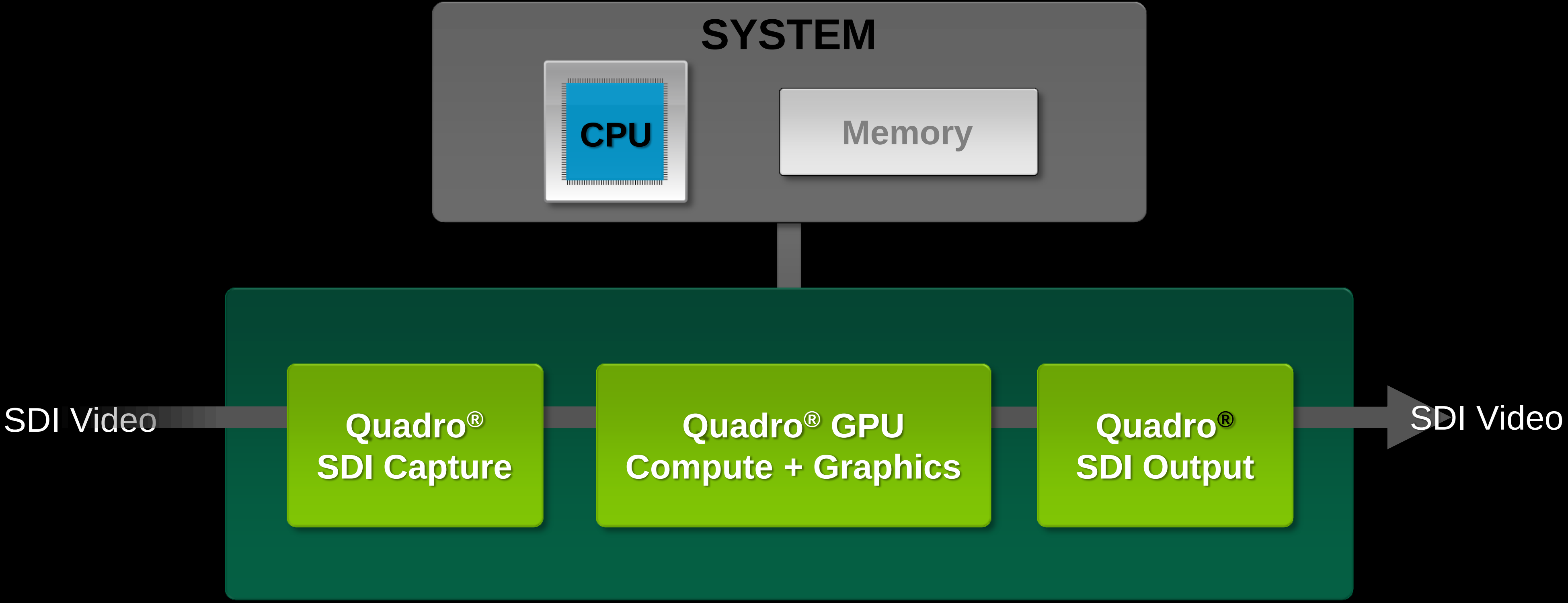
Video I/O with GPU: transfer path



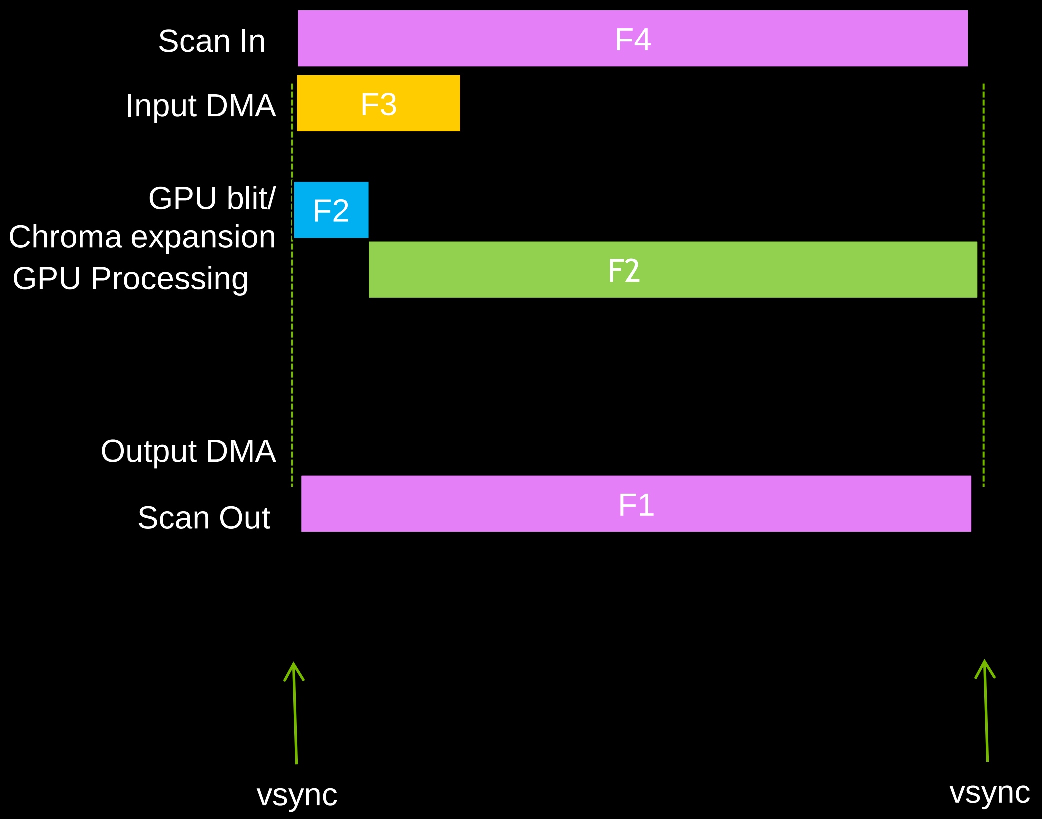
NVIDIA Digital Video Pipeline:

Peer-to-Peer Communication

NVIDIA SDI capture and output cards only



NVIDIA Digital Video Pipeline



3 frames of latency!

- The Input DMA is peer-to-peer of 4 inputs
- The output is a direct scanout from the GPU

BUT…

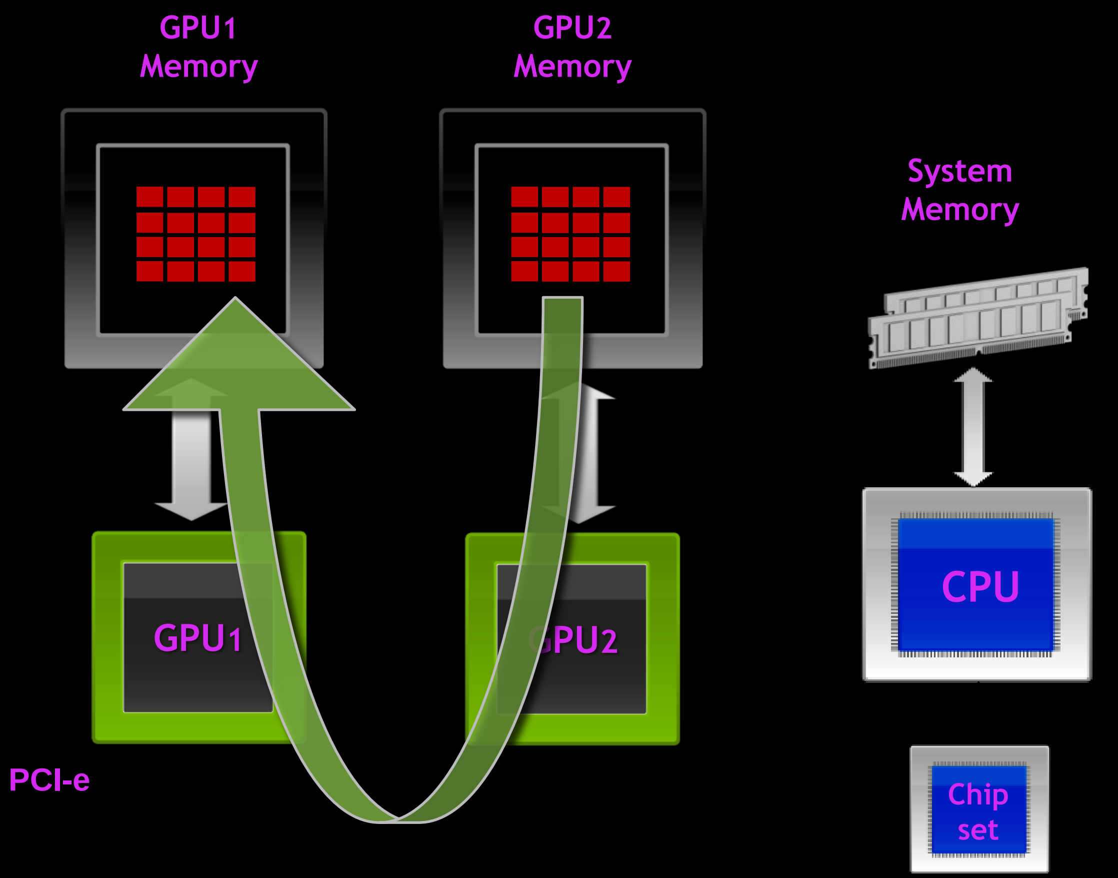
- Fixed 3 frames of latency
- Each I/O board must be tied to a single GPU

NVIDIA GPUDirect™ v2.0:
Peer-to-Peer Communication

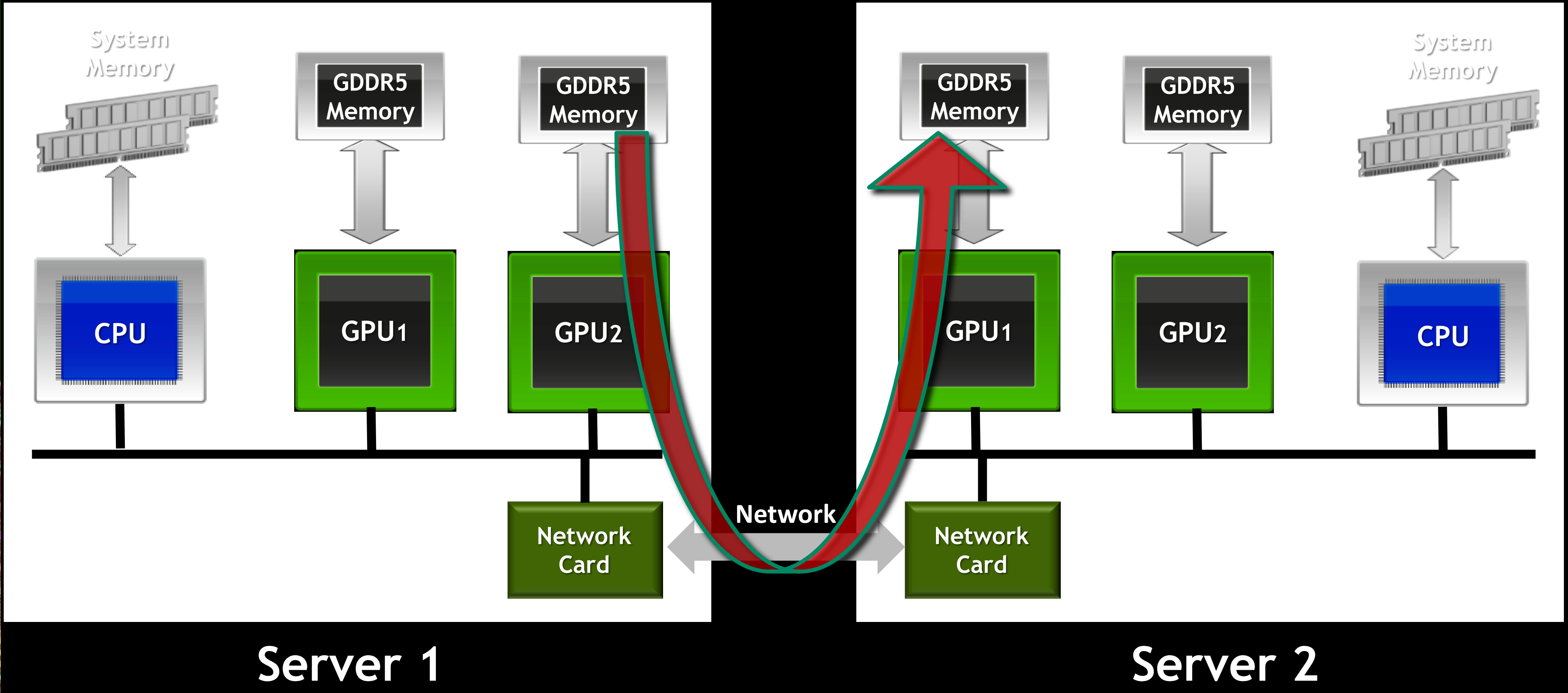
Direct Transfers between GPUs with CUDA

Only one copy required:

1. cudaMemcpy(GPU2, GPU1)



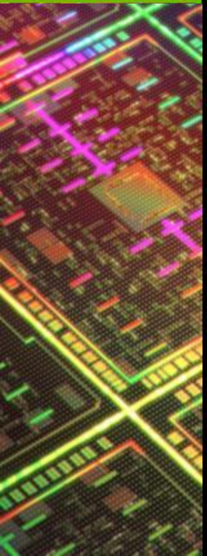
NVIDIA GPUDirect™ now supports RDMA



Linux only, HPC centric

Why Not Peer-to-Peer?

- Supportability
- Same performance
- IOH-to-IOH communication issues
- Limited PCIE slot options due to lane allocations
- Support Graphics APIs as well as CUDA
- Multi-GPU Support!
- Multi-OS support!

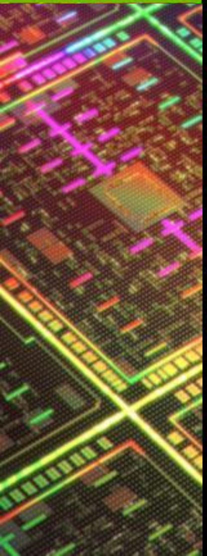


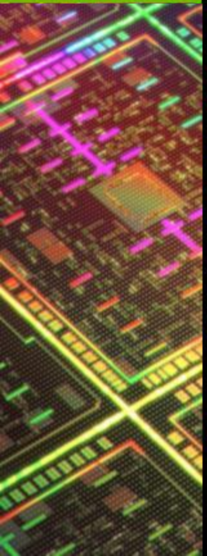
Video I/O Card Vendors

Use the NVIDIA GPU Direct for Video SDK:

http://developer.nvidia.com/nvidia-gpudirect%E2%84%A2-video

- Samples (OpenGL, D3D9, D3D11, CUDA)
- Programming Guide
- Windows7, Linux
- Static and Shared Libraries





Conclusions

- Lowest latency video I/O in and out of the GPU
- Optimal transfer times
- Optimal GPU processing time
- Supports OpenGL and DirectX as well as CUDA
- Does not require sophisticated programming.
- Scales to multiple GPUs