NVIDIA Parallel Nsight: Debugging Massively Parallel Applications

San Jose | September 2010
NVIDIA Parallel Nsight™

Application Development Environment for Heterogeneous Platforms

- Build GPU code
- Debug GPU APIs and source code
- Profile GPU APIs and source code
- Trace and analyze GPU+CPU activities
Debugging Massively Parallel Application

- 10000’s threads
- Complex memory Hierarchy
- Warp debugging
- Synchronization/barrier
CUDA Architecture

Thread

per-thread local memory

Block

per-block shared memory

Global barrier

Kernel 0

Kernel 1

Local barrier

per-device global memory
Warp Debugging

32 thread SIMD

Breakpoint Hit

Stepping

Stepping over a barrier

Active Mask 0x00000001

Warp N

mov

Add

BR

mul

Warp N+1
Parallel Nsight - Debugger

- CUDA-C kernels
- DirectCompute shaders
- HLSL graphics shaders
- Conditional breakpoint
- Expression engine
- Data breakpoint
- GPU memory viewer
## Hardware Configurations

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single machine, Single GPU</strong></td>
<td>Analysis, Graphics Inspector</td>
</tr>
<tr>
<td><strong>Two machines, connected over the network</strong></td>
<td>Debugger, Analyzer, Graphics Inspector</td>
</tr>
<tr>
<td><strong>SLI or SLI MultiOS: Single machine, Dual GPU</strong></td>
<td>Debugger, Analyzer, Graphics Inspector</td>
</tr>
</tbody>
</table>
Tesla Compute Cluster (TCC) Support

- GPU registered as a compute device (WDM)
- Enables GPUs for Remote Desktop Protocol (RDP)
- Delivers increased performance

- Supported on Windows HPC Server 2008 R2 and Windows 7

- Tesla S/C/M series
Setting up CUDA builds in VS2010
Remote Debugging
CUDA Memory Checker

```c
x = 1-y;
offset = (x - w) * 3;
color = make_float3(float(pCudaTexPtr[offset + 2]), float(pCudaTexPtr[offset+1]), float(pCudaTexPtr[offset]));
pix00 = (color.x + color.y + color.z) / 3;
```
Sobel Filter Debugging

\[
C_{\text{horizontal}} = \begin{pmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{pmatrix}
\]

\[
C_{\text{vertical}} = \begin{pmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{pmatrix}
\]

\[
\text{Magnitude}_{\text{Sobel}} = \text{norm} \cdot \sqrt{G_{\text{horizontal}}^2 + G_{\text{vertical}}^2}
\]

\[
\text{Direction}_{\text{Sobel}} = \arctan \left( \frac{G_{\text{vertical}}}{G_{\text{horizontal}}} \right)
\]
```c
__global__ void SobelTex( unsigned char *pCudaTex, unsigned char *pSobelOriginal, unsigned char *pSobel ) {
    unsigned char *pSobel = pSobelOriginal + blockIdx.x * Pitch;
    unsigned char *pCudaTexPtr = pCudaTex + blockIdx.x * Pitch3;
    for ( int i = threadIdx.x; i < w; i += blockDim.x ) {
        unsigned int x, offset; x = i;
        offset = x * 3;
        color = make_float3(float(pCudaTexPtr[offset + 2]), float(pCudaTexPtr[offset + 1]), float(pCudaTexPtr[offset]));
        color = make_float3((color.x + color.y + color.z) / 3);
        unsigned char pixi11 = pix11;
        unsigned char pixi01 = pix11;
        unsigned char pixi21 = pix11;
        unsigned char pixio = pix11;
```
offset = (x + w) * 3;

    color = make_float3(float(pCudaTexPtr[offset + 2]), float(pCudaTexPtr[offset+1]), float(pCudaTexPtr[offset]));
    pix21 = (color.x + color.y + color.z) / 3;

    if (i < w - 1)
    {
        x = i+1;
        offset = (x + w) * 3;
        color = make_float3(float(pCudaTexPtr[offset + 2]), float(pCudaTexPtr[offset+1]), float(pCudaTexPtr[offset]));
        pix22 = (color.x + color.y + color.z) / 3;
    }

    pSobel[i] = ComputeSobel(pix00, pix01, pix02,
    pix10, pix11, pix12,
    pix21, pix22, fScale );

// Wrapper for the __global__ call that sets up the texture and threads
extern "C" void sobelFilter(unsigned char * cuda tex, unsigned char * pSobelOriginal, unsigned int int j, unsigned int int i, SobelMode mode, float fScale, cudaStream_t

__device__ unsigned char
ComputeSobel(unsigned char ul, // upper left
unsigned char um, // upper middle
unsigned char ur, // upper right
unsigned char ml, // middle left
unsigned char mm, // middle (unused)
unsigned char mr, // middle right
unsigned char ll, // lower left
unsigned char lm, // lower middle
unsigned char lr, // lower right
float fScale )
{
    short Horz = ur + 2*mr + lr - ul - 2*ml - ll;
    short Vert = ul + 2*um + ur - ll - 2*ml - lr;
    short Sum = (short) (fScale*(abs(Horz)+abs(Vert)));
    //
    return (unsigned char) Sum;
}

// cuda tex - BGR image
CUDA Memory Breakpoint

```c
unsigned int x = blockIdx.x*blockDim.x + threadIdx.x;
unsigned int y = blockIdx.y*blockDim.y + threadIdx.y;

int idx = y*width+x;
if (idx < width && idx % 2 == 0)
g_data[idx] = 255;

// Wrapper for the __global__ call that sets up the texture and threads
extern "C" void sobelFilter(unsigned char *cuda_tex, unsigned char *sobel_tex, int iw, int ih, enum SobelMode mode, float fScale, cudaS
```
Schedules and Where To Get It...

- Parallel Nsight 1.5 RC available now
- Parallel Nsight 1.5 Final soon after GTC

www.nvidia.com/ParallelNsight
Conclusion and Q&A

- GPU is a first-class development target
- CUDA-C development in Visual Studio 2010
- Tesla Compute Cluster support for Professionals
- CUDA Toolkit 3.1 and 3.2 Support
NVIDIA Parallel Nsight™ at GTC 2010

Parallel Nsight Lounge by Microsoft (Ballroom Concourse)
From 10am-8pm each day, give-a-ways daily at 3pm

All Parallel Nsight Sessions at GTC are in Room B
Tues, 3-3:50pm: GPGPU Development for Windows HPC Server [Microsoft]
Tues, 4-4:50pm: Parallel Nsight: Analyzing Massively Parallel Applications
Tues, 5-5:50pm: Parallel Nsight for Accelerated DirectX 11 Development