NVIDIA PROVIZ VR UPDATE

Robert Menzel, Ingo Esser
Siggraph 2018, August 14 2018
GRAPHICS PIPELINE

VR Workloads
GRAPHICS PIPELINE

VR Workloads

249M Pix/s
N vertices
30 Hz
(4K display)

792M Pix/s
2N vertices
90 Hz
(Vive Pro /w oversampling)
GRAPHICS PIPELINE

VR Workloads

249M Pix/s
N vertices
30 Hz
(4K display)

792M Pix/s
2N vertices
90 Hz
(Vive Pro /w oversampling)

6x
Application

6x
Driver

6x
Geometric Pipeline

3x
Rasterization Fragment Shader
INCREASE FRAMERATE

Why can’t we use SLI?

Application / API / Driver

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N+1</td>
<td>N+2</td>
<td>N+3</td>
</tr>
</tbody>
</table>

GPU 0

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N+1</td>
<td>N+2</td>
<td>N+3</td>
</tr>
</tbody>
</table>

Scanout

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N+1</td>
<td>N+2</td>
<td>N+3</td>
</tr>
</tbody>
</table>

Display Update (Photons)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N+1</td>
<td>N+2</td>
<td>N+3</td>
</tr>
</tbody>
</table>

Latency
INCREASE FRAMERATE

Why can’t we use SLI?

Application / API / Driver

GPU 0

Scanout

Display Update (Photons)

Latency
INCREASE FRAMERATE

Why can’t we use SLI?

Application / API / Driver

- N
- N+1
- N+2
- N+3

GPU 0

- N
- N+2

GPU 1

- N+1
- N+3

Scanout

- N
- N+1
- N+2
- N+3

Display Update (Photons)

- N
- N+1
- N+2
- N+3

Latency
INCREASE FRAMERATE

Dual GPU

Application / API / Driver

GPU 0

GPU 1

Scanout

Display Update (Photons)

Latency
INCREASE FRAMERATE

Dual GPU

- **One** Command Buffer, different viewmatrix
- Send this Command Buffer to each GPU
- Copy result from GPU 1 to GPU 0
- Present rendering on GPU 0 to VR run-time
INCREASE FRAMERATE

GPU Affinity vs. VR SLI

- OpenGL (NVIDIA Quadro)
- 2 OpenGL contexts
- Full CPU overhead of 2nd context
- Full flexibility
- Allowing different objects per GPU

- All modern APIs
- 1 context
- Low CPU overhead
- Only masking draw calls
- Allowing different buffer / texture content per GPU
VR SLI
Crash course

Left view data

Geometry
Materials

Right view data
VR SLI
Scaling 1 vs 2 GPUs

App Left
GPU L

App Right
GPU R

App Both
GPU L
GPU R
Copy

Time: GPU L + GPU R
Time: GPU + Copy

Scaling = \frac{2 \times \text{GPU}}{\text{GPU} + \text{Copy}}
VR SLI
Scaling determined by workload and copy time

\[ Scaling = \frac{2 \times GPU}{GPU + Copy} \]

Typical render resolution for Vive
1512 x 1680 (per eye)

Copy time over PCIe (@6GB/s)
1.5ms

Max scaling with 11ms frame time

\[ \frac{2 \times 9.5ms}{9.5ms + 1.5ms} = 1.72 \]

x1.72
VR SLI
Higher resolutions limit scalability

Scaling = \frac{2 * GPU}{GPU + Copy}

Vive Pro render resolution
2016 x 2240

Copy time over PCIe (@6GB/s)
2.8ms

Max scaling with 11ms frame time
\frac{2*8.2ms}{8.2ms+2.8ms} = 1.49
**VR SLI**

Improve scaling using NVLink

Copy times can hurt scaling with higher resolutions

NVLink on dual Quadro GP100: *4x faster* than PCIe 3.0

Copy time for Vive Pro (2016 x 2240): 0.7ms

Max scaling with 11ms frame time

\[
\frac{2 \times 10.3m}{10.3ms + 0.7ms} = 1.87
\]

NVLink is used automatically if present

NVLink speed measured with 2 bridges, copy via OpenGL multicast, single frame of HTC Vive, on HP z840 workstation
OPENGL VR SLI: MULTICAST 2
Feedback on Multicast led to new functionality

Command & data broadcast
BufferSubData to specific GPU
CopyImageSubData & CopyBufferSubData
GPU-GPU Framebuffer Blit
Global barrier & directed sync functions
GPU Masks
Per-GPU sample locations
Per-GPU queries

Dynamic Multicast toggle (WGL_NV_multigpu_context)
GPU_ID built-in in GLSL shader
Per-GPU viewports & scissors
Texture & Buffer upload mask
Asynchronous copies
New extension WGL_NV_multigpu_context: Request SLI mode per context

No need to restart application

Possible to share resources between contexts
MULTICAST 2
Dynamic SLI mode

New extension WGL_NV_multigpu_context: Request SLI mode per context

No need to restart application

Possible to share resources between contexts

On toggle:

Clean up per-GPU resources

Keep scene data

Alternate Frame Rendering (AFR)
Multicast v1 required per-GPU uploads
Larger code changes in some renderers
Add shader built-in: `gl_DeviceIndex`
- Upload all views to all GPUs
- Use per-GPU data in shaders
Renderer can remain unchanged
Just modify shaders instead
MULTICAST 2
Per-GPU Viewports & Scissors

Add new function to set viewports and scissors per GPU

```c
glMulticastViewportArrayvNVX( ... );
glMulticastScissorArrayvNVX( ... );
```

Per-GPU Lens Matched Shading
MULTICAST 2
Per-GPU Viewports & Scissors

Add new function to set viewports and scissors per GPU

```c
glMulticastViewportArrayvNVX( ... );
glMulticastScissorArrayvNVX( ... );
```

Per-GPU Lens Matched Shading

Per-GPU Multi Resolution Shading
MULTICAST 2
Per-GPU Viewports & Scissors

Add new function to set viewports and scissors per GPU

```cpp
glMulticastViewportArrayvNVX( ... );
glMulticastScissorArrayvNVX( ... );
```

Per-GPU Lens Matched Shading

Per-GPU Multi Resolution Shading

Easily set up Split Frame Rendering (SFR)
MULTICAST 2
Texture & Buffer Upload Mask

Multicast provides per-GPU buffer uploads

Asymmetrical functionality wrt texture upload functions

Add new mask function to modify texture & buffer uploads

```c
glfwUploadGpuMaskNVX( GLbitfield mask );
```

Useful for simpler per-GPU texture streaming

Conserve PCIe bandwidth
MULTICAST 2
Asynchronous Copies

Multicast copies stall source GPU while copy takes place

Easy to use because of implicit synchronization

New copy functions do not stall, but also need more synchronization

```c
glAsyncCopyBufferSubDataNVX(...);
```

```c
glAsyncCopyImageSubDataNVX(...);
```

Copy while both GPUs can continue rendering

Allows for more complex rendering algorithms
Update: `VK_KHX_device_group` ratified to `VK_KHR_device_group` with Vulkan 1.1

Make sure to use the right extension/Vulkan version combination!

Usage is the same, so migration is painless
VR SLI covers a wide variety of workloads.

Almost perfect load balancing between left/right eye and two GPUs.

Copy overhead and view independent workloads limit scaling.

NVLink can help improve scaling.
OUTLOOK
Pushing the Limits

Professional users demand higher resolutions

Professional HMDs reach the limits of DP bandwidth

StarVR HMD image courtesy of Starbreeze
OUTLOOK

Pushing the Limits

Professional users demand higher resolutions

Professional HMDs reach the limits of DP bandwidth
OUTLOOK
Pushing the Limits

Professional users demand higher resolutions
Professional HMDs reach the limits of DP bandwidth

StarVR HMD image courtesy of Starbreeze
OUTLOOK
Pushing the Limits

Professional users demand higher resolutions

Professional HMDs reach the limits of DP bandwidth

StarVR HMD image courtesy of Starbreeze
OUTLOOK
Pushing the Limits

StarVR HMD image courtesy of Starbreeze
OUTLOOK
Pushing the Limits

Opens new possibilities

StarVR HMD image courtesy of Starbreeze
MORE BOOTH TALKS @ SIGGRAPH 2018

https://www.nvidia.com/en-us/events/siggraph/schedule/?session-type=talk

Tue, 4:00 PM  Tackling the Realities of Virtual Reality (David Luebke)
Wed, 5:30 PM  NVIDIA Holodeck Update (Dave Weinstein)
Thu, 2:00 PM  How VirtualLink Drives a Revolution in VR Connectivity (Rambo Jacoby)
TRY IT OUT!
..and more information

NVIDIA VRWorks SDK provides OpenGL, Direct3D & Vulkan samples

developer.nvidia.com/vrworks

More detail in our previous GTC talks:

2018 - S8695 - NVIDIA VR Update
2017 - S7191 - Vulkan Technology Update
2016 - S6338 - VR Multi GPU Acceleration Featuring Autodesk VRED
2015 - S5668 - VR Direct: How NVIDIA Technology Is Improving The VR Experience
THANK YOU!

Booth #801, West Hall
Nvidia.com/siggraph2018