

GPU Computing Master Class

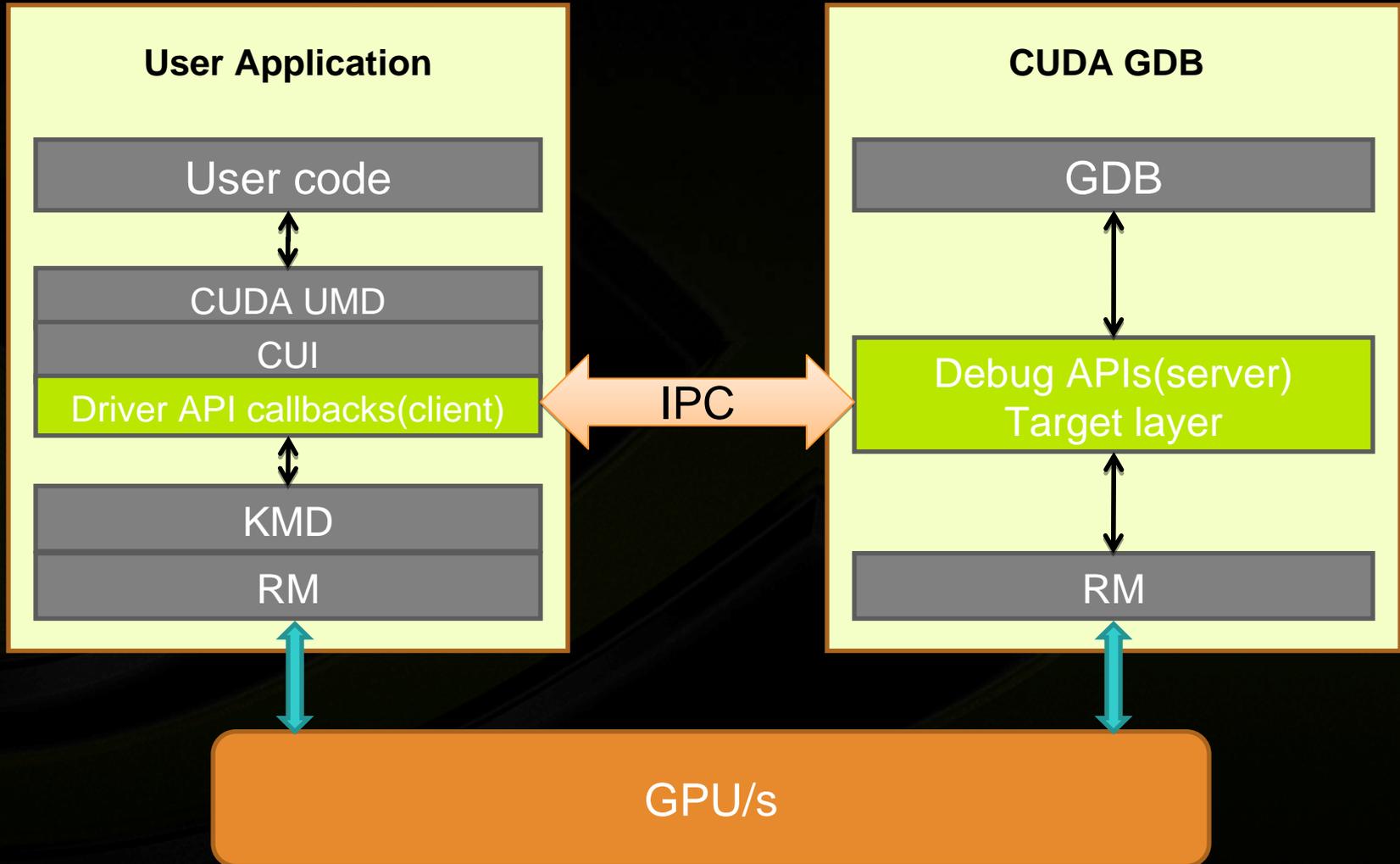
Development Tools



Generic CUDA debugger goals

- **Support all standard debuggers across all OS**
 - Linux – GDB, TotalView and DDD
 - Windows – Visual studio
 - Mac - XCode
- **Support CUDA runtime APIs**
- **Support CUDA driver APIs**
- **Unified debug environment**
 - **Provide simultaneous access to CUDA and host variables**
 - Present CUDA threads same as host threads
 - Present CUDA memory same as host memory
- **Support single and multi GPU debugging**
- **Attach to a running process – local or remote.**

Linux debugger model



IPC – Driver APIs to Debug APIs



- **Driver API callbacks retrieves the fatbin/SASS line information**
- **Debug APIs translates the SASS information to actual GPU address**
- **Debug APIs tracks all memory allocations**
- **Driver and Debug APIs report and manage kernel launch, execution and termination.**

Current Status



- **CUDA-GDB 3.0 Beta shipping to customers supporting:**
 - **New CUDA Memory Checker**
 - **Support for all the OpenCL features**
 - **Early support for Fermi Architecture**
 - **etc...**

Upcoming Roadmap



- **Migrate to standard binary formats – DWARF and ELF**
- **Performance Improvement**
- **New debug APIs to enable external customers**
- **Support source and assembly level debugging**
- **Support for Apps with multiple host threads using CUDA**
- **Support for Apps using multi-GPUs**

Windows Development



Development Environment



Visual Studio

CPU Code

System

Rendering

AI

Sound

Gameplay

Tools

Project Wizards

Editor

Debugger

Performance

Build System

GPU Code

Shader

CUDA

OpenCL

Shader
Authoring

Graphics
Debugging

Graphics
Performance

CUDA
Performance

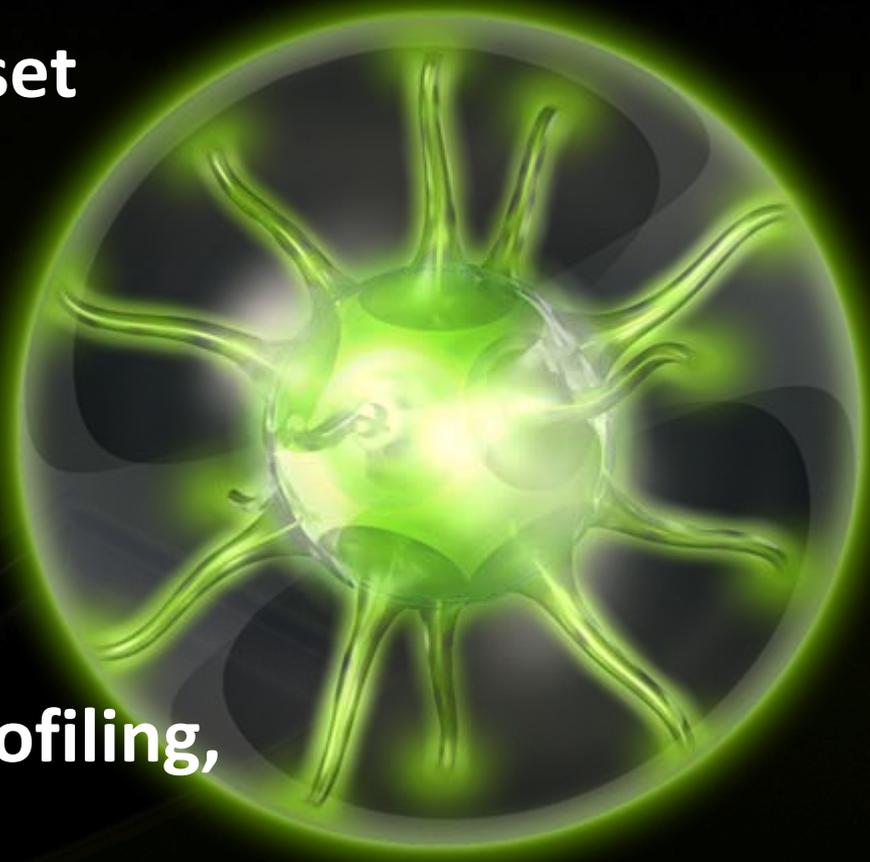
NVIDIA Nexus



Next-generation
development toolset

Homogeneous development
for CPU and GPU

Seamless debugging, profiling,
and visualization



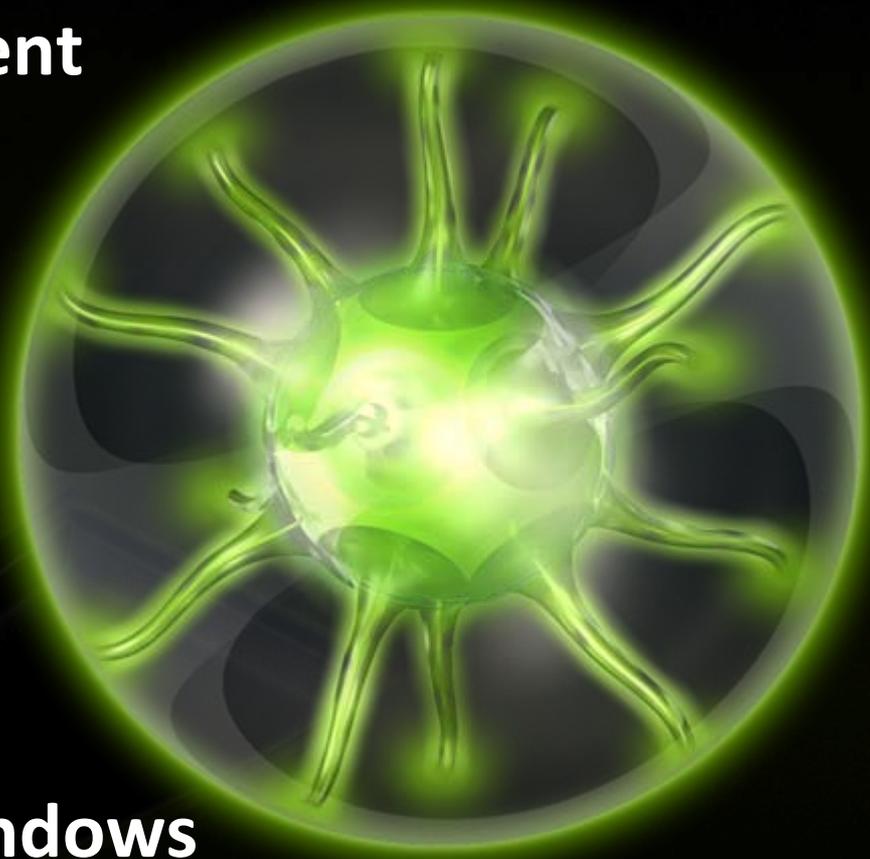
The Nexus 1.0 Release

A full **Visual Studio-integrated**
development environment

Supporting

CUDA C, DirectCompute,
OpenCL, DirectX 11,
DirectX 10, OpenGL

Requires Windows Vista or Windows
7, Visual Studio 2008 SP1



The Nexus Ecosystem



Development Environment



Visual Studio

CPU Code

System

Rendering

AI

Sound

Gameplay

Tools

Build System

Editor

Debugger

Performance

Gfx Analysis

CUDA Analysis

PhysX Analysis

GPU Code

Shader

CUDA

OpenCL*

Shader
Authoring

Code Authoring



Strong integration with the Visual Studio Shell.

Syntax highlighting

Code completion

Nexus toolbar, menu, and project wizards

```
effect.fx@ Main.cpp State Viewer
// this is our white rectangle
float2 vLightMax = vLightPos + g_fFilterSi
vLightPos.xy -= g_fFilterSize;
uint pStackLow = 0xffffffff, pStackHigh;
uint iPix = 0;
uint3 iLevel = uint3(0, 0, N_LEVELS - 1);
float fTotShadow = 1; ///< completely unshad

[loop] for ( ; ; )
{
    uint2 iPixel = uint2(iLevel.x + (iPix
    float fDiag = (float)(1 << iLevel.z);
    float2 vTexMin = iPixel * fDiag;

    // shrink texel to the white rectangle
    float2 vTexMax = min(vTexMin + fDiag, v
    vTexMax -= max(vTexMin, vLightPos.xy);

    // fetch the depth map
    float2 vPixel = iPixel + 0.5;
    [flatten] if (iLevel.z != 0)
```

Code Debugging



Step through GPU code, and examine program values.

```

FluidSim.cpp
extern "C"
__global__ void
matrixMul( float* C, float* A, float* B, int wA, int wB)
{
    // Block index
    int bx = blockIdx.x;
    int by = blockIdx.y;

    // Thread index
    dim3 threads(BLOCK_SIZE, BLOCK_SIZE);
    int tx = threadIdx.x;
    int ty = threadIdx.y;
    dim3 grid(WC / threads.x, HC / threads.y);

    // Index of the first sub-matrix of A processed by the block
    int aBegin = wA * BLOCK_SIZE * by;

```

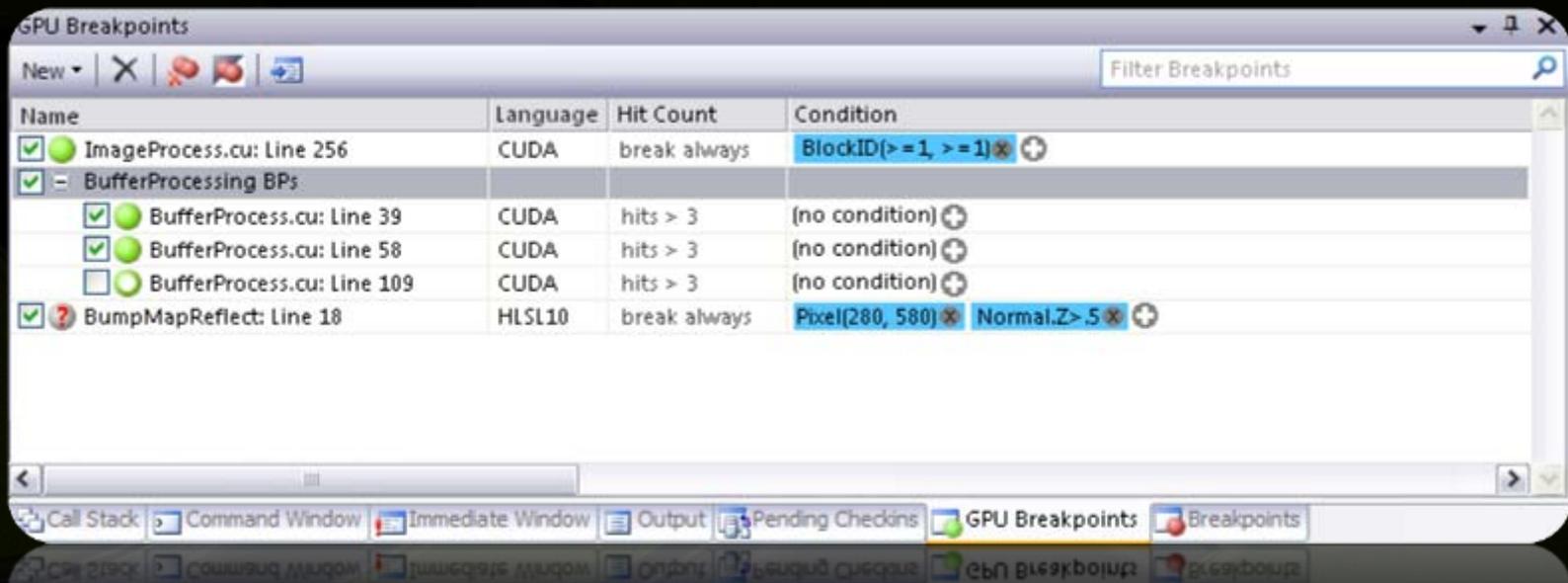
Call Stack		Locals		Language
Name	Name	Value	Type	
Test.exe!F	thid	24	int	CUDA
Test.exe!G	offset	1	int	\$GPU@Fermi
Test.exe!w	temp	0x0012edbc	float [1024]	C++
Test.exe!_	[0]	5.0000000	float	C
Test.exe!w	[1]	9.0000000	float	C
kernel32.d	[2]	1.0000000	float	
[Frames be	[3]	1.0000000	float	
	[4]	6.0000000	float	
	[5]	7.0000000	float	
	[6]	6.0000000	float	
	[7]	4.0000000	float	

processed by the block
the sub-matrices of A
B processed by the block
the sub-matrices of B

Code Debugging



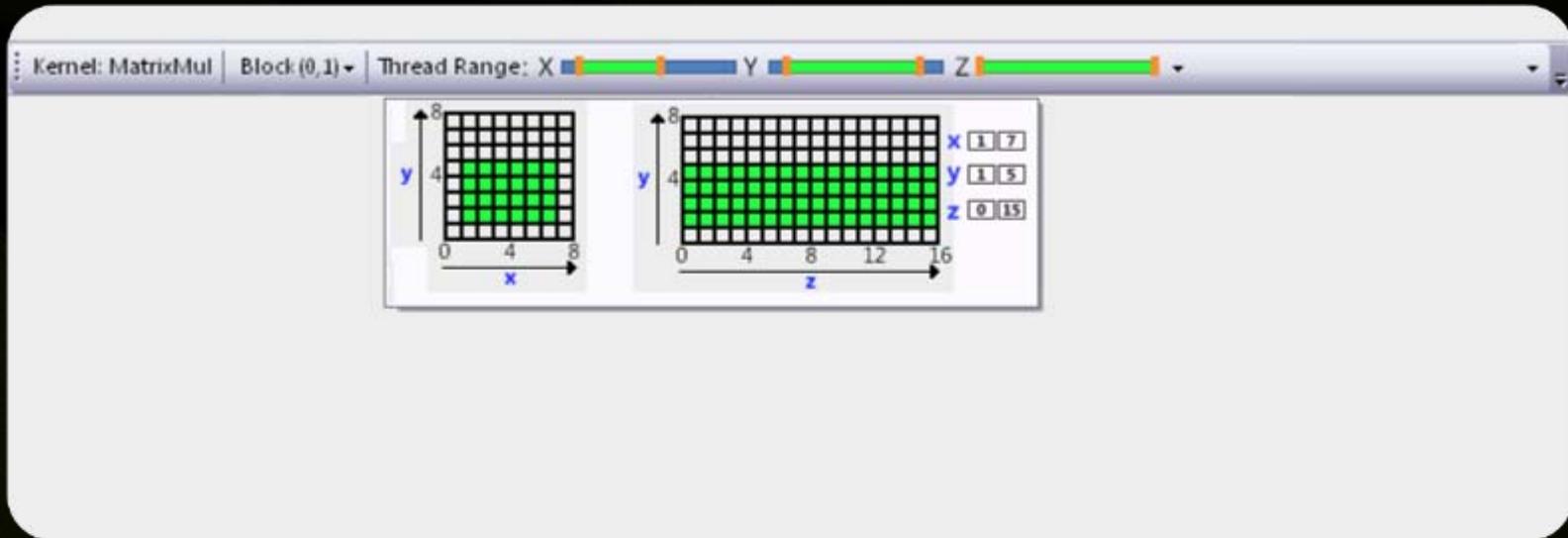
Power-up your breakpoints with parallel-aware conditionals.



Code Debugging



Manage debugging across thousands of threads.



Graphics



Drill down from examining a frame...

Capture ▶ Frame 2

457 Draw Calls
108,243 Primitives
2.1 Million pixels shaded (1M rejected)

Textures Render Targets Shaders

Search Textures

64x64 R8G8B8A8 NoiseTexture 1 8x8 L16A16 8x8x8 R8G8B8 64x64 R8G8B8A8

Draw Call Time

Sky Bullet Decal Character 1 Light Pass #1

0 457 226

Render Target Draw Call Geom

Tabbed information viewer allows the user to explore the resource types over the entire frame. The view of each tab can be changed between a thumbnail view, and a tabular details view. (Like Windows Explorer)

The current draw call is previewed here, in Render Target or Draw Call mode. Render Target mode shows the current render target for the draw call with the draw call wireframe highlighted. Draw Call Geom mode shows the draw call geometry only, and can be rotated.

Graphics



...to a draw call...

Capture ▶ Frame 2 ▶ Draw Call 37 ▶

◀ DrawIndexedInstance(3, 12, 4, 0, 0) ▶

Shader Resources (4 textures)

- 64x64 R8G8B8A8
- 8x8 L16A16
- 8x8x8 R8G8B8

Render Targets (800x600 backbuffer)

- Color
- Depth
- Stencil

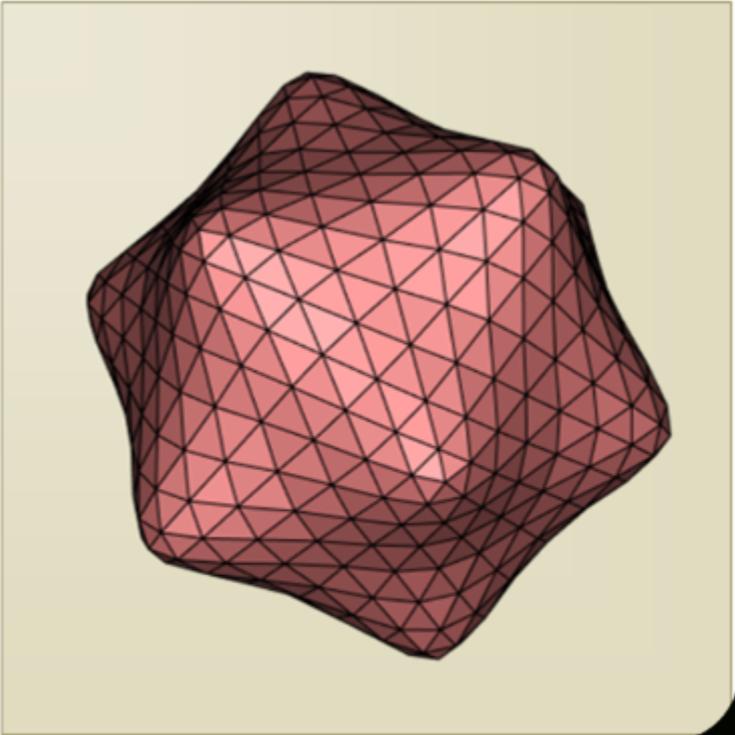
Pipeline State

Memory

Stream-Output

IA → VS → GS → RS → PS → OM

Active Shaders | Vertex Buffers | Dependencies | Performance



Graphics



...to specific resources and GPU state.

Capture > Frame 2 > Draw Call 37 > Shader Resource 0

2D Texture



D3D10_TEXTURE_2D_DESC

Width = 64
Height = 64
MipLevels = 1
ArraySize = 1
Format = 0x2 (R16F16B16A16_FLOAT)
SampleDesc = (Count = 1, Quality = 0)
Usage = DEFAULT
BindFlags = 0x00000028 (...)
CPUAccessFlags = 0x00000000
MiscFlags = 0x00000000

Miplevel atlas
5

Layer
4

Experiments
 2x2 Texture
 Colorize Mips
 Black and White

Visualization
Normal Map
Sampled Area
Transfer Function

Cubemap 3D / Array
Cross Cube
Cube Stack
Sphere

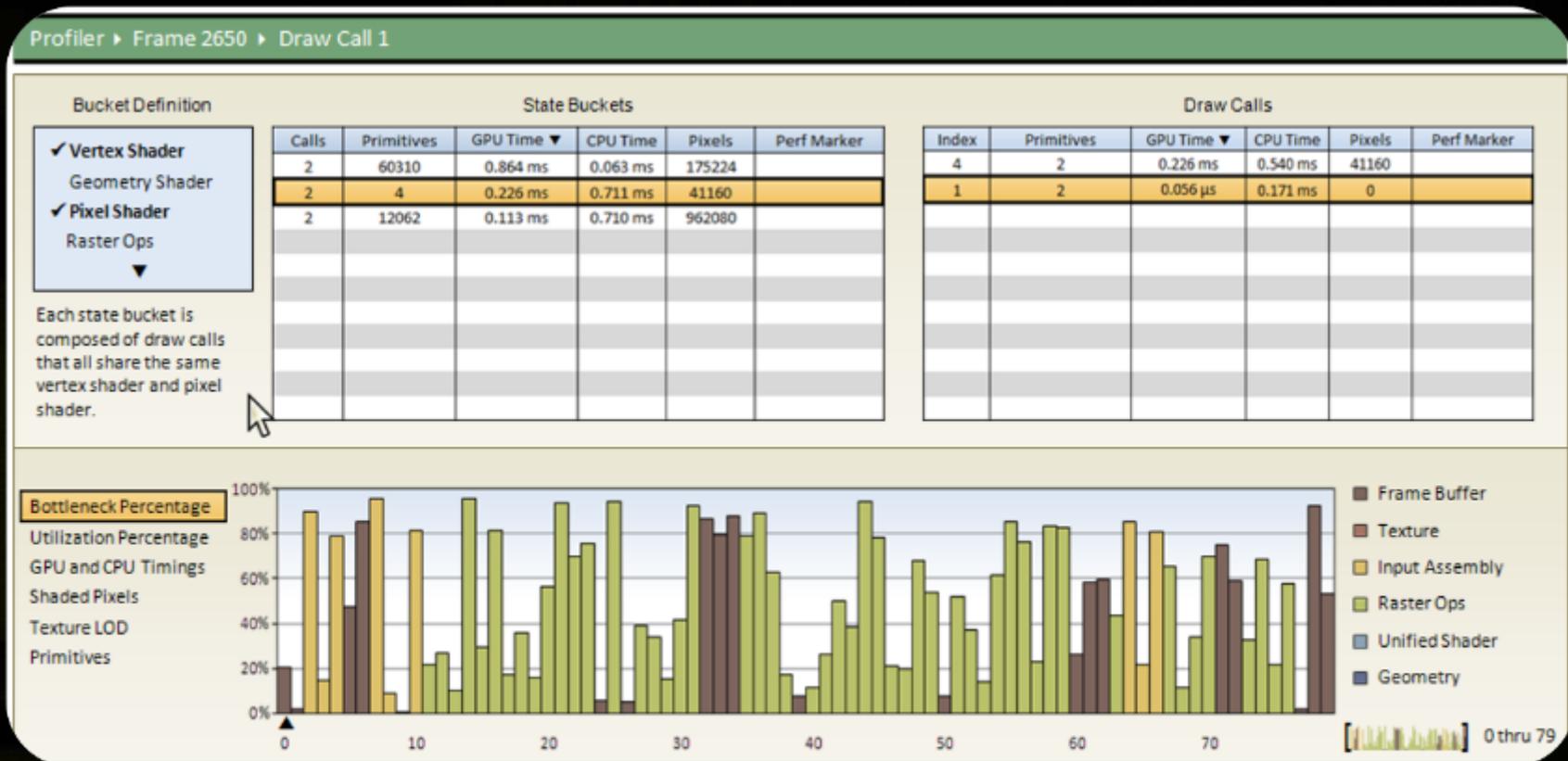


1:2 (r, g, b, a) = (0.4, 0.8, 0.2, 1.0) (s, t) = (0.152, 0.203)

Graphics



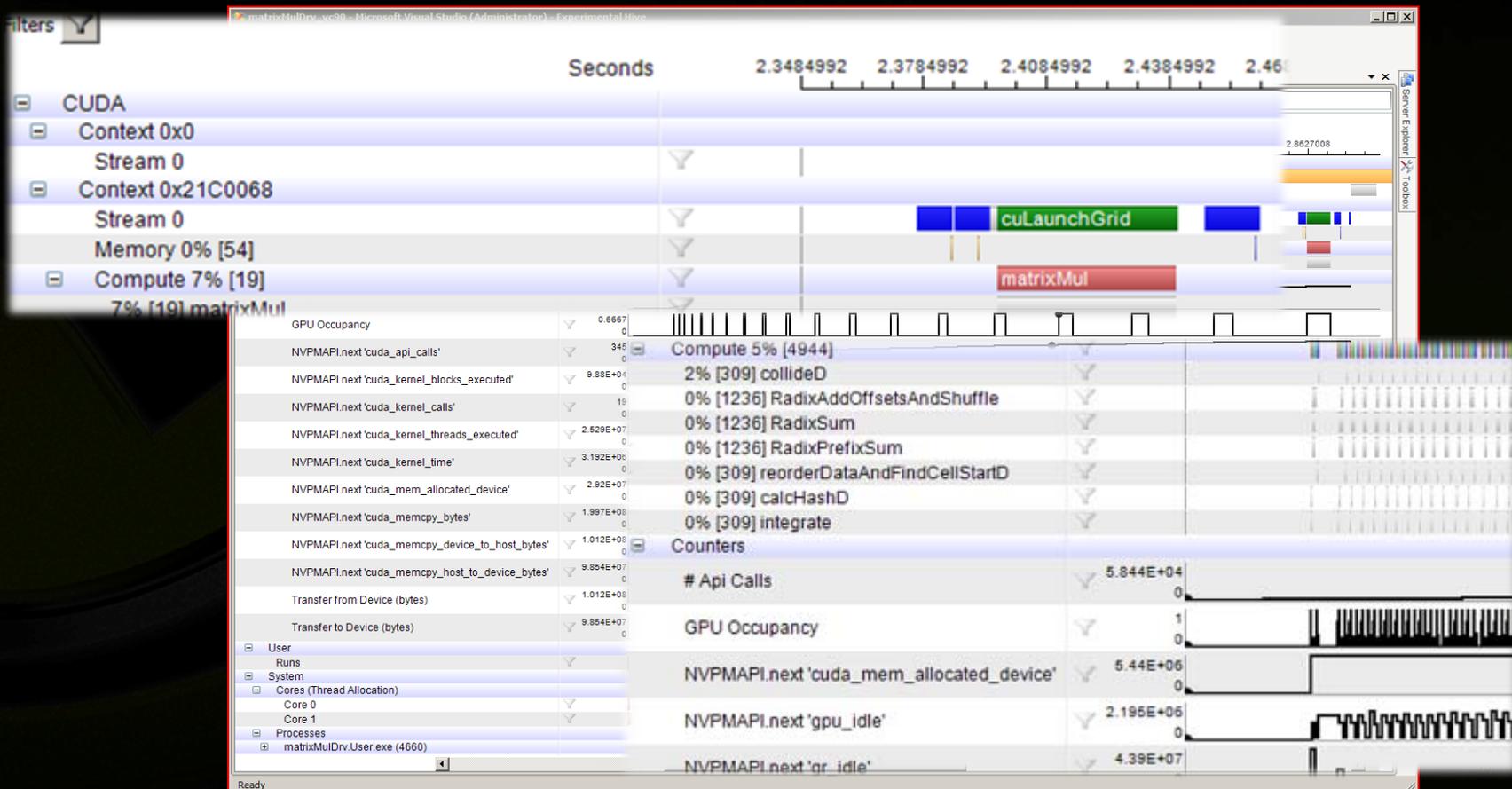
See GPU usage on a per-draw call basis.



Analysis



View performance from higher level using correlated CPU, GPU, thread, and OS data.



Analysis



Get detailed performance information for every kernel.

The screenshot shows a window titled 'particles.user.00030.nvreport' with a sub-tab 'Activity'. The main content is a table titled 'CUDA Methods Table'. The table has 12 columns: Name, Context ID, Stream ID, Duration (us), Occupancy, Regs/Thread, StatShm/Block, DynShm/Block, Blocks, Threads, GridDim, and BlockDim. The data consists of 15 rows, all for the kernel name 'integrate'. The Context ID is consistently 61334480 and the Stream ID is 0. The Duration values range from 54.40 to 59.60. The Occupancy is consistently 0.6666667. The Regs/Thread is 14, StatShm/Block is 48, DynShm/Block is 0, Blocks is 128, Threads is 256, and GridDim is (1, 128). The BlockDim is (256, 1, 1). The table is displayed in a standard Windows-style window with navigation buttons and a scrollbar.

Name	Context ID	Stream ID	Duration (us)	Occupancy	Regs/Thread	StatShm/Block	DynShm/Block	Blocks	Threads	GridDim	BlockDim
integrate	61334480	0	59.00	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	56.10	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	57.70	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	55.60	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	57.50	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	57.50	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	55.90	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	58.50	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	55.20	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	54.40	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	56.10	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	59.60	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	57.70	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	56.00	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	56.90	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	58.30	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)
integrate	61334480	0	57.00	0.6666667	14	48	0	128	256	(1, 128)	(256, 1, 1)

Nexus everywhere



Debug, visualize, and optimize your code remotely.

