



NVIDIA Parallel Nsight™

Siggraph Asia 2010 | Seoul



Agenda

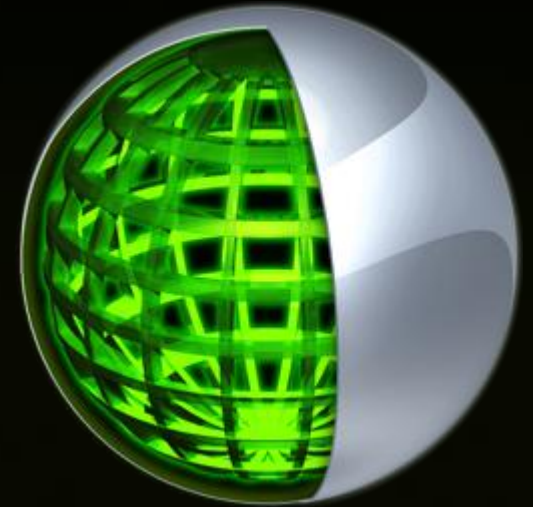


- **Introduction to Parallel Nsight**
- **CUDA C/C++ Source Debugging**
- **Analysis/System Trace**
- **Q&A**

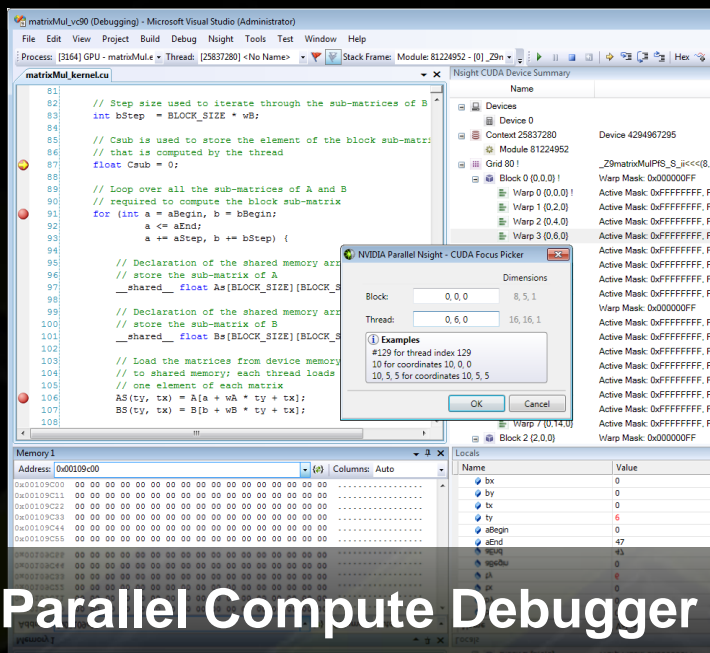
What is Parallel Nsight?



- **Development environment for heterogeneous platforms (CPU and GPU)**
- **Fully integrated into Microsoft Visual Studio 2008 and 2010**
- **Dramatic productivity improvement in common development tasks**



Parallel Nsight for Compute



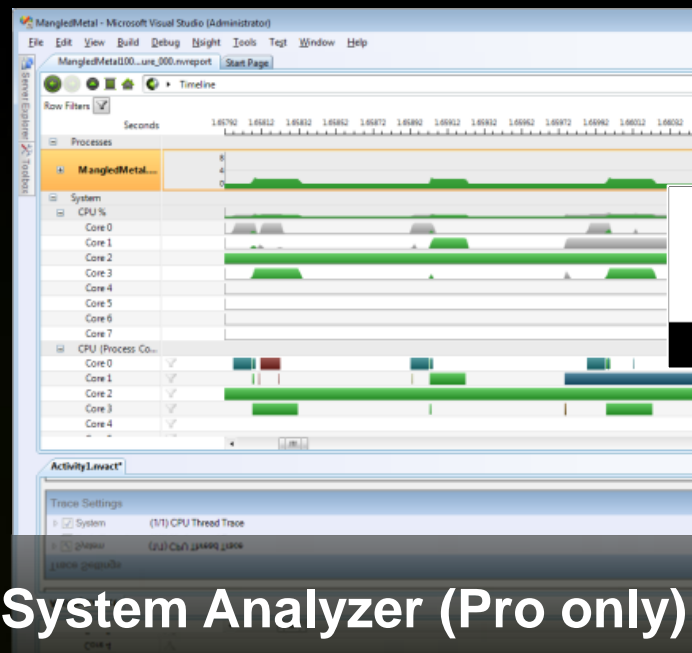
Parallel Compute Debugger

Examine compute kernels directly on GPU hardware

Debug CUDA C/C++ and DirectCompute applications

Visualize thousands of threads executing in parallel using Visual Studio

Use conditional breakpoints to correct errors in massively parallel code



System Analyzer (Pro only)

Capture and visualize CPU and GPU level events on a single correlated timeline

Inspect workload dependencies using the Timeline View

Profile CUDA kernels using GPU performance counters



CUDA C/C++ Debugging



- **Compile your code with Debug flag**
- **Use the familiar Visual Studio interface to debug your GPU code**
- **Dramatic productivity improvements**
 - **Explore memory during a live session vs. coding specific transfers**
 - **Immediately view live variables vs. printf/recompile loop**
 - **Set data breakpoints on memory area vs. trial and error**
 - **And much more!**

Setting Breakpoints



```
simpleStreams_vc10...pture_000.nvreport  Activity3.nvact*  simpleStreams.cu  matrixMul_kernel.cu X
(Global Scope)
int bx = blockDim.x;
int by = blockDim.y;

// Thread index
int tx = threadIdx.x;
int ty = threadIdx.y;

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

// Index of the last sub-matrix of A processed by the block
int aEnd = aBegin + wA - 1;

// Step size used to iterate through the sub-matrices of A
int aStep = BLOCK_SIZE;

// Index of the first sub-matrix of B processed by the block
int bBegin = BLOCK_SIZE * bx;

// Step size used to iterate through the sub-matrices of B
int bStep = BLOCK_SIZE * wB;

// Csub is used to store the element of the block sub-matrix
// that is computed by the thread
float Csub = 0;

// Loop over all the sub-matrices of A and B
// required to compute the block sub-matrix
for (int a = aBegin, b = bBegin;
     a <= aEnd;
     a += aStep, b += bStep) {

    // Declaration of the shared memory array As used to
    // store the sub-matrix of A
```

Viewing Variable Values



Watch 1

Name	Value	Type
$a + wA * ty + tx$	16	int
$C[a + wA * ty + tx]$	13.683188	<code>__device__ float&</code>
$C[a + wA * ty + tx + 10]$	13.49013	<code>__device__ float&</code>
$C[16] + 34$	47.683189	float

Autos Locals Watch 1

Viewing GPU Memory



Memory 1

Address: C Columns: 8

0x0000000000216000	10.265490	12.644010	12.404924	12.270883	13.781327	12.225714	11.5
0x0000000000216020	14.802514	13.659491	12.088912	13.482843	14.150903	11.562345	10.6
0x0000000000216040	13.683188	11.923380	11.407702	12.153029	11.826190	12.571199	14.3
0x0000000000216060	12.318817	11.895715	13.490130	11.528786	11.917480	11.057190	11.4
0x0000000000216080	14.219353	12.578764	12.981959	11.614140	11.367486	11.236432	10.8
0x00000000002160A0	12.162021	12.395756	12.909150	10.340583	11.732928	11.289549	13.2
0x00000000002160C0	13.021592	13.032330	11.214454	10.869614	14.484652	11.907482	15.5
0x00000000002160E0	14.306890	13.693730	14.891927	13.105434	13.241500	12.827445	14.5

matrixMul_kernel.cu

(Global Scope) matrixMul(float * C, float * A, float * B, int wA, int wB)

```
// sub-matrices of A and B in the next iteration
__syncthreads();
}

// Write the block sub-matrix to device memory;
// each thread writes one element
int c = wB * BLOCK_SIZE * by + BLOCK_SIZE * bx;
C[c + wB * ty + tx] = Csub;

#endif // #ifndef _MATRIXMUL_KERNEL_H_
```

100 %

Switching Between Threads



Memory 1

Address: 0x000000005206000

0x000000005206000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x000000005206020	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x000000005206040	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x000000005206060	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x000000005206080	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x0000000052060A0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x0000000052060C0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0x0000000052060E0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Memory 1 Modules

simpleStreams_vc10...pture_000.nvreport Activit

(Global Scope)

```
// Index of the last sub-matrix
int aEnd = aBegin + wA - 1;

// Step size used to iterate th
int aStep = BLOCK_SIZE;

// Index of the first sub-matrix
int bBegin = BLOCK_SIZE * bx;

// Step size used to iterate through the sub-matrices of B
int bStep = BLOCK_SIZE * wB;

// Csub is used to store the element of the block sub-matrix
// that is computed by the thread
float Csub = 0;
```

100 %

NVIDIA Parallel Nsight - CUDA Debug Focus

Dimensions

Block: 3, 8, 5, 1

Thread: 0-4

Valid Indices for Y (Filtered)

Examples

- #129 for block index 129
- 10 for coordinates 10, 0
- 10, 5 for coordinates 10, 5

OK Cancel

Conditional Breakpoints



```
matrixMul_kernel.cu X
(Global Scope)
    a <- aEnd,
    a += aStep, b += bStep) {

    // Declaration of the shared memory array As
    // store the sub-matrix of A
    __shared__ float As[BLOCK_SIZE][BLOCK_SIZE];

    // Declaration of the shared memory array Bs
    // store the sub-matrix of B
    __shared__ float Bs[BLOCK_SIZE][BLOCK_SIZE];

    // Load the matrices from device memory
    // to shared memory; each thread loads
    // one element of each matrix
    AS(ty, tx) = A[a + wA * ty + tx];
    BS(ty, tx) = B[b + wB * ty + tx];

    // Synchronize to make sure the matrices are
```

Data Breakpoints



Breakpoints

New Columns Search:

Break at Function... Ctrl+B

New Data Breakpoint...

Condition	Hit Count
(no condition)	break always (current:0)

(Global Scope) matrixMul(float * C, float * A, float * B, int wA, int wB)

```
// computation is done before loading two new
// sub-matrices of A and B in the next iteration
__syncthreads();
}

// Write the block sub-matrix to device memory;
// each thread writes one element
int c = wB * BLOCK_SIZE * by + BLOCK_SIZE * bx;
C[c + wB * ty + tx] = Csub;
}

#endif // #ifndef _MATRIXMUL_KERNEL_H_
```

100 %

Microsoft Visual Studio

The following breakpoint was hit:

When '0x0000000005206000' changes (4 bytes) in process 'matrixMul_vc100.exe'

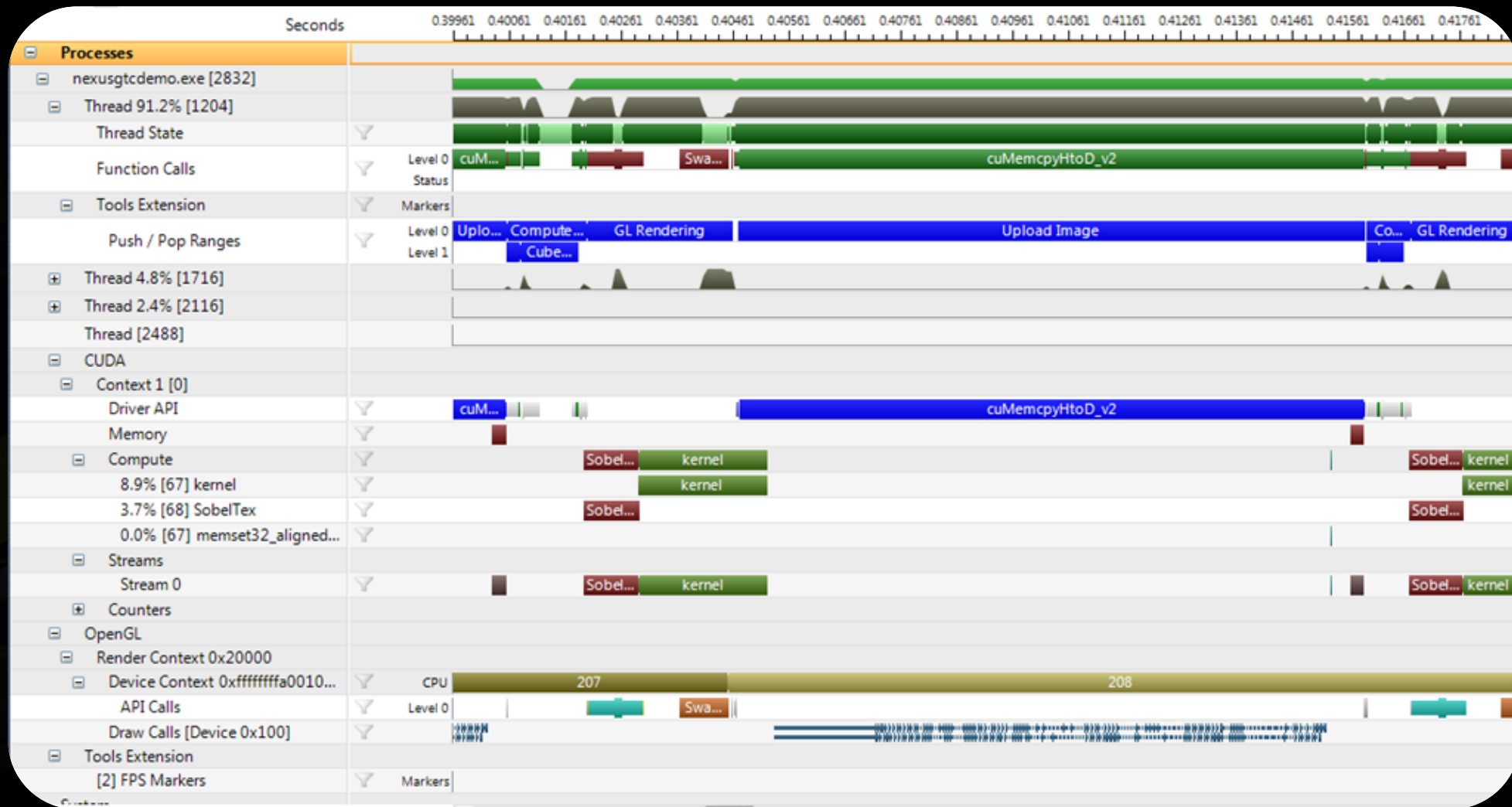
OK

Trace

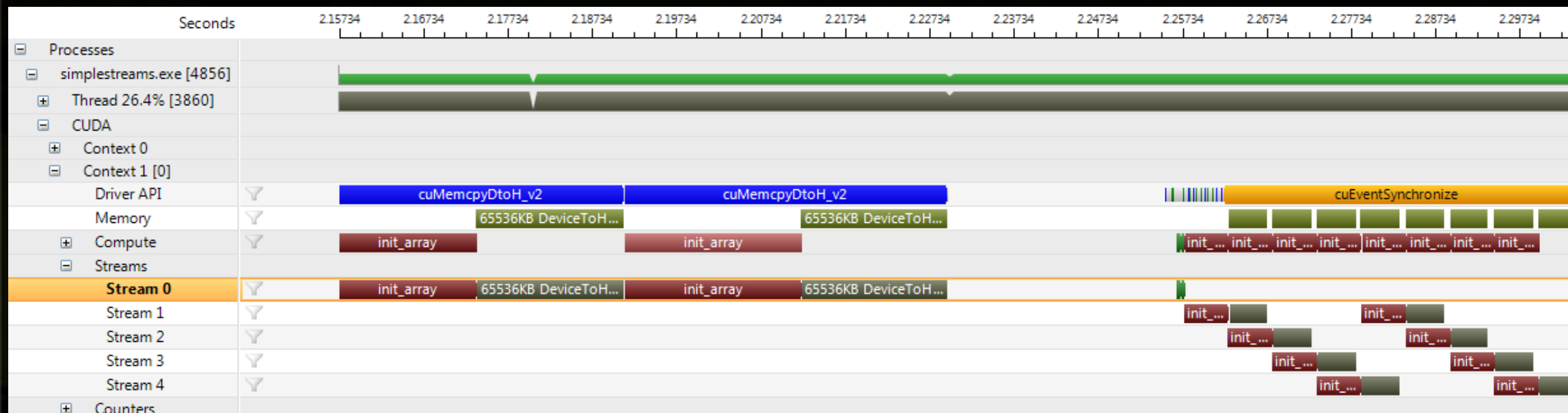


- **Powerful performance tool**
- **Correlated timeline between CPU and GPU**
- **Bottleneck identification**
 - Find CPU vs. GPU boundedness
 - Find memory transfer vs. kernel computation bounded
 - Get macro-level information on which CUDA kernels use the most time

Trace across the CPU and GPU



Trace – Overlap Memory Transactions



Tesla Compute Cluster Support (TCC)



- **TCC is a special driver mode for Windows 7, Vista, and HPC Server 2008.**
 - **Included in our most recent R260 driver release.**
- **Benefits**
 - **Lower overhead to kernel launches = higher performance**
 - **Running CUDA on a MS Remote desktop**
- **Parallel Nsight 1.5 now supports debugging on GPUs using a TCC driver.**

4 Flexible GPU Development Configurations



Desktop

Single machine, Single NVIDIA GPU

Analyzer, Graphics Inspector



Single machine, Dual NVIDIA GPUs

Analyzer, Graphics Inspector, Compute Debugger

Networked

Two machines connected over the network

Analyzer, Graphics Inspector, Compute Debugger, Graphics Debugger



Workstation SLI

SLI Multi OS workstation with two Quadro GPUs

Analyzer, Graphics Inspector, Compute Debugger, Graphics Debugger



Parallel Nsight 1.5 Feature Support



	Standard (no cost)	Professional (\$349) <i>available for purchase in December</i>
Compute Debugger	✓	✓
DirectX 10 & 11 Debugger & Graphics inspector	✓	✓
GeForce Support: 9 series or higher	✓	✓
Tesla Support: C1050/S1070 or higher	✓	✓
Quadro Support: G9x or higher	✓	✓
Windows 7, Vista and HPC Server 2008	✓	✓
Visual Studio 2008 SP1 and Visual Studio 2010	✓	✓
Compute Analyzer		✓
OpenGL and OpenCL Analyzer		✓
DirectX 10 & 11 Analyzer		✓
Tesla Compute Cluster (TCC) Debugging		✓

<http://www.nvidia.com/GetParallelNsight>

Parallel Nsight Resources



- **Parallel Nsight GPU Computing Forum**
- **The Parallel Nsight User Guide**
 - Installed with the Host installer
 - Available on the Web
- Links to these from: <http://developer.nvidia.com/ParallelNsight>



 developer.nvidia.com/ParallelNsight 

**Free Download
Videos
Documentation**

