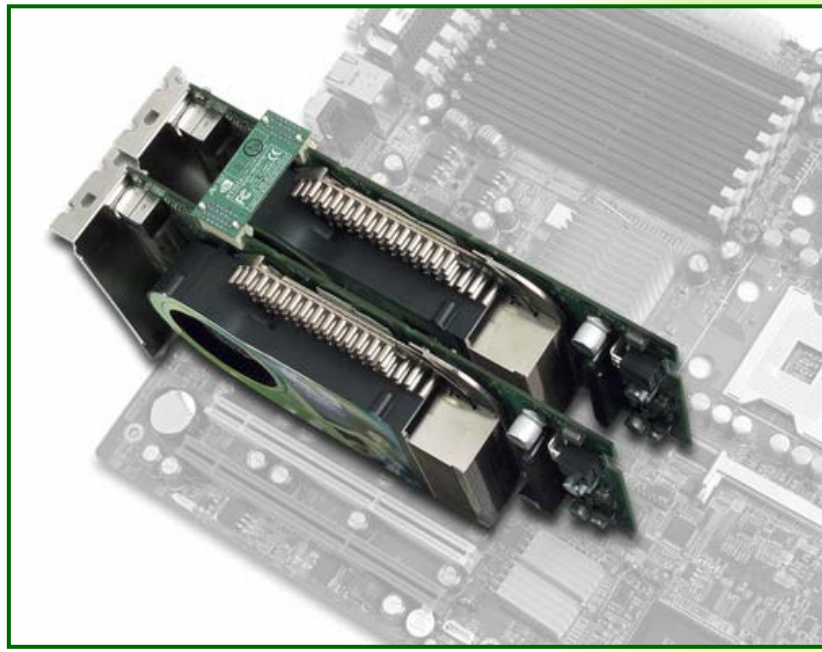
NVIDIA

Simon Green

NVIDIA Corporation

SLI: Scalable Link Interface

- NVIDIA's scalable graphics solution
- Uses multiple PCI-Express GPUs in one machine
- Improves rendering performance up to 1.9x with two GPUs



How SLI Works

- Plug 2 identical GPUs into motherboard
- NVIDIA driver reports as one logical device
  - Video memory does NOT double
- Video scan out happens from one board
  - Bridge connector transmits digital video between boards

SLI Market

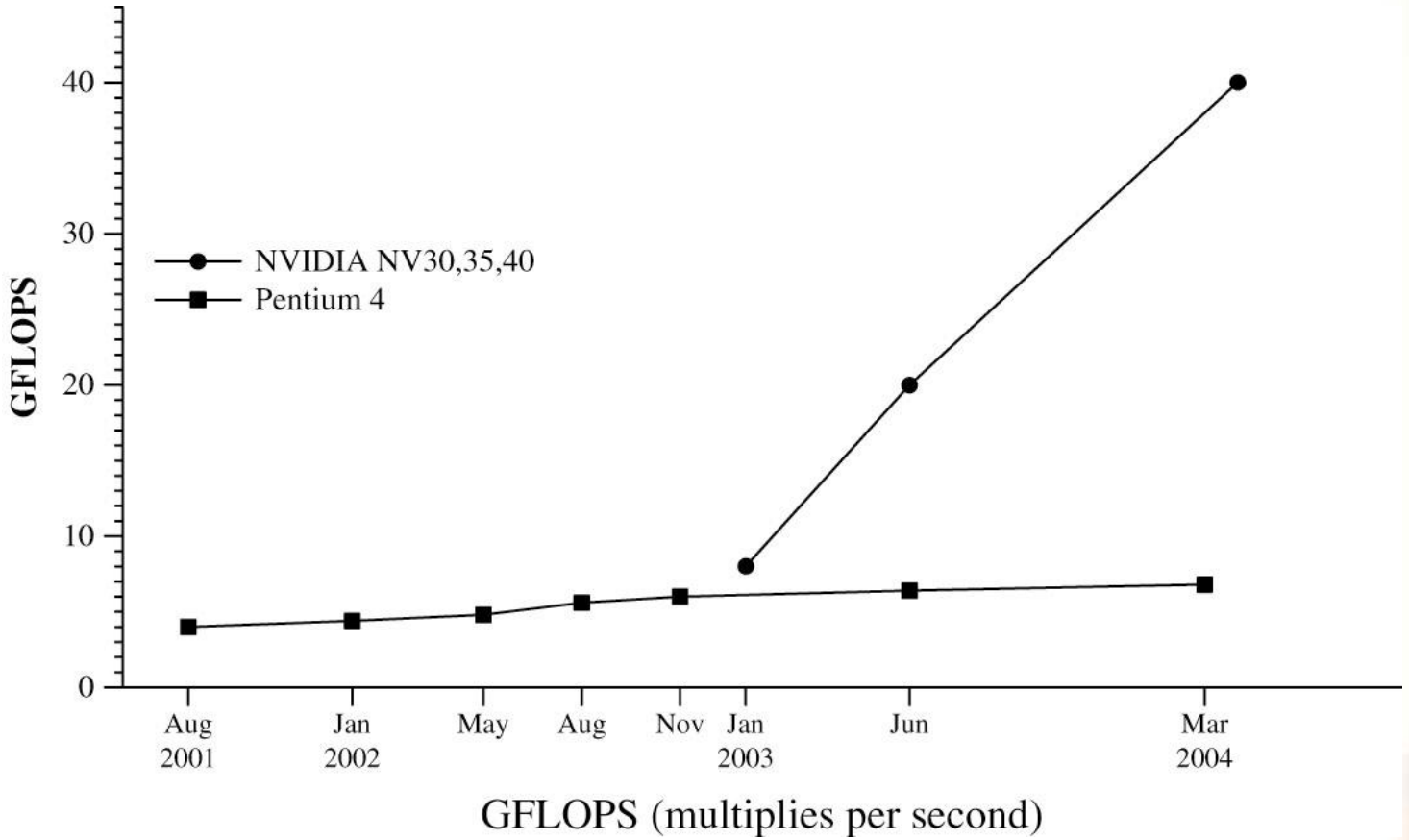
- SLI moving to mainstream:
  - GeForce 6600 GT SLI
  - In addition to 6800 GT and 6800 Ultra
- Dual-GPU boards
  - Gigabyte 3D1
    - Dual 6600 GT

- SLI motherboards sold to date: > 350,000

Game Development Cycle

- 2 years (or more)
  - CPU performance doubles (or less)
  - GPU performance quadruples
- CPU/GPU balance shifts!
  - Worse: CPU-hungry modules come later: AI, physics, full game play
- SLI hints at future GPU vs. CPU balance
  - For target ‘mainstream’ spec

GPU vs. CPU Performance History



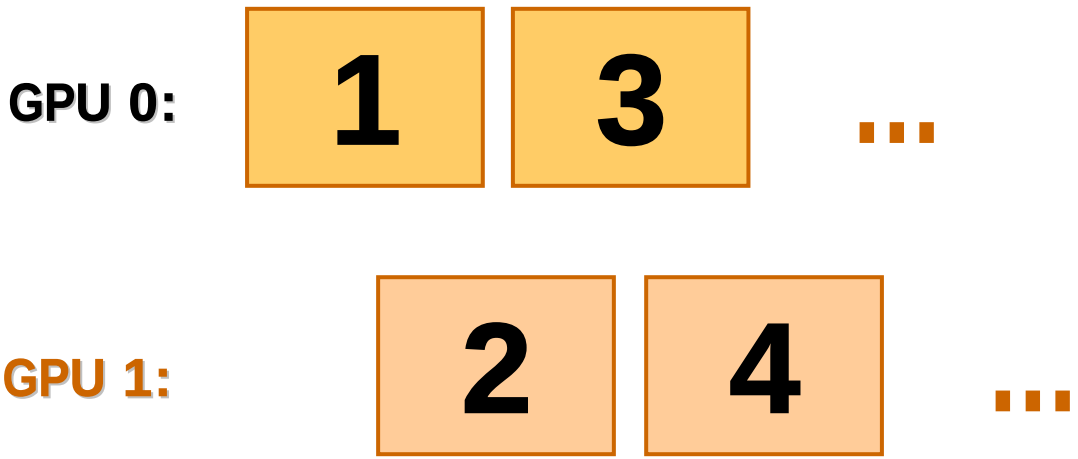
Courtesy Ian Buck, Stanford University

SLI Rendering Modes

- **Compatibility mode:**
  - **Only uses one GPU**
  - **No SLI benefits**
- **Alternate frame rendering (AFR)**
- **Split frame rendering (SFR)**

AFR

- Each GPU works on its own frame



- Scan-out toggles where to read framebuffer from

General Rendering Case for AFR

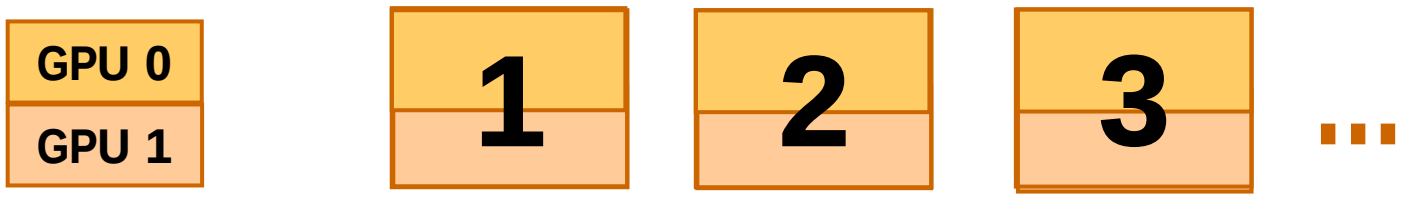
- If frame not self-contained:
  - Push necessary data to other GPU
  - E.g., updating render-to-texture targets only every other frame
- Pushing data to other GPU is overhead
  - Hence not 2x speed-up

AFR Advantages

- All work is parallelized
  - Pixel fill, raster, vertex transform
- Preferred SLI mode
- Works best when frame self-contained
  - No prior work is re-used
  - No communications overhead between GPUs

SFR

- Both GPUs work on the same frame
  - GPU 0 renders top portion
  - GPU 1 renders bottom portion



- Scan-out combines framebuffer data

General Rendering Case for SFR

- **Load-balance ‘top’ vs. ‘bottom’**
  - If one GPU took longer to render
  - Adjust load accordingly (make it work less)
- **Clip vertices to top/bottom portions**
  - Avoids both GPUs processing all vertices
  - But not perfect
- **Still requires data sharing:**
  - E.g., render to texture

SFR Compared to AFR

- SFR works even when limiting number of frames buffered
  - Or when AFR otherwise fails
- In general, SFR has more communications overhead
- Applications with heavy vertex load benefit less from SFR

Overview: Things Interfering with SLI

- CPU-bound applications
  - Or vertical-sync enabled
- Limiting number of frames buffered
- Communications overhead

CPU-Bound Applications

- SLI cannot help
- Reduce CPU work or better:
- Move CPU work onto the GPU
  - See http://GPGPU.org
- Don’t throttle frame-rate

V-Sync

- Enabling v-sync limits frame rate to multiples of the monitor refresh rate

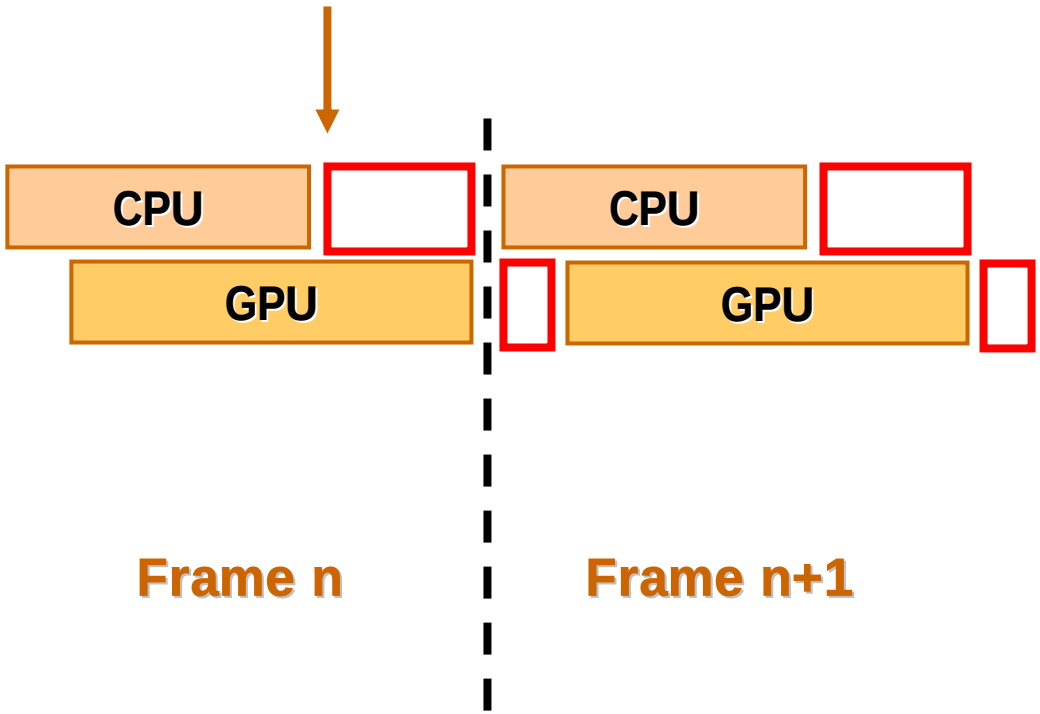
Limiting Number of Frames Buffered

- Some apps allow at most one frame buffered
  - To reduce lag
  - Via occlusion queries
  - Don't read back-buffer: causes CPU stall!
- Breaks AFR SLI
- But SLI is up to ~1.9x faster
  - I.e., SLI systems ~1.9x less lag

Why Reading the Back Buffer Is Bad

Back buffer read:

wait for GPU to finish rendering



OpenGL and SLI

- Limit OpenGL rendering to a single window
  - child windows shouldn't have OpenGL contexts
- Request pixel format with PDF_SWAP_EXCHANGE
  - tells driver that app doesn't need the back buffer contents after SwapBuffers()
- Avoid rendering to FRONT buffer
  - use overlays instead on Quadro GPUs

Offscreen Rendering and Textures

- **Limit pbuffer usage**
  - Often requires broadcasting rendering to both GPUs
- **Use render-to-texture rather than glCopyTexSubImage**
  - glCopyTexSubImage requires texture to be copied to both GPUs
  - Use FBO or pbuffers instead
- **Limit texture working set**
  - Textures have to be stored on both GPUs
  - Don’t download new textures unnecessarily

Geometry

- **Use Vertex Buffer Objects or display lists to render geometry**
  - **Don’t use immediate mode**
  - **Reduces CPU overhead**
- **Render the entire frame**
  - **Don’t use use glViewport or glScissor**
  - **Disables load balancing in SFR mode, and hurts performance in AFR mode**

More OpenGL SLI Tips

- **Limit read-backs**
  - **e.g. glReadPixel, glCopyPixels**
  - **causes pipeline to stall**
- **Never call glFinish()**
  - **doesn't return until all rendering is finished**
  - **prevents parallelism**

Using Multiple GPUs Without SLI

- Some applications want to handle distributing load between multiple GPUs themselves
  - e.g. do GPGPU processing on one, rendering on the other
- Switch off SLI mode
  - each GPU appears as a separate device
- Add details here!

How Do I Detect SLI Systems?

- **NVCpl API:**
  - NVIDIA-specific API supported by all NV drivers
- **Function support for:**
  - Detecting that NVCpl API is available
  - Bus mode (PCI/AGP/PCI-E) and rate (1x-8x)
  - Video RAM size
  - SLI

NVCpl API SLI Detection

- SDK sample and full documentation available

```
HINSTANCE hLib = ::LoadLibrary("NVCPL.dll");

NvCplGetDataIntType NvCplGetDataInt;
NvCplGetDataInt =
  (NvCplGetDataIntType)::GetProcAddress(hLib,
                        "NvCplGetDataInt");

long    numSLIGPUs  = 0L;
NvCplGetDataInt(NVCPL_API_NUMBER_OF_SLI_GPUS,
                                &numSLIGPUs);
```

Forcing SLI Support In Your Game

- **Use NVCpl**
  - **NvCplSetDataInt() sets AFR, SFR, Compatibility mode**
  - **See SDK sample**
- **Modify or create a profile:**
  - **http://nzone.com/object/nzone_sli_appprofile.html**
  - **End-users can create profile as well**

How Do I Detect SLI Systems?

- **NVCpl API:**
  - NVIDIA-specific API supported by all NV drivers
- **Function support for:**
  - Detecting that NVCpl API is available
  - Bus mode (PCI/AGP/PCI-E) and rate (1x-8x)
  - Video RAM size
  - SLI

NVCpl API SLI Detection

- SDK sample and full documentation available

```
HINSTANCE hLib = ::LoadLibrary("NVCPL.dll");

NvCplGetDataIntType NvCplGetDataInt;
NvCplGetDataInt =
  (NvCplGetDataIntType)::GetProcAddress(hLib,
                        "NvCplGetDataInt");

long    numSLIGPUs  = 0L;
NvCplGetDataInt(NVCPL_API_NUMBER_OF_SLI_GPUS,
                                &numSLIGPUs);
```

Forcing SLI Support In Your Game

- **Use NVCpl**
  - **NvCplSetDataInt() sets AFR, SFR, Compatibility mode**
  - **See SDK sample**
- **Modify or create a profile:**
  - **http://nzone.com/object/nzone_sli_appprofile.html**
  - **End-users can create profile as well**

SLI Performance Debug Support

- SLI support in NVPerfKit:
  - Pluggable hardware and driver signals for
  - PIX
  - perfmon.exe
  - pdh (your game, VTune…)

- "NVIDIA Performance Analysis Tools" Today, 2:30pm-3:30pm

Supported SLI Performance Signals

- Total SLI peer-to-peer bytes
- Total SLI peer-to-peer transactions

- Above originating from
  - Vertex/index buffers: bytes and transactions
  - Textures: bytes and transactions
  - Render targets: bytes and transactions

Questions?

- GPU Programming Guide, Chapter 8
  http://developer.nvidia.com/object/gpu_programming_guide.html

- http://developer.nvidia.com
  The Source for GPU Programming

- Thanks: Matthias Wloka, Jason Allen