GPU TECHNOLOGY CONFERENCE

April 4-7, 2016 | Silicon Valley

GET TO KNOW THE NVIDIA GRID™ SDK

Shounak Deshpande, NVIDIA

PRESENTED BY NVIDIA.

AGENDA

Background

NVIDIA GRID SDK

Measuring Performance

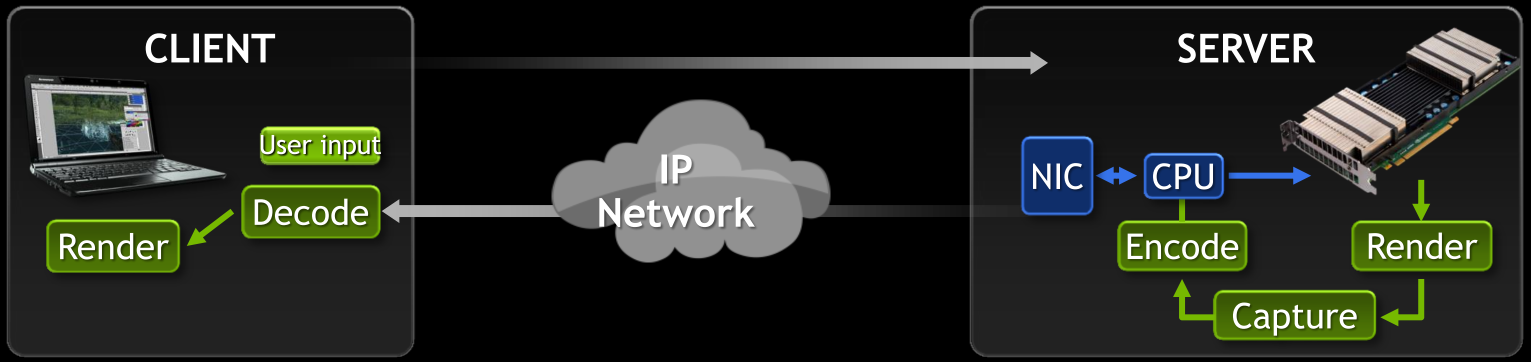
Maximizing Performance

Interactive Question-Answer Session

CLOUD\REMOTE GRAPHICS

- VDI Enterprise, Remote Workstation
  - VMWare, CITRIX, Dassault, and more
- Game streaming
  - GeForceNow
  - Windows DirectX / OpenGL
  - Linux OpenGL

REMOTE GRAPHICS ECOSYSTEM



Client | Network | Remote Graphics Server

GRID SW AND HW STACK COMPONENTS

Streaming

Capture (Pixel grabbing)
HW Accelerated video compression
HW Accelerated video decoding

Virtualization

Graphics Shim layers (app streaming)
Platform Virtualization (VDI)
Hypervisors (VDI)
Full Virtualization (VDI)

HW Platforms

Server

AWS G2 Instance
GRID K520, M30 GPU
Tesla M60 GPU
NVIDIA Quadro GPUs

Client

Anything

NVIDIA GRID SDK

NVIDIA CAPTURE SDK

(Formerly known as NVIDIA GRID SDK)

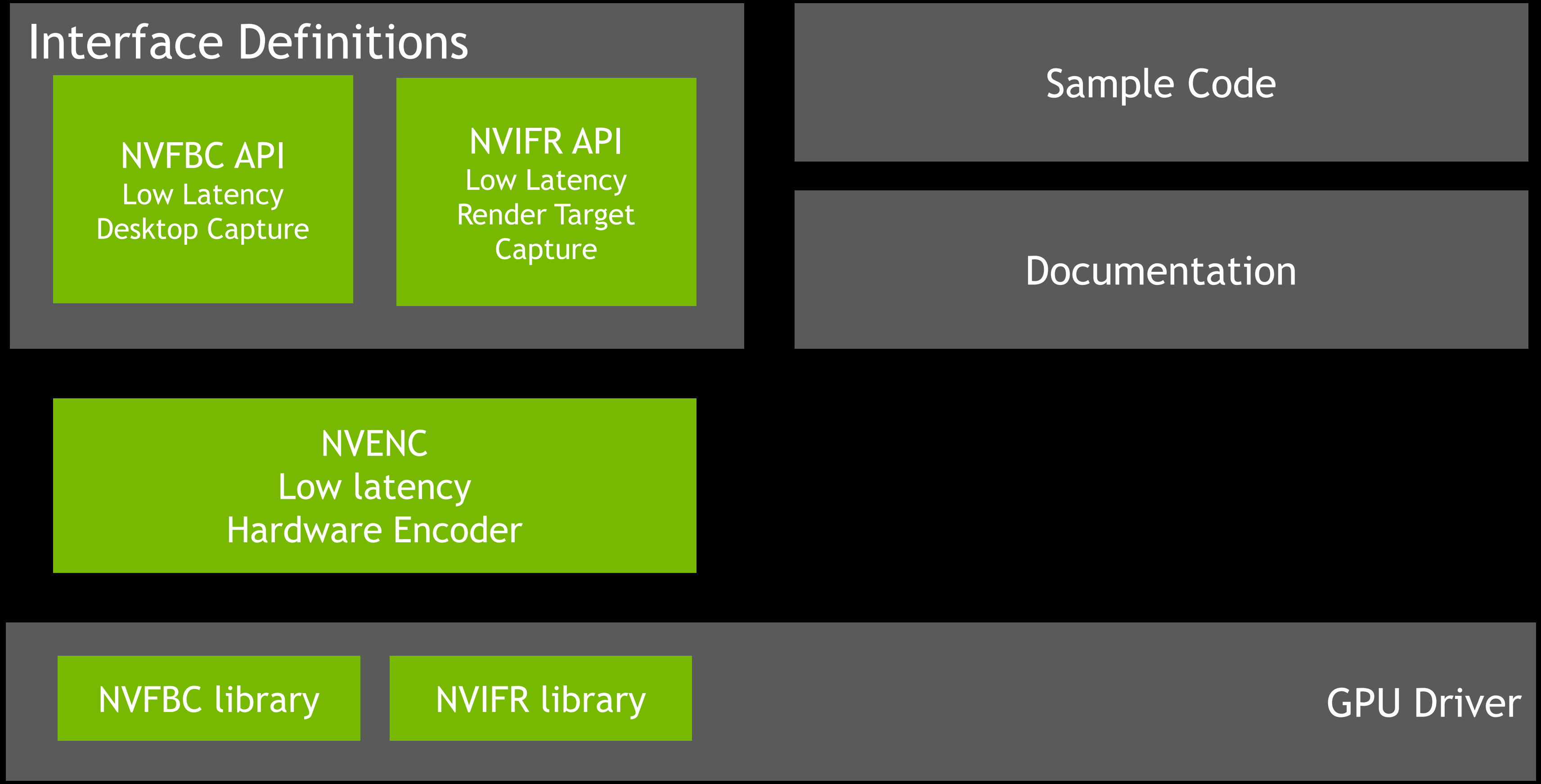
Goal: Enable Low Latency Remote Graphics Solutions by harnessing NVIDIA GPUs

OS: Windows 7+, Linux (CentOS, Debian, RedHat, more)

Download: https://developer.nvidia.com/grid-app-game-streaming

Support: GRID-devtech-support@nvidia.com

NVIDIA CAPTURE SDK COMPONENTS



GPU TECHNOLOGY CONFERENCE

NVIDIA CAPTURE SDK: THE “CAPTURE” PART

NVFBC	NVIFR
Brute force, capture all on screen	No-frills RenderTarget capture
Orthogonal to Graphics APIs Easy to integrate with NVENC API	Supports Directx9,10,11, OpenGL APIs
Easy onboarding, no process injection	Easy to integrate with NVENC API
Efficient than GDI-based screen scraping	Needs to be injected in target process
One session per display	One session per target window
	Enables higher density of streamed apps

NVIDIA CAPTURE SDK : INTERFACES

	NVFBC: NVIDIA Frame Buffer Capture	NVIFR: NVIDIA In-band Frame Render	
Windows	NVFBC NVFBCToSys NVFBCCuda NVFBCToHWEnc NVFBCToDX9Vid	NVIFR - Directx NVIFRToSys NVIFRToHWEnc	NVIFR - Directx NVIFRToSys NVIFRToHWEnc
Linux	NVFBC NVFBCToCuda NVFBCToSys NVFBCToHWEnc	NVIFR - OpenGL NVIFRToSys NVIFRToHWEnc	

-ToHWEnc interfaces internally invoke NVENC API (part of NVIDIA Video Codec SDK)

EVOLUTION OF NVIDIA CAPTURE SDK

	Legacy	2014	2015	2015	2016
SDK		2.3	3.0	4.0	5.0
Windows	• GRID K340, K520, K1, K2, Quadro 4000+ support • H.264 encode support • Windows 7, 8, 8.1 support	• GRID M30 limited support • Maxwell NVENC enhancements - quarter-res first pass; lossless encoding; 4:4:4 encoding	• GRID M30 full support • NVENC RC 2.0	• HEVC support • Tesla M60 support • New unified codec-agnostic interface for HW encoder • Driver support for H.264 YUV 4:4:4 NVIFR capture+encode for DX10/DX11 applications	• Enable NVFBC without driver reload • Windows 10 support • New NVFBC interface to capture desktop to DirectX 9 video memory surface, along with diffmap support • Timeout API for NVFBC blocking mode capture • Separate thread Mouse capture for all NVFBC interfaces • Propagate frame timestamp through NvIFRHWEncode
Linux	• GRID K340, K520, K1, K2, Quadro 4000+ support • H.264 encode support		• GRID M30 full support • NvIFR full parity for NVENC features with Windows • NVENC RC 2.0	• HEVC support • Tesla M60 support • New unified codec-agnostic interface for HW encoder	
HW	GRID K340, K520, K1, K2, Quadro K2000+ (Legacy–2016)	GRID M30, Quadro M6000 (2014–2016)			Tesla M60 (2016)

USING NVFBC API

USING NVFBC FOR DESKTOP CAPTURE

Enable NVFBC

Create NVFBC capture session object

Setup NVFBC capture session object

Capture

Release NVFBC capture session object

CAPTURING A SCREENSHOT WITH NVFBC

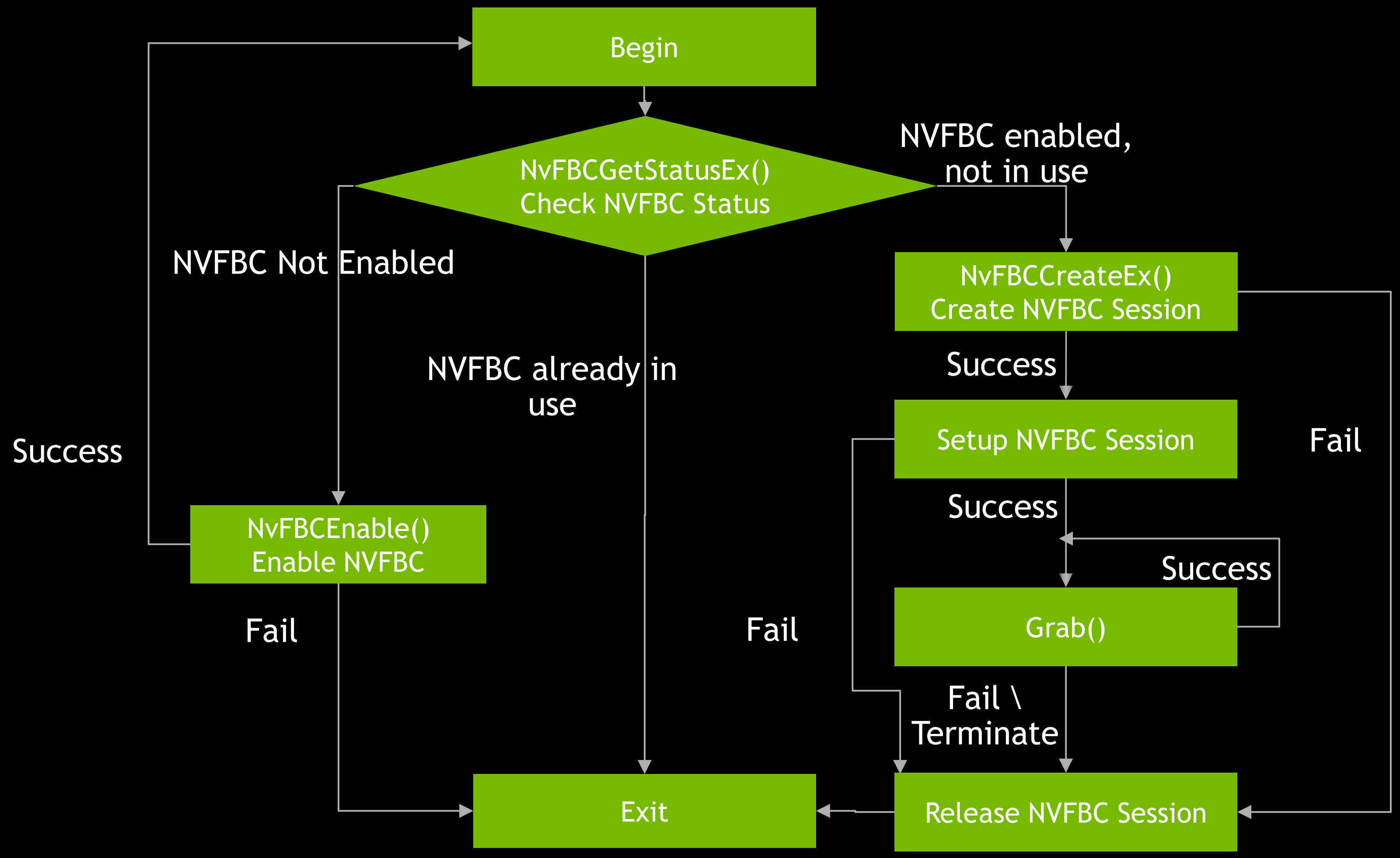
```
NvFBCToSys          *toSys;
NvFBCFrameGrabInfo grabInfo;
NVFBCRESULT fbcRes = NVFBC_SUCCESS;

//! output buffer
void                *frameBuffer = VirtualAlloc(framesz);

NvFBCCreateParams createParams;
memset(&createParams, 0, sizeof(createParams));
createParams.dwVersion = NVFBC_CREATE_PARAMS_VER;
createParams.dwInterfaceType = NVFBC_TO_SYS;

//! Create an instance of NvFBCToSys
fbcRes = NvFBCCreateEx(&createParams);
toSys = (NvFBCToSys *)createParams.pNvFBC;

//! Setup NvFBCToSys to grab the desktop without the mouse as RGB.
NVFBC_TOSYS_SETUP_PARAMS params = {0};
params.dwVersion = NVFBC_TOSYS_SETUP_PARAMS_VER;
params.eMode = NVFBC_TOSYS_ARGB;
params.ppBuffer = &frameBuffer;

fbcRes = toSys->NvFBCToSysSetUp(&params);

NVFBC_TOSYS_GRAB_FRAME_PARAMS params = {0};
params.dwVersion = NVFBC_TOSYS_GRAB_FRAME_PARAMS_VER;
params.dwFlags = NVFBC_TOSYS_NOFLAGS;
params.eGMode = NVFBC_TOSYS_SOURCEMODE_FULL;
params.pNvFBCFrameGrabInfo = &grabInfo;

fbcRes = toSys->NvFBCToSysGrabFrame(&params);
```

← Create NVFBC session object

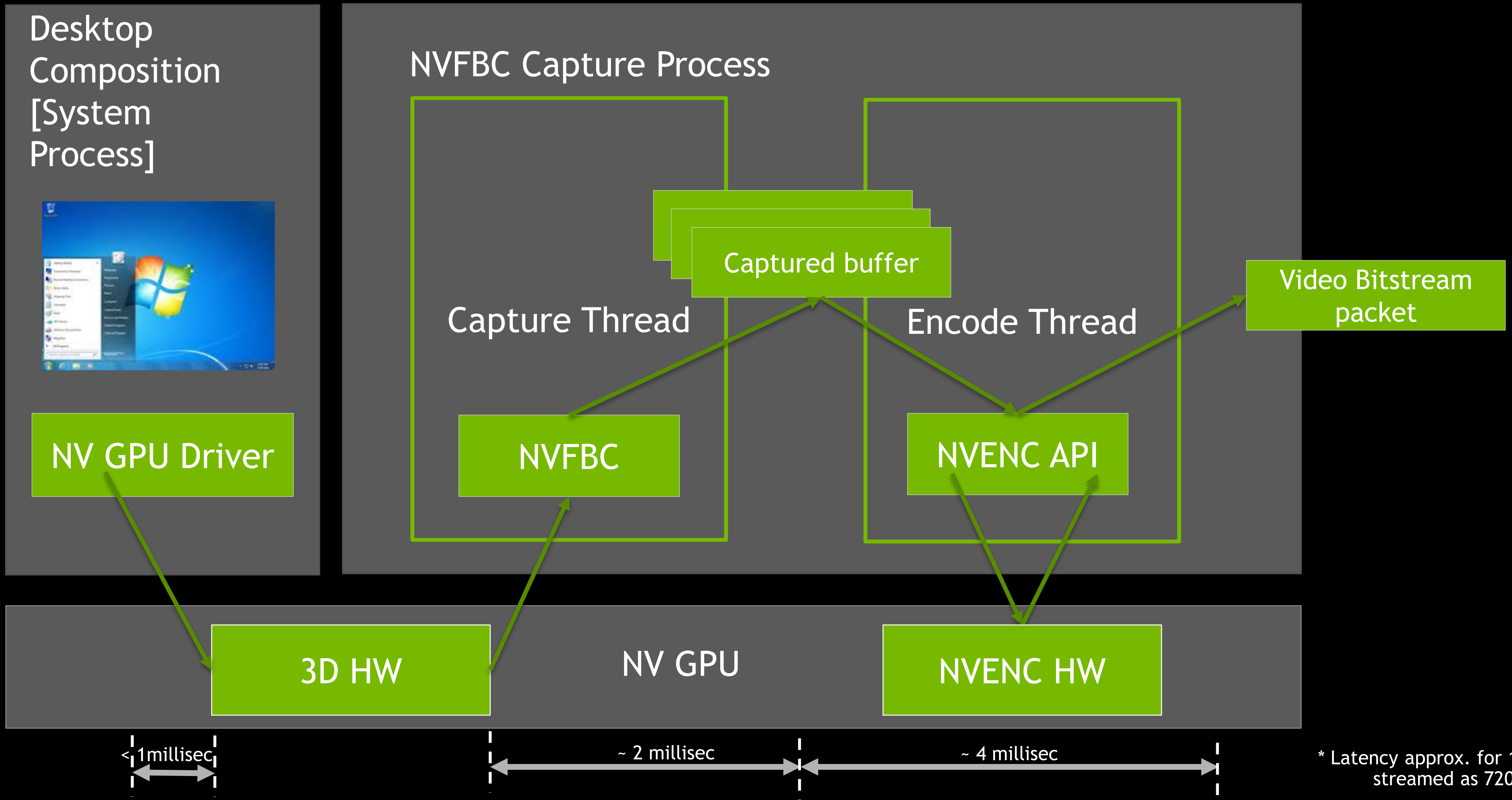
← Set up NVFBC session
"Capture" starts here

← Read Grabbed buffer

CAPTURING USING NVFBC



DESKTOP REMOTING
USING NVFBC + NVENC HW ENCODER



* Latency approx. for 1080p desktop streamed as 720p video

USING NVIFR API

USING NVIFR FOR APPLICATION STEAMING

Write a Shim layer to host NVIFR

Inject Shim layer into target application

Fetch rendering graphics context

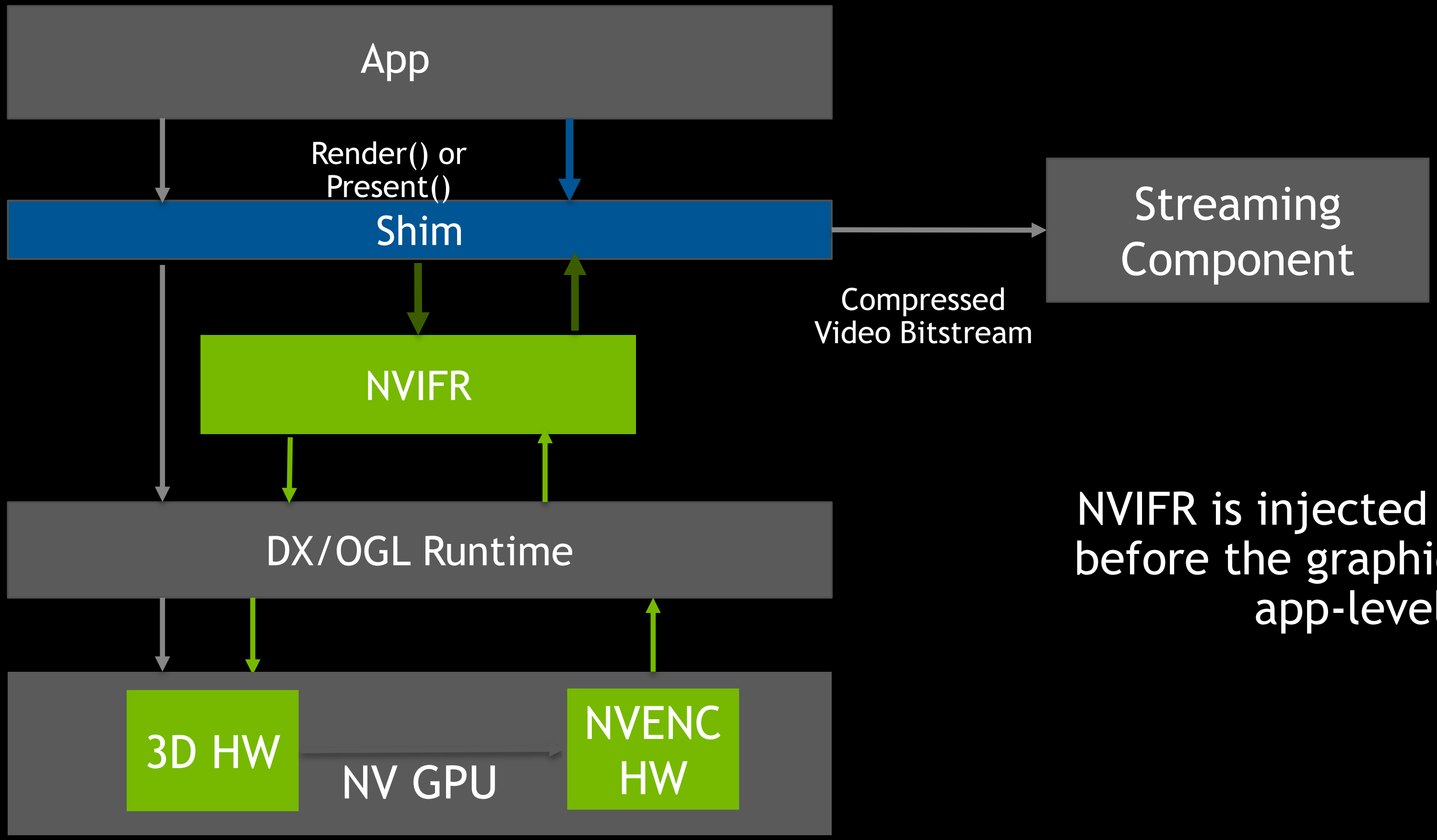
Create NVIFR session object using the context

Setup NVIFR session object

Capture

Release NVIFR session object

APP STREAMING USING HW ENCODER



NVIFR is injected into the application before the graphics runtime, using an app-level shim layer

DIRECTX APP STREAMING USING NVIFR HW ENCODER

```
NVIFR_HW_ENC_SETUP_PARAMS params = {0};
params.dwVersion = NVIFR_HW_ENC_SETUP_PARAMS_VER;
params.dwNBuffers = 1;
params.dwBSMaxSize = 2048*1024;
params.ppPageLockedBitStreamBuffers = &g_pBitStreamBuffer;
params.ppEncodeCompletionEvents = &g_EncodeCompletionEvent;

params.configParams.dwVersion = NV_HW_ENC_CONFIG_PARAMS_VER;
params.configParams.dwProfile = 100;
params.configParams.dwAvgBitRate = (DWORD)dBitRate;
params.configParams.dwFrameRateDen = 1;
params.configParams.dwFrameRateNum = 30;
params.configParams.dwPeakBitRate = (params.configParams.dwAvgBitRate * 12/10);   // +20%
params.configParams.dwGOPLength = 75;
params.configParams.dwQP = 26 ;
params.configParams.eRateControl = NV_HW_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP;
params.configParams.ePresetConfig= NV_HW_ENC_PRESET_LOW_LATENCY_HQ;
params.configParams.eCodec = g_Codec;

// Set Single frame VBV buffer size
params.configParams.dwVBVBufferSize = (params.configParams.dwAvgBitRate)/
                                      (params.configParams.dwFrameRateNum/params.configParams.dwFrameRateDen);

NVIFRRESULT res = g_pIFR->NvIFRSetUpHWEncoder(&params);
```

Application allocates output buffers and event handles

Select the rate control mode and encoder preset according to use case

DIRECTX APP STREAMING USING NVIFR HW ENCODER

```
//! Synchronously transfer the render target frame to the H.264 encoder
NVIFR_HW_ENC_TRANSFER_RT_TO_HW_ENC_PARAMS params = {0};
params.dwVersion = NVIFR_HW_ENC_TRANSFER_RT_TO_HW_ENC_PARAMS_VER;
params.encodePicParams.dwVersion = NV_HW_ENC_PIC_PARAMS_VER;
params.dwBufferIndex = 0;

NVIFRRESULT res = g_pIFR->NvIFRTransferRenderTargetToHWEncoder(&params);

if (res == NVIFR_SUCCESS)
{
    //! The wait here is just an example, in the case of disk IO,
    //! it forces the correct transfer to be done, not taking advantage of CPU/GPU concurrency
    WaitForSingleObject(g_EncodeCompletionEvent, INFINITE);
    ResetEvent(g_EncodeCompletionEvent);

    //! Get frame stats from the H.264 encoder
    NVIFR_HW_ENC_GET_BITSTREAM_PARAMS params = {0};
    params.dwVersion = NVIFR_HW_ENC_GET_BITSTREAM_PARAMS_VER;
    params.bitStreamParams.dwVersion = NV_HW_ENC_GET_BIT_STREAM_PARAMS_VER;
    params.dwBufferIndex = 0;
    NVIFRRESULT res = g_pIFR->NvIFRGetStatsFromHWEncoder(&params);

    //! Write new data to disk
    fwrite(g_pBitStreamBuffer, params.bitStreamParams.dwByteSize, 1, fileOut);
}
```

The event handles passed to NvIFRSetupHWEncoder will be signaled when NVENC has finished work submitted by NvIFRTransferRenderTargetToHWEncoder API

OPENGL APP STREAMING USING NVIFR HW ENCODER

```
if (nvIFR.initialize() == false)
{
    fprintf(stderr, "Failed to create a NvIFROGL instance.\n");
    return -1;
}
printf("NvIFROpenGL API version number: %d.%d\n\n", ENCODEAPI_MAJOR(nvIFR), ENCODEAPI_MINOR(nvIFR));
// A session is required. The session is associated with the current OpenGL context.
if (nvIFR.nvIFROGLCreateSession(&sessionHandle, NULL) != NV_IFROGL_SUCCESS)
{
    fprintf(stderr, "Failed to create a NvIFROGL session.\n");
    return -1;
}

memset(&config, 0, sizeof(config));

config.profile = (codecType == NV_IFROGL_HW_ENC_H264 ) ? 100 : 1;
config.frameRateNum = framesPerSecond;
config.frameRateDen = 1;
config.width = fboWidth;
config.height = fboHeight;
config.avgBitRate = calculateBitrate(fboWidth, fboHeight);
config.GOPLength = 75;
config.rateControl = NV_IFROGL_HW_ENC_RATE_CONTROL_CBR;
config.stereoFormat = NV_IFROGL_HW_ENC_STEREO_NONE;
config.preset = NV_IFROGL_HW_ENC_PRESET_LOW_LATENCY_HP;
config.codecType = codecType;
config.VBVBufferSize = config.avgBitRate;
config.VBVInitialDelay = config.VBVBufferSize;

if (nvIFR.nvIFROGLCreateTransferToHwEncObject(sessionHandle, &config, &transferObjectHandle) != NV_IFROGL_SUCCESS)
```

Create session

Create TransferObject

OPENGL APP STREAMING USING NVIFR HW ENCODER

```
// transfer the FBO
if (nvIFR.nvIFROGLTransferFramebufferToHwEnc(transferObjectHandle, NULL, fboID, GL_COLOR_ATTACHMENT0, GL_NONE) != NV_IFROGL_SUCCESS)
{
    fprintf(stderr, "Failed to transfer data from the framebuffer.\n");
    exit(-1);
}

// lock the transferred data
if (nvIFR.nvIFROGLLockTransferData(transferObjectHandle, &dataSize, &data) != NV_IFROGL_SUCCESS)
{
    fprintf(stderr, "Failed to lock the transferred data.\n");
    exit(-1);
}

// write to the file
if (isOutFile)
    fwrite(data, 1, dataSize, outFile);

// release the data buffer
if (nvIFR.nvIFROGLReleaseTransferData(transferObjectHandle) != NV_IFROGL_SUCCESS)
{
    fprintf(stderr, "Failed to release the transferred data.\n");
    exit(-1);
}
```

Capture + Encode

Retrieve output bitstream

Release buffers for re-use

MEASURING PERFORMANCE

MEASURING PERFORMANCE

Guidelines

Use high precision timers.

In-process performance measurement is suitable only for generating average numbers.

Measure GPU Utilization. (GPU-Z, NVIDIA SMI, etc.)

Note GPU clock values during measurement.

MEASURING PERFORMANCE

```
LONGLONG g_llBegin=0;
LONGLONG g_llPerfFrequency=0;
BOOL g_timeInitialized=FALSE;

#define QPC(Int64)  QueryPerformanceCounter((LARGE_INTEGER*)&Int64)
#define QPF(Int64)  QueryPerformanceFrequency((LARGE_INTEGER*)&Int64)


double GetFloatingDate()
{
    LONGLONG llNow;

    if (!g_timeInitialized)
    {
        QPC(g_llBegin);
        QPF(g_llPerfFrequency);
        g_timeInitialized = TRUE;
    }
    QPC(llNow);
    return(((double)(llNow-g_llBegin)/(double)g_llPerfFrequency) );
}
```

Use High Performance
Multimedia Timer
for accuracy

MEASURING PERFORMANCE

```
double dStart = GetFloatingDate();
double dNow = 0;
DWORD dwBufferIndex=0;
printf("Starting Encode\n");

for (DWORD i2=0; i2<g_dwMaxFrames; i2++)
{
    DWORD k = i2%g_dwNInputs;

    for (DWORD d=0; d<g_dwNEncoders; d++)
    {
        //printf("Input# %d encoded by Encoder# %d\n", i, d);
        WaitForSingleObject(aSideThreadData[d].m_ahCanRenderEvents[dwBufferIndex], INFINITE);
        ResetEvent(aSideThreadData[d].m_ahCanRenderEvents[dwBufferIndex]);

        IDirect3DSurface9 * pCurrentRenderTarget = NULL;
        ....
        ....
        ResetEvent(aSideThreadData[d].m_ahEncodeCompletionEvents[dwBufferIndex]);

        NVIFR_HW_ENC_TRANSFER_RT_TO_HW_ENC_PARAMS params = {0};
        params.dwVersion = NVIFR_HW_ENC_TRANSFER_RT_TO_HW_ENC_PARAMS_VER;
        params.encodePicParams.dwVersion = NV_HW_ENC_PIC_PARAMS_VER;
        params.dwBufferIndex = dwBufferIndex;
        res = g_apIFRs[d]->NvIFRTransferRenderTargetToHWEncoder(&params);
        ....
        ....
    }

    dwBufferIndex = (dwBufferIndex+1)%NUMFRAMESINFLIGHT;
}

double dEnd = GetFloatingDate();
printf("total time %f sec, FPS=%f\n", dEnd-dStart, double(g_dwMaxFrames)/(dEnd-dStart));
```

Start Measurement before capture loop

Run through capture\encode loop

Stop Measurement here

MAXIMIZING QUALITY & PERFORMANCE

MAXIMIZING QUALITY & PERFORMANCE

Goals & Challenges

Goals:

- Low latency
- Smooth playback of streamed video
- Minimum impact on target application\system performance

Challenge:

- Finding the right balance to get maximum CPU-GPU utilization without negative impact

MAXIMIZING QUALITY & PERFORMANCE

Guidelines

Know the system’s limits.

Memory management : Ensure there is no time lost for paging

Resource Utilization : GPU-intensive applications need frame rate throttling while lightweight appllications need pipelining and multithreading of capture – encode/post-process tasks

Timing : Ensure capture rate matches display rate

Impact on target : Use parallelism

MAXIMIZING QUALITY & PERFORMANCE

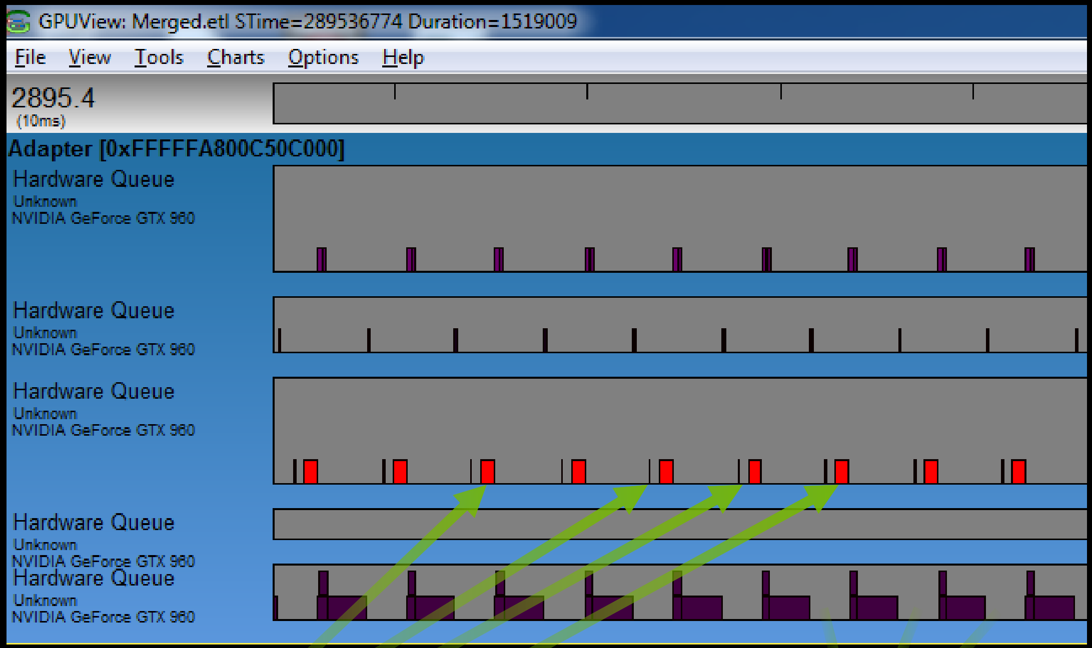
Memory management

Ensure no paging.

- Choose optimal rendering quality settings

- Choose optimal desktop or application window resolution

Loss due to paging (insufficient video memory)



Paging work

Encoder Idle

MAXIMIZING QUALITY & PERFORMANCE

Resource Utilization: Multithreading

Capture and encode/post-process should run on different threads

Constraints:

- Multiple threads must not concurrently access same DirectX context
- NVIFR Capture thread should never stall
- NVFBC Capture thread should never miss a display refresh

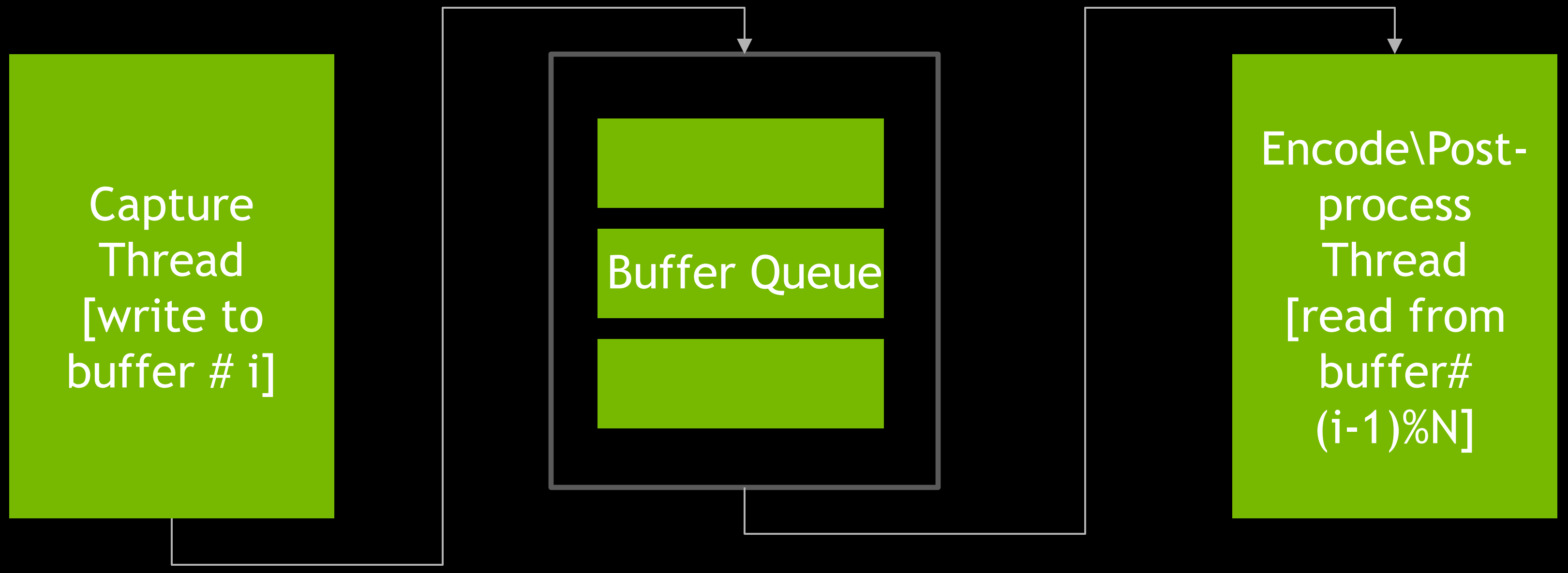
MAXIMIZING QUALITY & PERFORMANCE

Resource Utilization : Pipelining

Goal: Minimize time spent by encode thread to wait for capture to complete and vice versa

Benefit: Control on timing capture calls, less impact on application rendering performance

Triple buffering is sufficient in most cases



MAXIMIZING PERFORMANCE

Resource Utilization: Multiple Contexts with NVIFR

Why use multiple contexts?

> NVIFR capture happens in-band, shares the DirectX/OGL context used by the target application.
>
> Any GPU work scheduled by NVIFR on this context reflects as drop in rendering frame rate

Solution:

> Use shared buffers to hold captured output, for processing through a separate DirectX/OGL context running on a separate thread.

Game's D3D Context

NvIFRCopyToSharedSurface for DX9,

StretchRect to a shared surface for DX9Ex

ResourceCopyRegion to a shared surface for Dx1x

Encoder's D3D Context

NvIFRCopyFromSharedSurface for DX9,

StretchRect from a shared surface for DX9Ex

ResourceCopyRegion from a shared surface for Dx1x

Shared Surface

GPU TECHNOLOGY CONFERENCE

MAXIMIZING QUALITY & PERFORMANCE

NUMA

NUMA: Non-Uniform Memory Addressing

Create resources in the same part of the memory where the bus holding the GPU is located, reduces contention for bus bandwidth.

MAXIMIZING QUALITY & PERFORMANCE

QoS

Network bandwidth control :
NV_HW_ENC_PARAMS_RC_2PASS_FRAMESIZE_CAP

Recovering from packet loss :
Reference frame invalidation
NV_HW_ENC_PIC_PARAMS::bInvalidateReferenceFrames
NV_HW_ENC_PIC_PARAMS::ulInvalidFrameTimeStamps[]

Avoiding insertion of IDR frames :
Intra-Refresh
NV_HW_ENC_PIC_PARAMS::bStartIntraRefresh
NV_HW_ENC_PIC_PARAMS::dwIntraRefreshCnt

Dynamic bitrate change :
NV_HW_ENC_PIC_PARAMS::bDynamicBitRate
NV_HW_ENC_PIC_PARAMS::dwNewAvgBitrate,dwNewPeakBitRate,dwNewVBVBufferSize,dwNewVBVInitialDelay

COMPATIBILITY

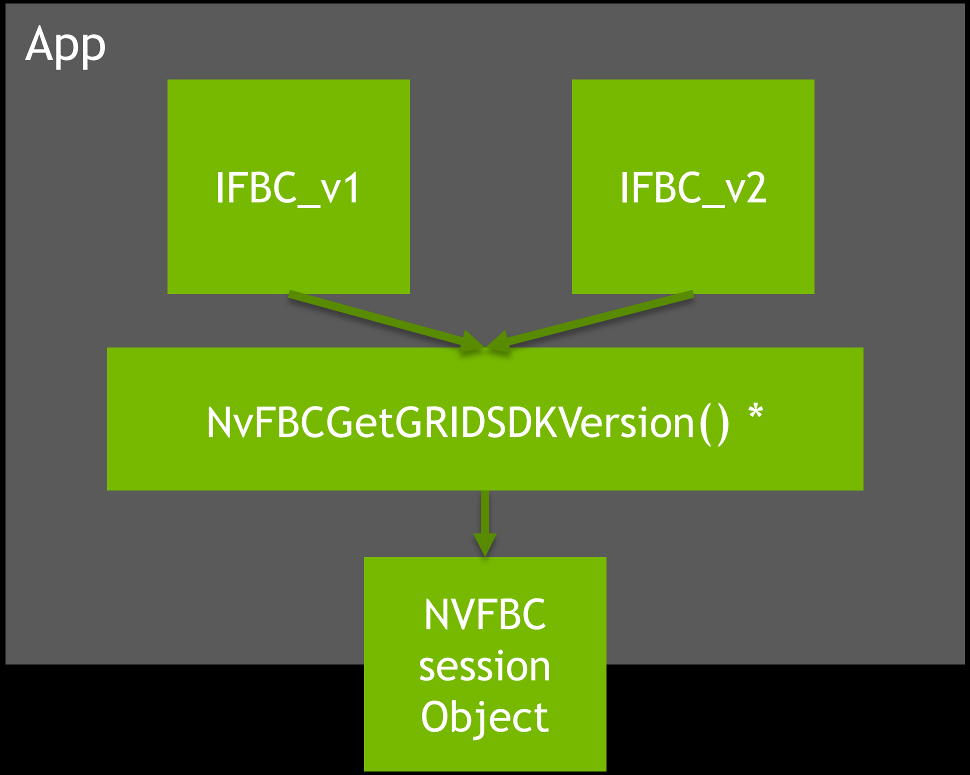
NVIDIA CAPTURE SDK - DRIVER COMPATIBILITY

GPU driver maintains backward compatibility with NVIDIA Capture SDK versions.

Compatibility of Upgraded Application (new SDK interfaces) with already deployed old GPU drivers needs special handling in application.

MANAGING SDK UPGRADES

Compile for multiple interface versions, select based on highest supported version at run-time



*Similar API is available for NVIFR

QUESTIONS ?

REFERENCES

Past GTC talks about related topics available here.

CLOUD VISUALIZATION

Presentation	Media
Accelerating Cloud Graphics **Franck Diard (NVIDIA)** A new NVIDIA SDK provides access to a class of key components which allow optimal capture, compression, streaming and low latency display of high performance games from the cloud. We demonstrate how all these components fit together to deliver a ... Read More **Keywords:** Cloud Visualization, GTC 2012 - ID S0627	**Streaming:** > Watch Now **Download:** > FLV > PDF

GRAPHICS VIRTUALIZATION

Presentation	Media
Cloud Gaming & Application Delivery with NVIDIA GRID Technologies **Franck Diard (NVIDIA)** This session presents the technologies behind NVIDIA GRID(TM) and the future of game engines and application delivery running in the cloud. The audience will learn about the key components of NVIDIA GRID, like optimal capture, efficient compression, ... Read More **Keywords:** Graphics Virtualization, Game Development, Remote Graphics & Cloud-Based Graphics, GTC 2014 - ID S4159	**Streaming:** > Watch Now **Download:** > PDF

REMOTE GRAPHICS & CLOUD-BASED GRAPHICS

Presentation	Media
Accelerating Cloud Graphics **Franck Diard (NVIDIA)** Franck Diard, Chief Software Architect at NVIDIA, will talk about the technologies behind GRID and how you can integrate them into your cloud products. The audience will learn about the key components, which allow optimal capture, efficient comp ... Read More **Keywords:** Remote Graphics & Cloud-Based Graphics, Media & Entertainment, GTC 2013 - ID S3543	**Streaming:** > Watch Now **Download:** > FLV > PDF

Resources

https://developer.nvidia.com/grid-app-game-streaming
http://www.nvidia.com/object/cloud-get-started.html
http://www.nvidia.com/object/enterprise-virtualization.html

NVIDIA VIDEO SDK: HW VIDEO ENCODING

Video Compression for game recording, remote desktop streaming

NVENC HW Encoder

- H.264 support
- HEVC (H.265) support
- Optimized encode settings for low latency streaming

NVIDIA Capture SDK enables easy integration with NVENC API

- NVIFRToHWEnc
- NVFBCToDX9Vid, NVFBCCuda, NVFBCToHWEnc

S6226 - High-Performance Video Encoding on NVIDIA GPUs

Abhijit Patait *Director, Multimedia System Software, NVIDIA*
Eric Young *GRID Developer Relations Manager , NVIDIA*

Day: Monday, 04/04
Time: 13:00 - 13:50
Location: Room 210F

We'll provide an overview of video encoding technologies available using NVIDIA GPUs. In particular, attendees can expect to learn the following: (1) overview of NVIDIA video encoding SDK (NVENC SDK); (2) new features in NVIDIA video encoding (NVENC) hardware with new GPU chips; (3) changes and new features in NVENC SDK 6.0 and NVENC SDK 7.0; and (4) differences between NVENC SDK and GRID SDK and using right SDK for a particular application.

Level: All
Type: Tutorial
Tags: Media & Entertainment; Video & Image Processing; Tools & Libraries

WELCOME TO THE NVIDIA VMWARE COMMUNITY

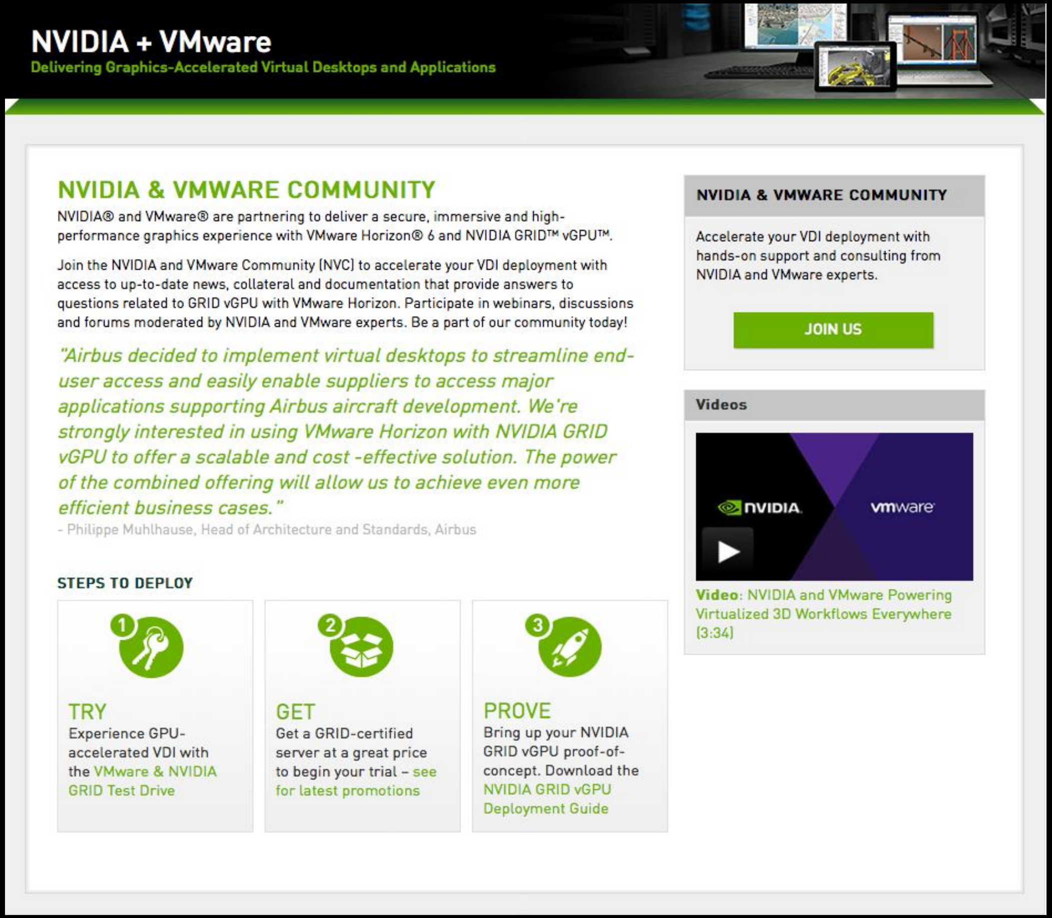
A community dedicated to NVIDIA and VMware solutions

Web portal with discussions, solution updates and basic sales support

Interact with peers, learn tips / tricks and accelerate NVIDIA GRID vGPU deployment on VMware

Available to any customer who completes a brief questionnaire

Join us today www.nvidia.com/nvc



GPU TECHNOLOGY CONFERENCE

April 4-7, 2016 | Silicon Valley

For Queries related to NVIDIA Capture SDK, get in touch with us at:
GRID-devtech-support@nvidia.com

THANK YOU

JOIN THE CONVERSATION
#GTC16

JOIN THE NVIDIA DEVELOPER PROGRAM AT developer.nvidia.com/join

PRESENTED BY NVIDIA.