

## CSE 564: Visualization

### CUDA Programming Environment

Klaus Mueller

Computer Science Department

Stony Brook University

## Setup CUDA

### Compute Unified Device Architecture

- Check hardware compatibility: [http://www.nvidia.com/object/cuda\\_gpus.html](http://www.nvidia.com/object/cuda_gpus.html)
  - Driver, Toolkit (3.2) and SDK [http://www.nvidia.com/object/cuda\\_get.html](http://www.nvidia.com/object/cuda_get.html)
  - Toolkit includes:
    - Compiler
    - Development tools
    - Libraries for scientific computation (CUBLAS, CUFFT, CUSPARSE, CURAND, etc.)
    - User guides and documents
- 

## Compilation and Linking

Any source file containing CUDA language extensions must be compiled with NVCC

NVCC is a compiler driver

- Works by invoking all the necessary tools and compilers like cudacc, g++, cl, ...

Any executable with CUDA code requires two dynamic libraries:

- The CUDA runtime library ([cudart](#))
  - The CUDA core library ([cuda](#))
- 

## Setup CUDA in Visual Studio

Use pre-defined build rules (simpler)

- invokes `nvcc` on the file and forwards the C/C++ options to `cl.exe`.

or Write compilation command manually

Modified from existing project

or Use CUDA Visual Studio Wizard

- <http://sourceforge.net/projects/cudavswizard>

Setup Visual Studio syntax highlighting (inside CUDA SDK)

---

## Debugging Using the Device Emulation Mode (Before CUDA 2.3)

An executable compiled in **device emulation mode** (`nvcc -deviceemu`) runs completely on the host using the CUDA runtime

- No need of any device and CUDA driver
- Each device thread is emulated with a host thread

Running in device emulation mode, one can:

- Use host native debug support (breakpoints, inspection, etc.)
- Access any device-specific data from host code and vice-versa
- Call any host function from device code (e.g. `printf`) and vice-versa
- Detect deadlock situations caused by improper usage of `__syncthreads`

## Device Emulation Mode Pitfalls (Before CUDA 2.3)

Emulated device threads execute sequentially, so **simultaneous accesses of the same memory location by multiple threads** could produce different results.

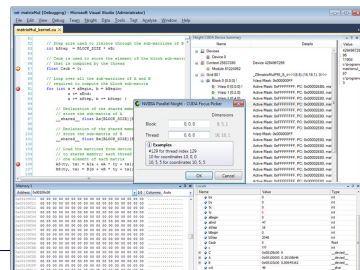
**Dereferencing device pointers** on the host or host pointers on the device can produce correct results in device emulation mode, but will generate an error in device execution mode

## New Debugging Tools (After CUDA 2.3)

Parallel Nsight (Windows 7 and Vista)

- Visual Studio Based GPU Development Environment  
<http://developer.nvidia.com/object/nsight.html>
- Debug CUDA C/C++ source code directly on the GPU
- Use the familiar Visual Studio Locals, Watches, Memory and Breakpoints windows
- Integrated analysis tool to isolate performance bottleneck
- Require 2 GPUs

CUDA-gdb for Linux



## Visual Profiler

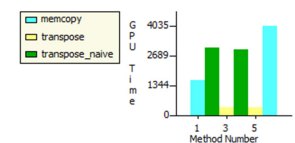
Method	GPU Time	CPU Time	Occupancy	grid size X	grid size Y	block size X	block size Y	block size Z	static shared memory per block	registers per thread	mem transfer size (bytes)	ran	gl coalesced	glt uncoalesced	glt coalesced	branch	instructions
1 memcopy	1573.92	4390.85									4194304	0					
2 transpose_naive	3041.95	3062.27	1	256	16	16	16	16	32	6		8192	262144	0	2048	32591	
3 transpose	341.92	364.64	1	256	16	16	16	16	1120	8		8192	0	32768	10240	42873	
4 transpose_naive	2945.89	2959.24	1	256	16	16	16	16	32	6		8192	262144	0	2048	32479	
5 transpose	343.808	356.759	1	256	16	16	16	16	1120	8		8192	0	32768	10240	43116	
6 memcopy	4039.91	4880.18									4194304	1					

A graphical profiling tool to measure and benchmark performance

tracks events with hardware counters on signals in the chip

Fine Tuning Performance by watching the following metric

- Coalescing
- Occupancy
- Branch diversity
- Instruction throughput
- Computing / Data transfer ratio
- Share memory and register per thread



## Other Tools

CUDA-MemCheck (Linux)  
out-of-bound and misaligned accesses

CUDA occupancy calculator

A spreadsheets calculating multiprocessor occupancy of a GPU (the ratio of active warps to the maximum number of warps supported)

## CUDA Libraries

CUBLAS (BLAS = Basic Linear Algebra Subprograms)

level1 (scalar, vector, vector-vector)

level2 (matrix-vector) , level3 (matrix-matrix)

```
void cublasSsymv(char uplo, int n, float alpha, const float *A, int lda, const float *x, int incx, float beta, float *y, int incy)
```

performs the matrix-vector operation where alpha and beta are single-precision scalars, and x and y are n-element single-precision vectors. A is a symmetric n\*n matrix that consists of single-precision elements and is stored in either upper or lower storage mode

CUFFT

- Similar “plan” configuration as in FFTW
- 1d, 2d, 3d transforms of complex and real data
- Batched execution of multiple 1D transform
- Both in-place and out-of-place transforms

## CUBLAS Example

Compute a vector’s L2 norm

$$\|x\| := \sqrt{x_1^2 + \dots + x_n^2}$$

- Single precision

```
float cublasSnrm2 (int n, const float *x, int incx)
```

- Double precision

```
double cublasDnrm2 (int n, const double *x, int incx)
```

```
cublasInit();  
float *h_A;  
h_A = (float*)malloc(n * sizeof(h_A[0]));  
...  
cublasAlloc(n, sizeof(d_A[0]), (void*)&d_A);  
cublasSetVector(n, sizeof(h_A[0]), h_A, 1, d_A, 1);  
float norm2result=cublasSnrm2 (n, const float *x, 1);  
cublasFree(d_A); free(h_A);  
cublasShutdown();
```

initialize library

initialize vector

data transfer

compute norm

Wrap-up

## CUDA Libraries (3<sup>rd</sup> party)

MAGMA (porting from LAPACK to GPU+multicore architectures)

CULA (3<sup>rd</sup> party implementation of LAPACK)

PyCUDA (CUDA via Python)

Thrust (C++ template for CUDA, open source)

Jasper for DWT (Discrete wavelet transform)

OpenViDIA for computer vision

CUDPP for radix sort

## To Probe Further

### NVIDIA CUDA Zone:

- [http://www.nvidia.com/object/cuda\\_home.html](http://www.nvidia.com/object/cuda_home.html)
- Lots of information and code examples
- NVIDIA CUDA Programming Guide

### GPGPU community:

- <http://www.gpgpu.org>
  - User forums, tutorials, papers
  - Good source: conference tutorials  
<http://www.gpgpu.org/developer/index.shtml#conference-tutorial>
-