GET TO KNOW THE NVIDIA GRID™ SDK

Shounak Deshpande, 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
- User Input
- Decode
- Render

SERVER
- NIC
- CPU
- Encode
- Capture
- Render

Network
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 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)
Support: GRID-devtech-support@nvidia.com
NVIDIA CAPTURE SDK COMPONENTS

**Interface Definitions**

- **NVFBC API**
  - Low Latency
  - Desktop Capture

- **NVIFR API**
  - Low Latency
  - Render Target Capture

- **NVENC**
  - Low latency
  - Hardware Encoder

- **NVFBC library**

- **NVIFR library**

**Sample Code**

**Documentation**

**GPU Driver**
# NVIDIA CAPTURE SDK: THE “CAPTURE” PART

<table>
<thead>
<tr>
<th>NVFBC</th>
<th>NVIFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute force, capture all on screen</td>
<td>No-frills RenderTarget capture</td>
</tr>
<tr>
<td>Orthogonal to Graphics APIs</td>
<td>Supports DirectX9,10,11, OpenGL APIs</td>
</tr>
<tr>
<td>Easy to integrate with NVENC API</td>
<td>Easy to integrate with NVENC API</td>
</tr>
<tr>
<td>Easy onboarding, no process injection</td>
<td>Needs to be injected in target process</td>
</tr>
<tr>
<td>Efficient than GDI-based screen scraping</td>
<td>One session per target window</td>
</tr>
<tr>
<td>One session per display</td>
<td>Enables higher density of streamed apps</td>
</tr>
</tbody>
</table>
# NVIDIA CAPTURE SDK: INTERFACES

<table>
<thead>
<tr>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NVFBC:</strong> NVIDIA Frame Buffer Capture</td>
<td><strong>NVIFR:</strong> NVIDIA In-band Frame Render</td>
</tr>
<tr>
<td>NVFBC</td>
<td>NVIFR - DirectX</td>
</tr>
<tr>
<td>NVFBCToSys</td>
<td>NVIFRToSys</td>
</tr>
<tr>
<td>NVFBCToCuda</td>
<td>NVIFRToCuda</td>
</tr>
<tr>
<td>NVFBCToHWEnc</td>
<td>NVIFRToHWEnc</td>
</tr>
<tr>
<td>NVFBCToDX9Vid</td>
<td>NVIFRToHWEnc</td>
</tr>
</tbody>
</table>

-ToHWEnc interfaces internally invoke NVENC API (part of NVIDIA Video Codec SDK)
# EVOLUTION OF NVIDIA CAPTURE SDK

<table>
<thead>
<tr>
<th>SDL</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDK</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Windows</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- GRID K340, K520, K1, K2, Quadro 4000+ support</td>
<td></td>
<td>• GRID M30 limited support</td>
<td>• GRID M30 full support</td>
</tr>
<tr>
<td>- H.264 encode support</td>
<td>2.3</td>
<td>• Maxwell NVENC enhancements - quarter-res first pass; lossless encoding; 4:4:4 encoding</td>
<td>• HEVC support</td>
</tr>
<tr>
<td>- Windows 7, 8, 8.1 support</td>
<td></td>
<td>• NVENC RC 2.0</td>
<td>• Tesla M60 support</td>
</tr>
<tr>
<td><strong>Linux</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- GRID K340, K520, K1, K2, Quadro 4000+ support</td>
<td></td>
<td>• GRID M30 full support</td>
<td>• HEVC support</td>
</tr>
<tr>
<td>- H.264 encode support</td>
<td></td>
<td>• NvIFR full parity for NVENC features with Windows</td>
<td>• Tesla M60 support</td>
</tr>
<tr>
<td><strong>HW</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- GRID K340, K520, K1, K2, Quadro K2000+</td>
<td>GRID M30, Quadro M6000</td>
<td>Tesla M60</td>
<td></td>
</tr>
</tbody>
</table>
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

Create NVFBC session object

Set up NVFBC session “Capture” starts here

Read Grabbed buffer
CAPTURING USING NVFBC

Begin

NvFBCGetStatusEx() Check NVFBC Status

NVFBC Not Enabled

NvFBCEnable() Enable NVFBC

NVFBC already in use

NSFBC disabled, not in use

NvFBCCreateEx() Create NVFBC Session

Success

Setup NVFBC Session

Success

Grab()

Success

Release NVFBC Session

Fail

Terminate

Fail

Exit

Success
Desktop Remoting using NVFBC + NVENC HW Encoder

- Desktop Composition [System Process]
- NV GPU Driver
- NVFBC Capture Process
  - Capture Thread
  - Captured buffer
- Encode Thread
- NVENC API
- Video Bitstream packet
- 3D HW
- NV GPU
- NVENC HW

Latency:
- 1 millisecond
- ~2 milliseconds
- ~4 milliseconds

*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

Application allocates output buffers and event handles

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

```
NVIFR_HW_ENC_SETUP_PARAMS params = {0};
params.dwVersion = NVIFR_HW_ENC_SETUP_PARAMS_VER;
params.dwBuffers = 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.cfgF = 26;
params.configParams.rateControl = NV_HW_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP;
params.configParams.presetConfig= NV_HW_ENC_PRES_LOW_LATENCY_HQ;
params.configParams.codec = g_CODEC;

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

NVIFRRESULT res = g_cIFR->NVIFRSetUpHWEncoder(&params);
```
DIRECTX APP STREAMING USING NVIFR HW ENCODER

```c
// 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_BIUSTREAM_PARAMS params = {0};
    params.dwVersion = NVIFR_HW_ENC_GET_BIUSTREAM_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);
```
OPENGL APP STREAMING USING NVIFR HW ENCODER

if (nvifr_initialize() == false)
{
    fprintf(stderr, "Failed to create a NzIFROGL instance.\n");
    return -1;
}

printf("NVIFROGL API version number: %d.%d\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) != NVIFROGL_SUCCESS)
{
    fprintf(stderr, "Failed to create a NvIFROGL session.\n");
    return -1;
}

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

config.profile = (codeType == NVIFROGL_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 = NVIFROGL_HW_ENC_RATE_CONTROL_CBR;
config.stereoFormat = NVIFROGL_HW_ENC_STEREO_NONE;
configpreset = NVIFROGL_HW_ENC_PRESET_LOW_LATENCY_HD;
config.codecType = codecType;
config.VBVBufferSize = config.avgBitRate;
config.VBVInitialDelay = config.VBVBufferSize;

if (nvifr_nvifroglCreateTransferToHwEncObject(sessionHandle, &config, &transferObjectHandle) != NVIFROGL_SUCCESS)
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.nvIFROGLockTransferData(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.nvIFROGReleaseTransferData(transferObjectHandle) != NV_IFROGL_SUCCESS)
{
    fprintf(stderr, "Failed to release the transferred data.\n");
    exit(-1);
}
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

Use High Performance Multimedia Timer for accuracy

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

Start Measurement before capture loop

Run through capture\encode loop

Stop Measurement here

double dStart = GetFloatingDate();
double dNow = 0;
DWORD dwBufferIndex=0;
printf("Starting Encode\n");
for (DWORD i=0; i<g_dMaxFrames; i++)
{
    DWORD k = i+g_dWinInputs;
    for (DWORD d=0; d<g_dWinEncoders; d++)
    {
        /*print("Input# %d encoded by Encoder# %d\n", i, d);
        WaitForSingleObject(aSideThreadData[d].m_abCanRenderEvents[dwBufferIndex], INFINITE);
        ResetEvent(aSideThreadData[d].m_abCanRenderEvents[dwBufferIndex]);
        */
        IDirect3DSurface9 * pCurrentRenderTarget = NULL;
        ....
        ....
        ResetEvent(aSideThreadData[d].m_abEncodeCompletionEvents[dwBufferIndex]);

        NVIFR_HW_ENG_TRANSFER_RT_TO_HW_ENC_PARAMS params = {0};
        params.dwVersion = NVIFR_HW_ENG_TRANSFER_RT_TO_HW_ENC_PARAMS_VER;
        params.encodePicParams.dwVersion = NVIFR_HW_ENC_PIX_PARAMS_VER;
        params.dwBufferIndex = dwBufferIndex;
        res = g_apIFRs[d]->NVIFRTransferRenderTargetToHWEncoder4(params);
        ....
    }
    dwBufferIndex = (dwBufferIndex+1)%UNFRAMESINFLIGHT;
}
double dEnd = GetFloatingDate();
printf("Total time %f sec, FPS=%f\n", dEnd-dStart, double(g_dMaxFrames)/(dEnd-dStart));
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 applications 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

Capture Thread
[write to buffer # i]

Buffer Queue

Encode\Post-process Thread
[read from buffer# (i-1)\%N]
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
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

App

IFBC_v1

IFBC_v2

NvFBCGetGRIDSDKVersion() *

NVFBC session Object

*Similar API is available for NVIFR
QUESTIONS ?
REFERENCES

Past GTC talks about related topics available [here](#).

**Cloud Visualization**

**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 53627

**Graphics Virtualization**

**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 S4129

**Remote Graphics & Cloud-Based Graphics**

**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

**Resources**

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

Abhiit Patani, Director, Multimedia System Software, NVIDIA
Eric Young, GRID Developer Relations Manager, NVIDIA

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
THANK YOU

JOIN THE CONVERSATION
#GTC16  

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