MIC-GPU: High-Performance Computing for Medical Imaging on Programmable Graphics Hardware (GPUs)

CUDA Programming Environment

Klaus Mueller and Sungsoo Ha

Stony Brook University

Computer Science

Stony Brook, NY

Setup CUDA

Compute Unified Device Architecture

- Check hardware compatibility: http://www.nvidia.com/object/cuda_gpus.html
- Driver, Toolkit (4.0) and SDK 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

- Compile device code
- Invoking the necessary compilers for host code like, g++, cl, ...

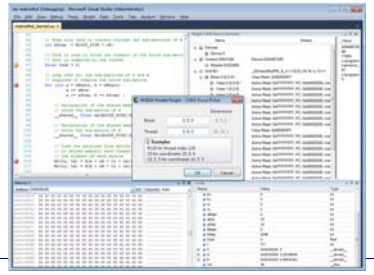
Any executable with CUDA code requires dynamic libraries:

- The CUDA runtime library (**`cudart`**) OR
- The CUDA core library (**`cuda`**)

Development Tools

Parallel Nsight (Windows)

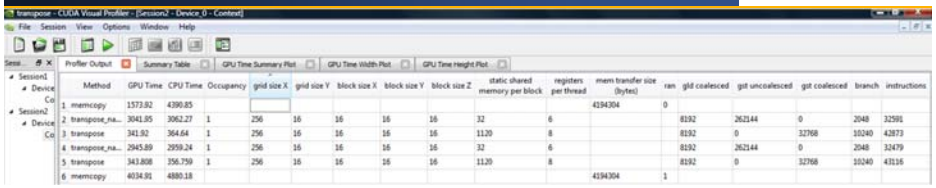
- Visual Studio Based GPU Development Environment
  http://developer.nvidia.com/object/nsight.html
- Debug CUDA C/C++ source code directly on the GPU
- Use the familiar Visual Studio Locals, Watches, Memory and Breakpoints windows
- Integrated analysis tool to isolate performance bottleneck



CUDA-GDB debugger

for Linux and MacOS

Visual Profiler

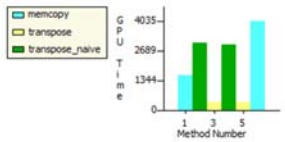


A graphical profiling tool to measure and benchmark performance

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



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

CUBLAS Example

Compute a vector's L2 norm $\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$

- Single precision

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

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

```
cublasInit();                                                  initialize library
float *h_A;
h_A = (float*)malloc(n * sizeof(h_A[0]));                      initialize vector
…
cublasAlloc(n, sizeof(d_A[0]), (void**)&d_A);                  data transfer
cublasSetVector(n, sizeof(h_A[0]), h_A, 1, d_A, 1);
float norm2result=cublasSnrm2 (n, d_A, 1);                     compute norm
cublasFree(d_A); free(h_A);
cublasShutdown();                                              Wrap-up
```


CUDA Libraries (3rd party)

MAGMA (porting from LAPACK to GPU+multicore architectures)

CULA (3rd 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

Thrust: Introduction

SPIE Medical Imaging

Offers

- STL compatible containers (vector, list, map)
- ~50 algorithm (reduction, prefix sum, sorting)
- Rapid prototyping

Container

- Hides cudaMalloc & cudaMemcpy
- Iterators behave like pointer

Thrust Example: Sorting

SPIE Medical Imaging

generate 16M random numbers on the host

```
thrust::host_vector<int> h_vec(16*1024*1024);

thrust::generate(h_vec.begin(), h_vec.end(), rand);

thrust::device_vector<int> d_vec = h_vec;

thrust::sort(d_vec.begin(), d_vec.end());

thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());
```

transfer data to the device

sort data on the device

transfer data back to host

SPIE Medical Imaging 2013

Thrust: Operators

SPIE Medical Imaging

```
thrust::device_vector<int>   i_vec = ...
thrust:: device_vector<float> f_vec = ...

thrust:: reduce(i_vec.begin(), i_vec.end());
thrust:: reduce(i_vec.begin(), i_vec.end(),    0, thrust:: plus<int>());

thrust:: reduce(f_vec.begin(), f_vec.end());
thrust:: reduce(f_vec.begin(), f_vec.end(), 0.0f, thrust:: plus<float>());

thrust:: reduce(i_vec.begin(), i_vec.end(),    0, thrust:: maximum<int>());
```

declare storage

sum of integers (equivalent calls)

sum of floats (equivalent calls)

maximum of integers

SPIE Medical Imaging 2013

Thrust Example: Vector L2 Norm

SPIE Medical Imaging

More like C++

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

```
template <typename T> struct square
{ __host__ __device__
T operator()(const T& x) const {
        return x * x;       }
};
 square<float>      unary_op;
plus<float> binary_op;
float init = 0;

device_vector<float>  A(3);
A[0] = 20;  A[1] = 30;  A[2] = 40;
float norm = sqrt( transform_reduce(A.begin(), A.end(),  unary_op, init, binary_op));
```

define transformation f(x) -> x^2

setup arguments

initialize vector

compute norm

SPIE Medical Imaging 2013

To Probe Further

NVIDIA CUDA Zone:

- 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