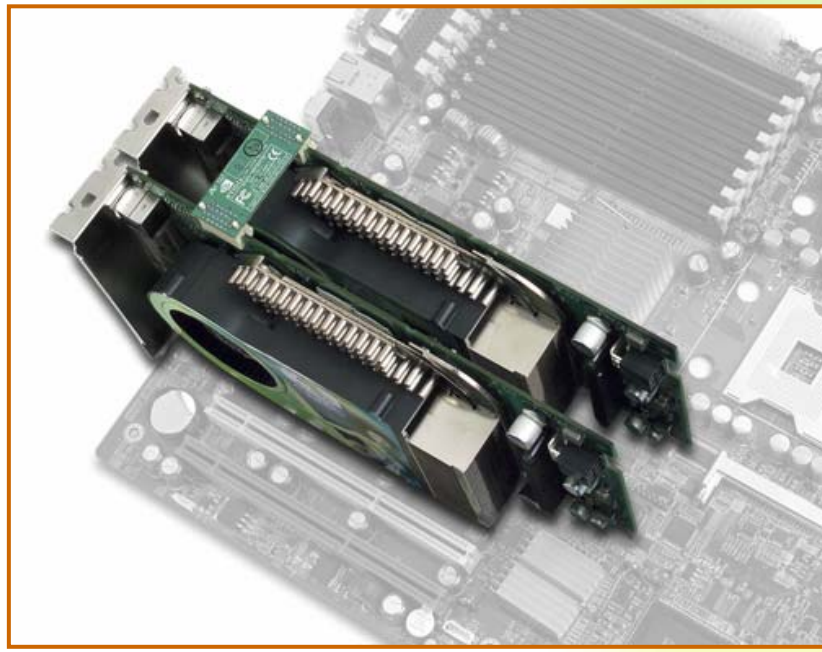
NVIDIA

Matthias M Wloka

NVIDIA Corporation

SLI: Scalable Link Interface

- Plug 2 identical GPUs into PCI-E motherboard
- Driver still reports only one (logical) device
  - Renders up to 1.9x faster
- Video memory does NOT double



Don't Care For High-End Niche Markets

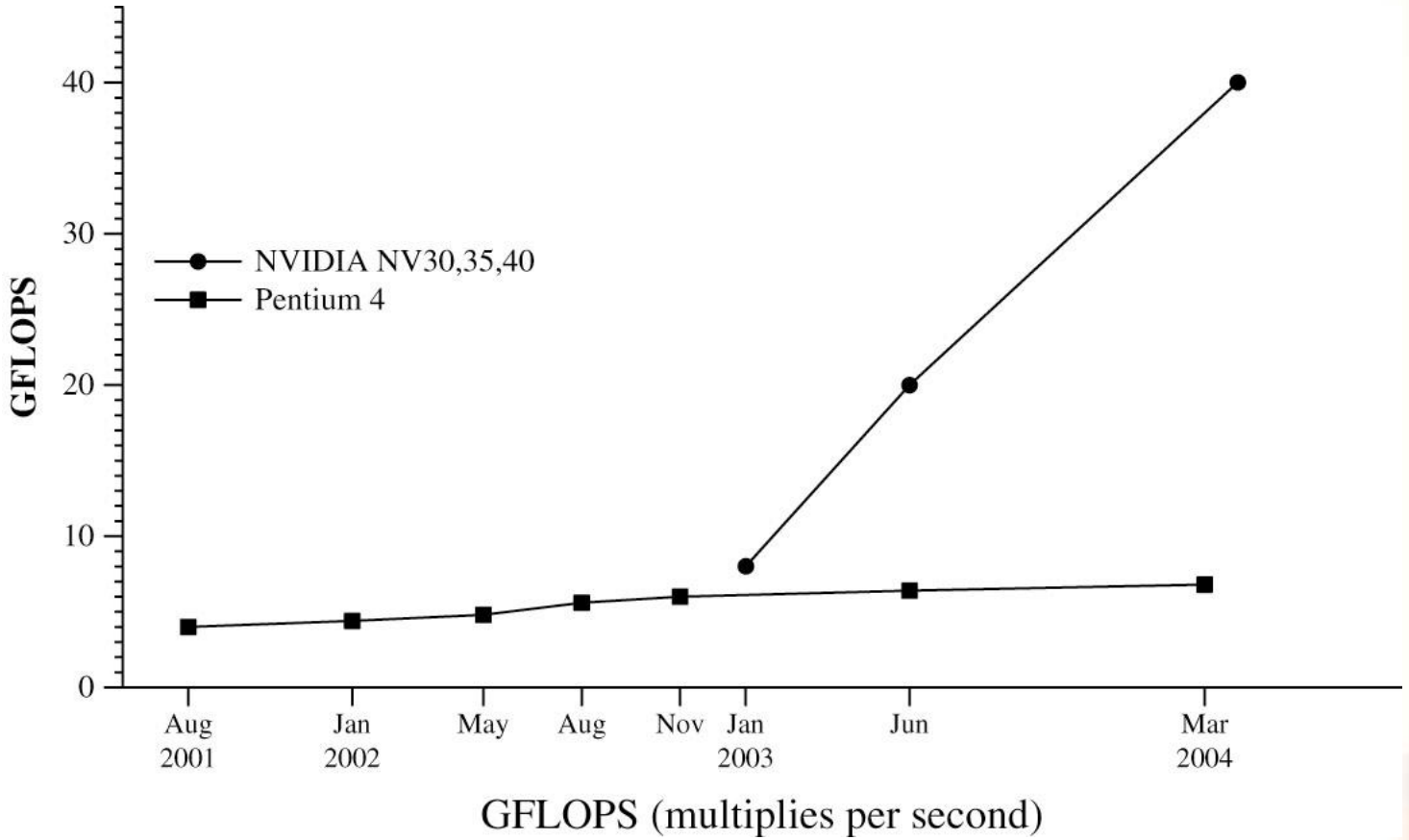
- SLI becoming mainstream:
  - GeForce 6600 GT SLI
  - In addition to 6800 GT and 6800 Ultra
- Dual core boards
  - Gigabyte 3D1: Dual 6600 GT

- SLI motherboards sold to date: > 350,000 units
  - That's > 25% of total nForce 4

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

The Last Couple of Years



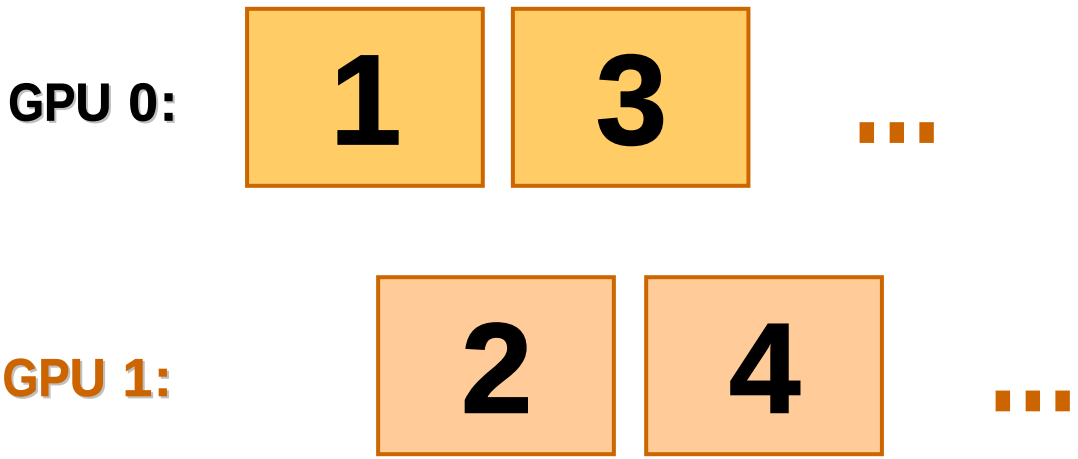
Courtesy Ian Buck, Stanford University

Ok, How Does SLI Work?

- **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

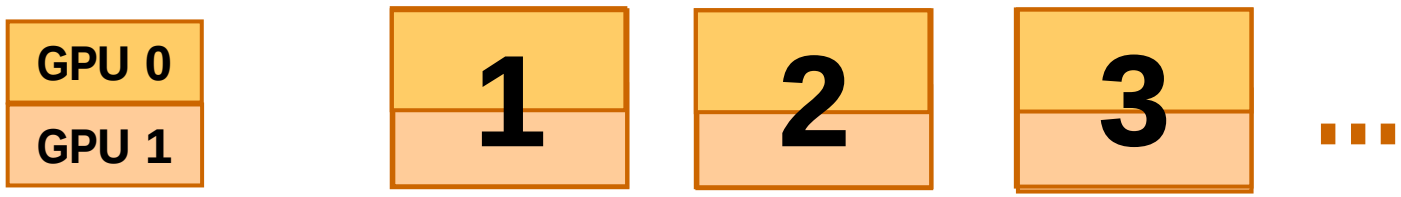
- If frame not self-contained:
  - Push necessary data to other GPU
  - E.g., updating render-to-texture targets only every other frame
- Pushing data to other GPU is overhead
  - Hence not 2x speed-up

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

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 profiles as well**

Overview: Things Interfering with SLI

- CPU-bound applications
  - Or vsync 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

VSync Enabled

- Throttles frame-rate to monitor refresh
- Enabling triple-buffering does NOT offset enabling vsync:
  - If render-rate faster than monitor refresh,
  - Then vsync still gates GPU
- Worse, triple-buffering
  - Increases lag
  - Consumes (much) more video-memory

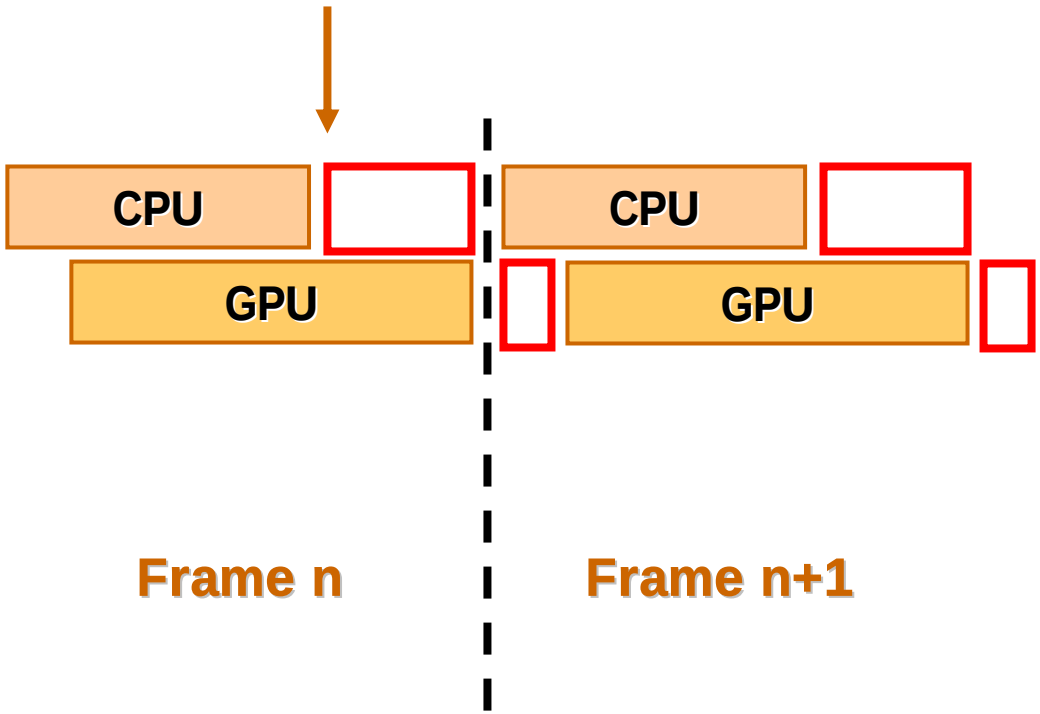
Limiting Number of Frames Buffered

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

Why Locking the Back-Buffer Is Bad

Back-buffer lock:

wait for GPU to finish rendering



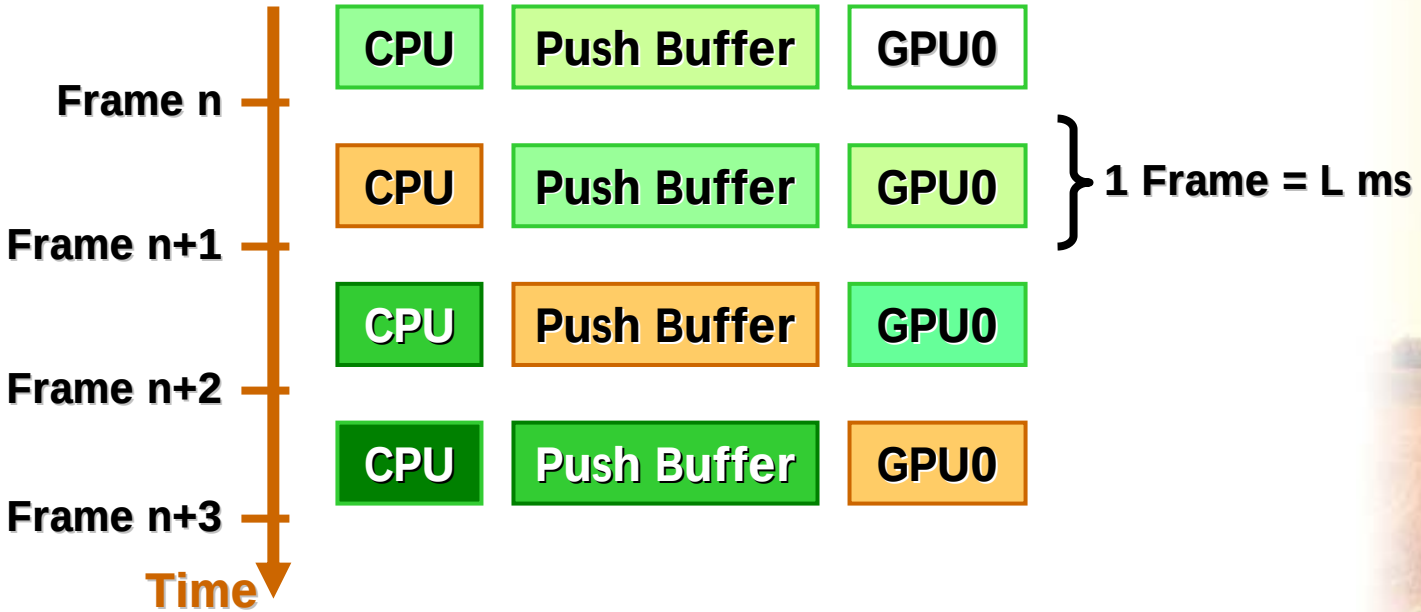
Limit Frames Buffered to Number of GPUs

- **Single GPU system:**
  Buffer at most 1 frame

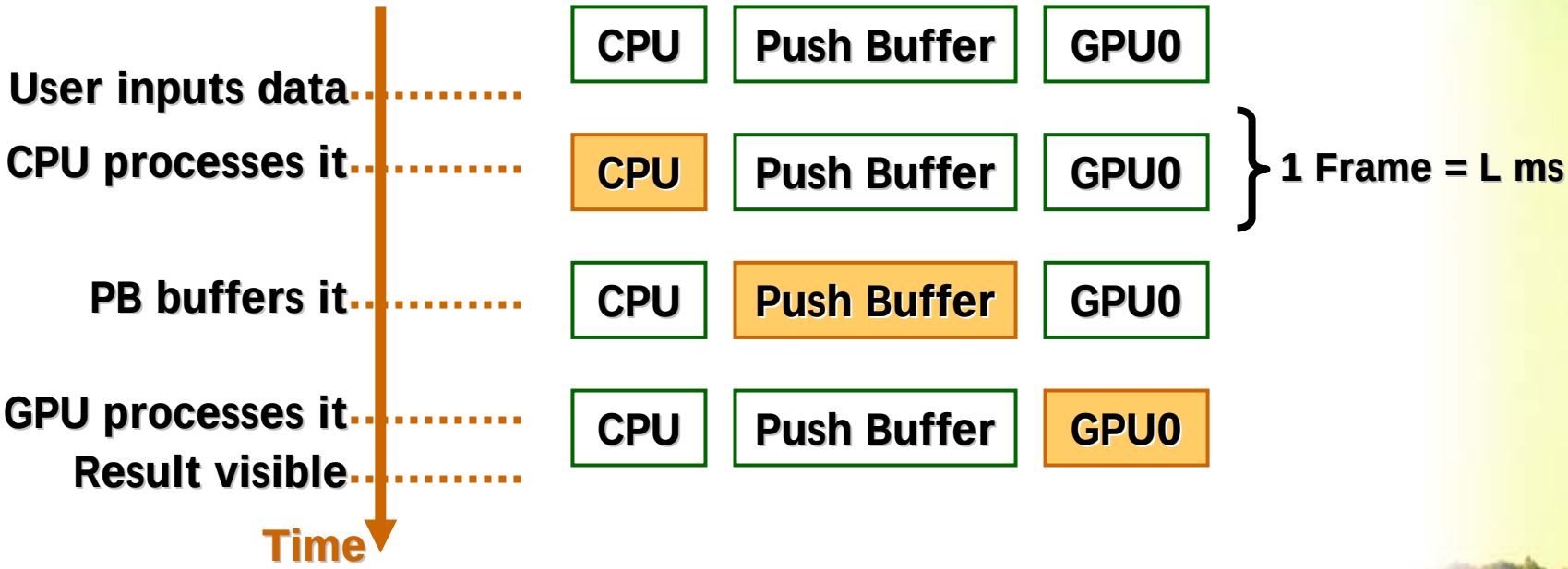
- **When detecting SLI system:**
  Buffer at most 2 frame

The Basic Pipeline

Frames flow through pipe over time:



Single GPU Latency

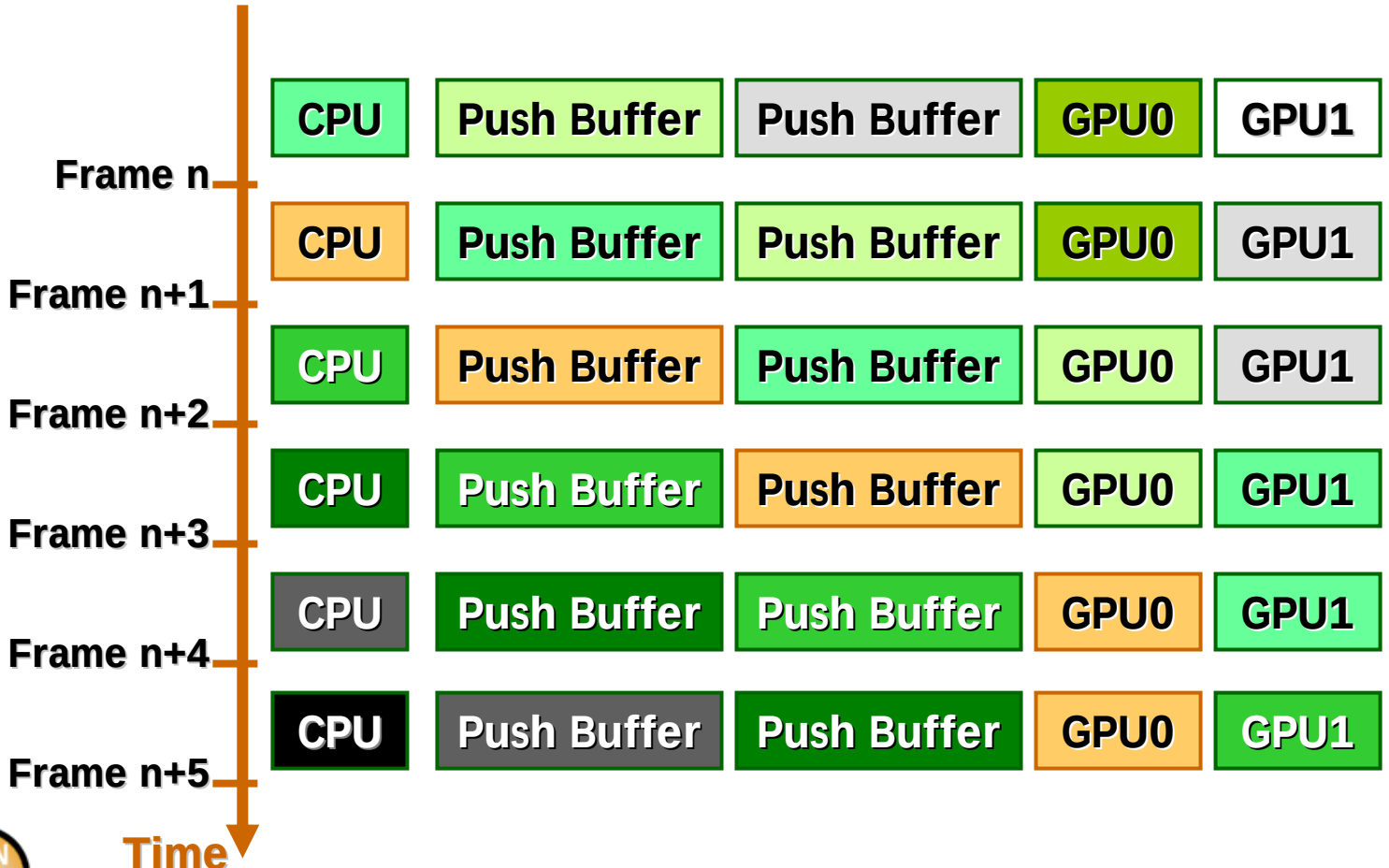


Total latency: 3L ms

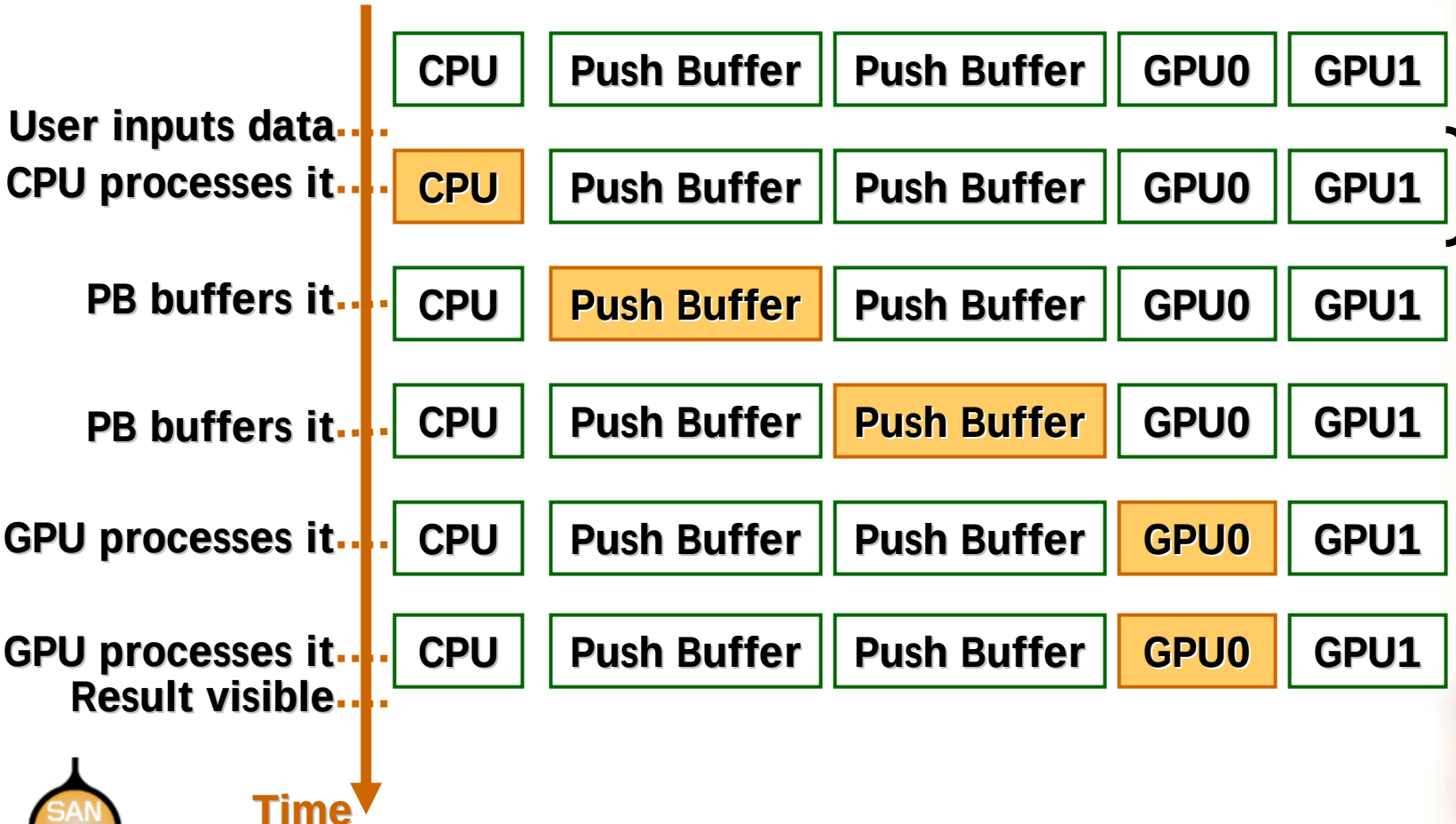
Latency Assumptions

- GPU limited
  - If not, then push buffer contains <1 frame
  - No point in limiting push buffer
- SLI is 2x faster
  - Can relax this later!
- Increase frames buffered to 2:

Frames Flowing Through AFR SLI



AFR SLI Latency



Total latency: 5·L/2 ms

Latency Comparison: Single vs. AFR

- **Single GPU latency: 3L ms**
  - 3 frames of length L ms
- **AFR SLI GPU latency: 5 L/2 = 2.5L ms!**
  - 5 frames of length L/2 ms (i.e., double frame rate)
  - Despite buffering twice as many frames
- **SLI speed-up only needs to be 1.66!**
  - $3L = 5L/x \rightarrow x = 5L/3L = 1.66$
  - Most games speed-up by ~1.8

SFR Latency?

- SFR unaffected by buffering one frame
- SFR speed-up directly reduces lag
  - If SFR 2x faster,
  - Then latency 2x shorter

Even Better: Limit Lag Based on FPS

- If your game runs at over 100 fps
  - Reasonable to buffer 3 frames
- If your game runs at less than 15 fps
  - Only allow one frame to buffer
- Faster SLI system gets automatic benefit
- Our drivers already do that
  - > 15 fps buffer 3 frames as usual
  - < 15fps reduce number of frames buffered

Overview: Things Interfering with SLI

- CPU-bound applications
  - Or vsync enabled
- Limiting number of frames buffered
- **Communications overhead**

Communications Overhead

- Peer to peer SLI memory transfers
  - Transfer itself costs bandwidth and time
  - GPU stalls waiting for transfer to complete
- Or replicate operations on both GPUs
  - For example, render to texture
- Relevant resources:
  - Vertex/index buffers
  - Textures
  - Render targets

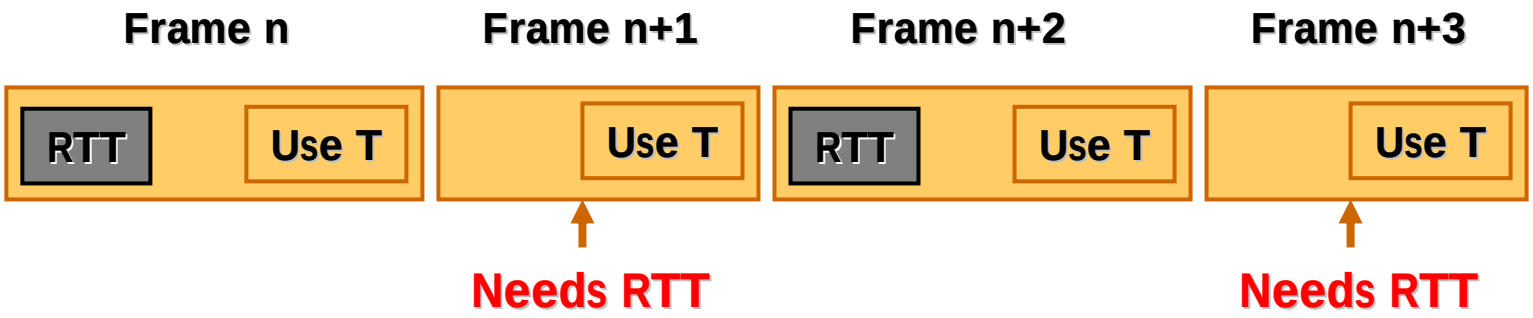
Uploading Resources On the Fly

- Remember video RAM is duplicated
- Need to transfer to both video RAMs
- Not much developers can do to avoid this
  - Oh well

Render Targets

- Clear Z
  - Always clear Z!
- Clear color when detecting SLI
  - Tells driver that the old data is irrelevant
  - No need to transfer old data across GPUs
- Don’t reuse data across frames
  - Make frames self sufficient, i.e., independent from one another

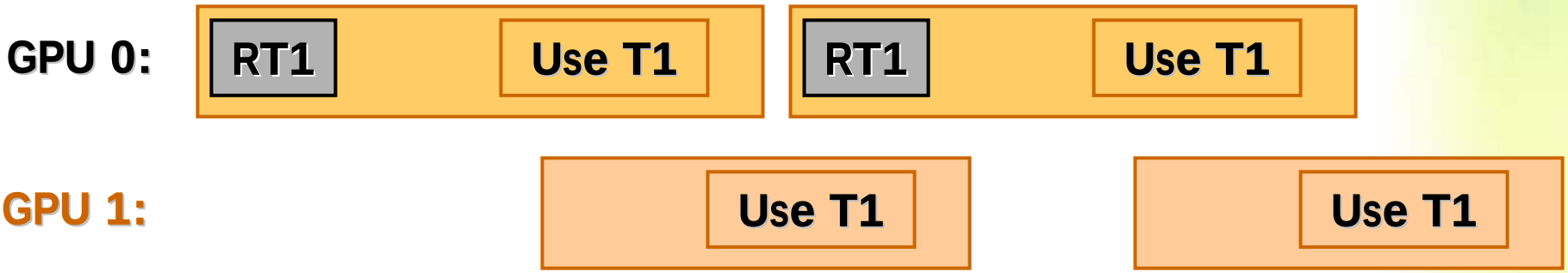
Update-Skipping "Optimization"



- **Added SLI overhead:**

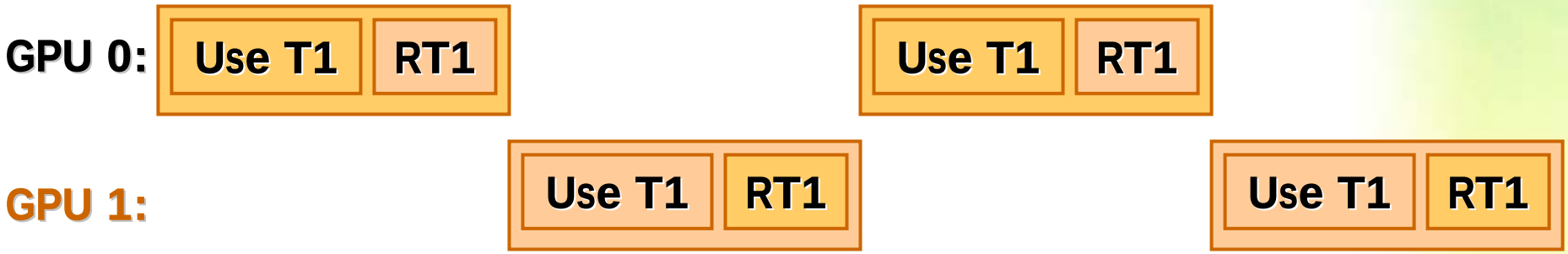
  - GPU 1 stalls until GPU 0 RTT finishes and transfers
  - Or GPU 1 duplicates RTT operation
  - Might as well do right thing when on SLI

Render Early, Use Late!

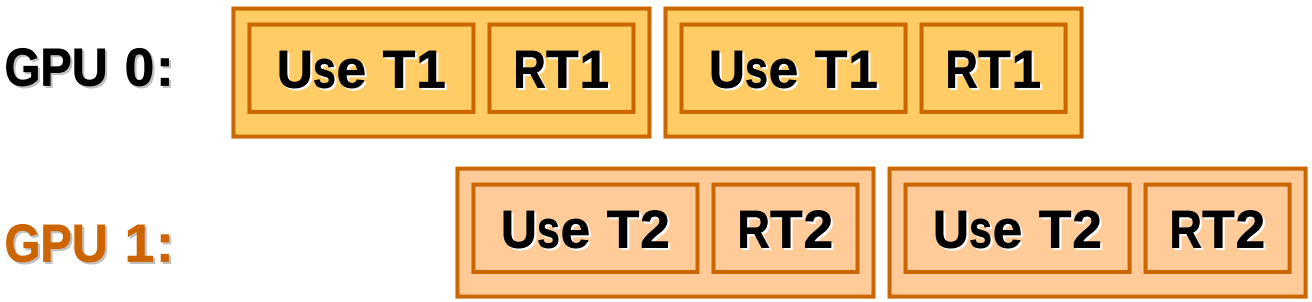


- Avoid sync-stalls
  - In AFR SLI as shown
  - And in single GPU mode
  - But still has communications overhead

Really Bad: Use Early, Render Late



Instead: Ring-buffer textures when on SLI!



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

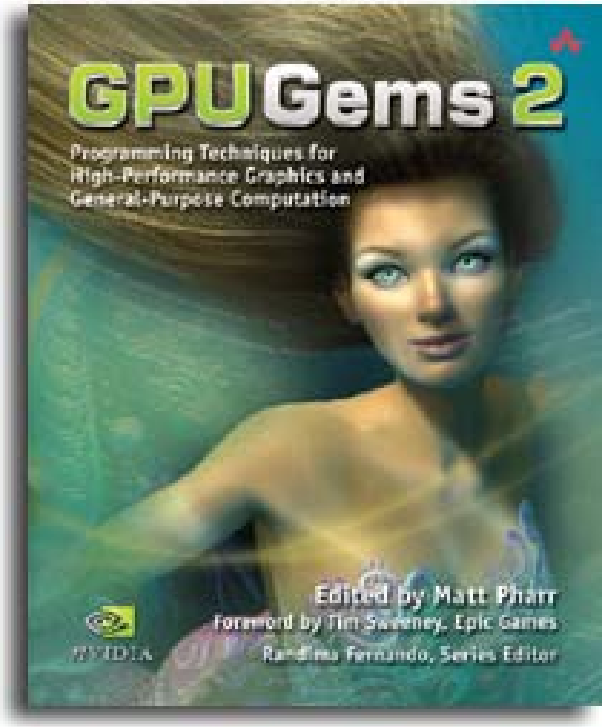
- Above originating from
  - Vertex/index buffers: bytes and transactions
  - Textures: bytes and transactions
  - Render targets: bytes and transactions

Questions?

- GPU Programming Guide, Chapter 8
  http://developer.nvidia.com/object/gpu_programming_guide.html
- http://developer.nvidia.com
  The Source for GPU Programming
- mwloka@nvidia.com
- Slides available online

GPU Gems 2

Programming Techniques for High-Performance Graphics and General-Purpose Computation



- **880 full-color pages, 330 figures, hard cover**
- **$59.99**
- **Experts from universities and industry**

"The topics covered in *GPU Gems 2* are critical to the next generation of game engines."

— Gary McTaggart, Software Engineer at Valve, Creators of Half-Life and Counter-Strike

"*GPU Gems 2* isn't meant to simply adorn your bookshelf—it's required reading for anyone trying to keep pace with the rapid evolution of programmable graphics. If you're serious about graphics, this book will take you to the edge of what the GPU can do."

—Rémi Arnaud, Graphics Architect at Sony Computer Entertainment