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 I/O with GPU: transfer path
Video I/O with GPU: inefficiencies

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

2 frames of latency
Video I/O with GPU: improvements

- Increasing latency

4 frames of latency!
Video I/O with GPU: improvements

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

2 frames of latency
Video I/O with GPU: improvements

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

2 frames of latency
Video I/O with GPU: improvements

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

2 frames of latency
Video I/O with GPU: improvements

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

3 frames of 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

- 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

2 frames of latency!
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
GPUDirect for Video: Application Usage
Video Capture to OpenGL Texture

main()
{
    ....
    GLuint glTex;
    glGenTextures(1, &glTex); \ Create OpenGL texture object
    glBindTexture(GL_TEXTURE_2D, glTex);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, bufferWidth, bufferHeight, 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:

\[
\text{transfer time} \approx \frac{\text{frame size}}{\text{PCIE bandwidth}} \times 2 \approx \frac{497664000 \text{ bytes per second}}{6000000000 \text{ bytes per second}} \times \frac{1}{60} \times 2
\]

\[\text{transfer time} \approx 2.397 \text{ msec}\]

* 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:

<table>
<thead>
<tr>
<th>Capture Latency</th>
<th>Playout Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtDev to SysMem</td>
<td>SysMem to GPU</td>
</tr>
<tr>
<td>1.4ms</td>
<td>1.5ms</td>
</tr>
<tr>
<td></td>
<td>2.9 ms</td>
</tr>
</tbody>
</table>
Results

Without Direct to GPU Transfers

<table>
<thead>
<tr>
<th>Capture</th>
<th>GPU Processing</th>
<th>Readback</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8 ms</td>
<td>8.9 ms</td>
<td>3.9 ms</td>
</tr>
</tbody>
</table>

With Direct to GPU Transfers

<table>
<thead>
<tr>
<th>Capture</th>
<th>GPU Processing</th>
<th>Readback</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9 ms</td>
<td>10.7 ms</td>
<td>3.0 ms</td>
</tr>
</tbody>
</table>

Direct to GPU Transfers with Chunking

<table>
<thead>
<tr>
<th>Capture</th>
<th>GPU Processing</th>
<th>Readback</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 ms</td>
<td>13.6 ms</td>
<td>1.6 ms</td>
</tr>
</tbody>
</table>
Video I/O with GPU: transfer path

Why not Peer-to-Peer?
NVIDIA Digital Video Pipeline:
Peer-to-Peer Communication

NVIDIA SDI capture and output cards only
NVIDIA Digital Video Pipeline

- The Input DMA is peer-to-peer of 4 inputs
- The output is a direct scanout from the GPU
  \textbf{BUT...}
- Fixed 3 frames of latency
- Each I/O board must be tied to a single GPU

\textbf{3 frames of latency!}
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

Server 1

Server 2

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