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



*	🖓 matrixMul vc90 (Debugging) - Microsoft Visual Studio (Administrator)											
The maximum of the second se												
	Hie Eart View Project Build Debug Nsight Tools Test Window Help											
P	Process: [3164] GPU - matrixMuLe 🗸 Thread: [25837280] <no name=""> 👻 👻 Stack Frame: Module: 81224952 - [0] _Z9n 🖕 🖡 🕨 💷 💷 🔶 🌳 🕮 💭 😤 🚰 Hex 🥱 🗓</no>											
1	matrixMul_kernel.cu - × Nsight CUDA Device Summary											
	81			Name								
	82	// Step size used to iterate through the sub-m	atrices of B *	P Devices								
	83	int bStep = BLOCK_SIZE * wB;		E Device 0								
	84			- Context 25927290	Davias 4204067205							
	85	// Csub is used to store the element of the bl	ock sub-matri	Madula 91224052	Lievice 420400/200							
	86	<pre>// that is computed by the thread</pre>		Module 81224952	70							
	87	float Csub = 0;		I III Gna su !	_Z9matrixMulPfS_S_ii<<<(8,5							
	88	// Teen ever all the sub matrices of 2 and P		Block 0 {0,0,0} !	Warp Mask: 0x000000FF							
	0.9	// required to compute the block sub-matrix		■ Warp 0 {0.0.0} !	Active Mask: 0xFFFFFFFF, P							
	91	for (int a = aBegin b = bBegin;		Warp 1 {0,2,0}	Active Mask: 0xFFFFFFFF, P							
	92	a <= aEnd;		Warp 2 (0.4.0)	Active Mask: 0xFFFFFFFF, P0							
	93	a += aStep, b += bStep) {			Active Mask: 0xFFFFFFFF, PC							
	94	(A)	CUDA Focus Picker	Active Mask: 0xFFFFFFFF, PC								
	95	// Declaration of the shared memory arr	A state of the state of the signed		Active Mask: 0xFFFFFFFF, PC							
	96	// store the sub-matrix of A	Dimensions Active Mask: 0xFFFFFFFF, PC									
	97	sharedfloat As[BLOCK_SIZE][BLOCK_S	Plack	0.0.0 8.5.1	Active Mask: 0xFFFFFFFF, P0							
	98		01010	0, 0, 0	Warp Mask: 0x000000FF							
	100	// store the sub-matrix of B	0, 6, 0 16, 16, 1	Active Mask: 0xEEEEEEE P								
	101	shared float Bs(BLOCK SIZE)(BLOCK S	(Ar I		Active Mask: 0xFFFFFFFFF, PC							
	102		1 Examples	120	Active Mask OxEEEEEEE PC							
	103	// Load the matrices from device memory	+129 for thread ind	10 0 0	Active Mask: 0xFFFFFFFF, PI							
1	104	<pre>// to shared memory; each thread loads</pre>	10.5.5 for coordin	ates 10. 5. 5	Active Mask, UXPEPEPEP, PC							
	105	// one element of each matrix			Acuve mask. UXPEPEPEPE, PC							
	106	AS(ty, tx) = A[a + wA * ty + tx];		OK Cancel	Active Mask: 0xFFFFFFFF, PI							
	107	BS(ty, tx) = B[b + wB * ty + tx];			Active Mask: 0xFFFFFFFF, P0							
1	108			Warp 7 {0.14.0}	Active Mask: 0xFFFFFFFF, PC							
			-	Block 2 {2,0,0}	Warp Mask: 0x000000FF							
Memory 1 - A × Locals												
Ac	dress: 0:	x00109c00 - {#} Colum	Name	Value								
0ж0	0109000	0 00 00 00 00 00 00 00 00 00 00 00 00 0		OX OX	0							
0x0	0109011	1 00 00 00 00 00 00 00 00 00 00 00 00 00		a tx	0							
0x0	00109C33 00 00 00 00 00 00 00 00 00 00 00 00 0											
0×0	0109044	4 00 00 00 00 00 00 00 00 00 00 00 00 00		🥥 aBegin	0							
0×0	0109055	$5 00 \ 00 \ 00 \ 00 \ 00 \ 00 \ 00 \ 0$		aEnd	47							
				0 9010	-43							

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 Server Explorer 🔭 Toolbo (Global Scope) IIIC DX - DIOCKIUX.X, int by = blockIdx.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; É. // Csub is used to store the element of the block sub-matrix // that is computed by the thread float Csub = 0; Ė // 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) { Ē // Declaration of the shared memory array As used to // store the sub-matrix of A 100 % - 4

Viewing Variable Values



Ŧ

Wa	₩atch 1								
	Name	Value	Туре						
	🥥 a + wA * ty + tx	16	int						
	C[a+ wA *ty + tx]	13.683188	devicefloat&						
	C[a + wA * ty + tx + 10]	13.49013	devicefloat&						
	C[16] + 34	47.683189	float						



Viewing GPU Memory



Memory1									
Addre	ss: C					🚽 🕼 Column	s: 8	•	
0x000	000000216000	10.265490	12.644010	12.404924	12.270883	13.781327	12.225714	11.5 🔺	
0x0000	000000216020	14.802514	13.659491	12.088912	13.482843	14.150903	11.562345	10.6	
0x0000	000000216040	13.683188	11.923380	11.407702	12.153029	11.826190	12.571199	14.3	
0x0000	000000216060	12.318817	11.895715	13.490130	11.528786	11.917480	11.057190	11.4	
0x0000	000000216080	14.219353	12.578764	12.981959	11.614140	11.367486	11.236432	10.8	
0x0000	30000002160A0	12.162021	12.395756	12.909150	10.340583	11.732928	11.289549	13.2	
0x0000	30000002160C0	13.021592	13.032330	11.214454	10.869614	14.484652	11.907482	15.5	
0x0000	30000002160E0	14.306890	13.693730	14.891927	13.105434	13.241500	12.827445	14.5 -	
٠ 📃			III					P.	
matrix	Mul_kernel.cu ×							-	
(Gl	obal Scope)			👻 💷 🕶	trixMul(float * C, float	* A, float * B, int wA, i	int wB)	-	
-	syncthrea	ds().		acton				÷	
	}	us();						*	
	<pre>// Write the bl // each thread int c = wB * BL C[c + wB * ty + }</pre>	<pre>lock sub-matrix writes one elem .OCK_SIZE * by tx] = Csub;</pre>	to device memory; ment ⊦ BLOCK_SIZE * bx;						
L	#endif // #ifndef _	MATRIXMUL_KERN	EL_H_					-	
100 %	- ₹							÷.	

Switching Between Threads





Conditional Breakpoints





// Synchronize to make sure the matrices are

Data Breakpoints









- 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



Seco	onds	0.399	61 0.40061 0	40161 0.40261 (0.40361 0.40461	0.40561 0.40661	0.40761 0.40	861 0.40961	0.41051 0.411	61 0.41261 0	41361 0.41461	0.41561	0.41661 0.41761
Processes													
 nexusgtcdemo.exe [2832] 				-								-	-
Thread 91.2% [1204]		1											
Thread State	Y.												
Function Calls	Y	Level 0 Status	cuM		Swa			cuMemo	cpyHtoD_v2				
Tools Extension	Y	Markers											
Push / Pop Ranges	∇	Level 0 Level 1	Uplo Comp	ute GL Rer	ndering			Uploa	ad Image			(o GL Render
 Thread 4.8% [1716] 		L											
 Thread 2.4% [2116] 		L											
Thread [2488]		1											
CUDA													
Context 1 [0]													
Driver API	Y.		cuM	10				cuMem	cpyHtoD_v2				
Memory	Y												
 Compute 	Y			Sobel	kernel								Sobel ke
8.9% [67] kernel	Y				kernel								ke
3.7% [68] SobelTex	Y			Sobel									Sobel
0.0% [67] memset32_aligne	ed 🍸												
 Streams 													
Stream 0	Y			Sobel	kernel								Sobel ke
 Counters 													
OpenGL													
Render Context 0x20000													
 Device Context 0xfffffffa001 	0 🍸	CPU		207					208				
API Calls	Y	Level 0			Swa								
Draw Calls [Device 0x100]	Y	13	10 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 -	_		_			+++			1 ¹¹	_
Tools Extension													
[2] FPS Markers	Y	Markers											
Container.													

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) available for purchase in December
Compute Debugger	\checkmark	\checkmark
DirectX 10 & 11 Debugger & Graphics inspector	\checkmark	✓
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	✓	\checkmark
Compute Analyzer		✓
OpenGL and OpenCL Analyzer		\checkmark
DirectX 10 & 11 Analyzer		\checkmark
Tesla Compute Cluster (TCC) Debugging		\checkmark

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: <u>http://developer.nvidia.com/ParallelNsight</u>



developer.nvidia.com/ParallelNsight<



Made for 00 Visual Studio

NSIGHT

PARAL

FI Power of GPU Computing, Simplicity of Visual Studio

Free Download Videos Documentation