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, 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
- 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
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

**CUBLAS Example**

Compute a vector’s L2 norm:

\[ ||x|| := \sqrt{x_1^2 + \cdots + x_n^2} \]

**Single precision**

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

**Double precision**

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

**CUDA Libraries**

- **CUBLAS (BLAS = Basic Linear Algebra Subprograms)**
  - Level 1 (scalar, vector, vector-vector)
  - Level 2 (matrix-vector)
  - Level 3 (matrix-matrix)

  ```c
  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 \( n \times n \) matrix that consists of single-precision elements and is stored in either upper or lower storage mode.

- **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

Offers
• STL compatible containers (vector, list, map)
• ~50 algorithms (reduction, prefix sum, sorting)
• Rapid prototyping

Container
• Hides cudaMalloc & cudaMemcpy
• Iterators behave like pointer

Thrust Example: Sorting

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());

generate 16M random numbers on the host
transfer data to the device
sort data on the device
transfer data back to host

Thrust Example: Vector L2 Norm

More like C++

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);


float norm = sqrt(
transform_reduce(A.begin(), A.end(), unary_op, init, binary_op));

define transformation \( f(x) \rightarrow x^2 \)
setup arguments
initialize vector
compute norm

Thrust: Operators

thrust::device_vector<int>   i_vec = ...

thrust:: device_vector<float> f_vec = ...

thrust:: reduce(i_vec.begin(), i_vec.end());

thrust:: reduce(f_vec.begin(), f_vec.end());

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

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>());

maximum of integers
To Probe Further

NVIDIA CUDA Zone:
- Lots of information and code examples
- NVIDIA CUDA Programming Guide

GPGPU community:
- [http://www.gpgpu.org](http://www.gpgpu.org)
- User forums, tutorials, papers

Course Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:30 – 1:45</td>
<td>Introduction</td>
</tr>
<tr>
<td>1:45 – 2:00</td>
<td>Parallel programming primer</td>
</tr>
<tr>
<td>2:00 – 2:15</td>
<td>GPU hardware</td>
</tr>
<tr>
<td>2:15 – 3:00</td>
<td>CUDA API, threads level optimization</td>
</tr>
<tr>
<td></td>
<td>Coffee Break</td>
</tr>
<tr>
<td>3:30 – 4:00</td>
<td>CUDA memory optimization</td>
</tr>
<tr>
<td>4:00 – 4:15</td>
<td>CUDA programming environment</td>
</tr>
<tr>
<td>4:15 – 4:45</td>
<td>Parallelism in medical image (Klaus)</td>
</tr>
<tr>
<td>4:45 – 5:25</td>
<td>CT reconstruction examples (Eric + Ziyi)</td>
</tr>
<tr>
<td>5:25 – 5:30</td>
<td>Closing remarks (Klaus)</td>
</tr>
</tbody>
</table>