Lecture 5
Libraries and tools

Prof Wes Armour

wes.armour@eng.ox.ac.uk

Oxford e-Research Centre

Department of Engineering Science

Learning outcomes

In this fifth lecture we will learn about GPU libraries and tools for GPU programming.

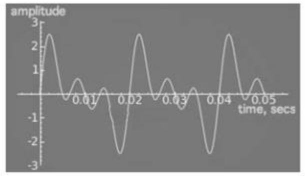
You will learn about:

- NVIDIA GPU libraries and there usefulness in scientific computing.
- Third party libraries.
- Directives based approaches to GPU computation.
- Tools for GPU programming.

CUDA Libraries and tools

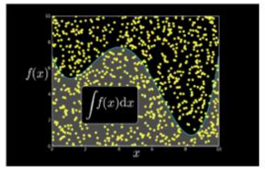
NVIDIA provides a rich ecosystem of software tools that allow you to easily utilise GPUs in your projects.

Many of the libraries and tools are now (fairly) mature and are feature rich.



cuFFT

GPU-accelerated library for Fast Fourier Transforms

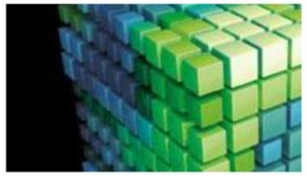


cuRAND

GPU-accelerated random number generation (RNG)

CUDA Toolkit

Provides a comprehensive environment for C/C++ developers building GPU-accelerated applications.



cuBLAS

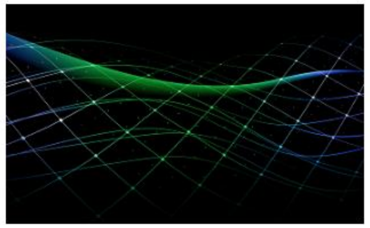
GPU-accelerated standard BLAS library

CUDA Math Library

The CUDA math library provides core (GPU accelerated) functionality to many of the other CUDA libraries that you might want to use in your projects.

The library provides all of the standard math functions you would expect (i.e. very similar to what you would get from Intel).

- Various exponential and log functions
- Trigonometric functions and their inverses
- Hyperbolic functions and their inverses
- Error functions and their inverses
- Bessel and Gamma functions
- Vector norms and reciprocals (esp. for graphics)
- Mainly single and double precision – although there is a growing number of functions which support half precision.



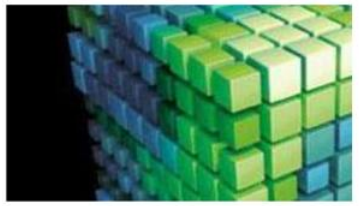
CUDA Math Library

GPU-accelerated standard mathematical function library

cuBLAS Library

cuBLAS is a GPU accelerated library that provides basic linear algebra subroutines for dense matrices.

- It includes matrix-vector and matrix-matrix product.
- Significant input from Vasily Volkov at UC Berkeley; one routine contributed by Jonathan Hogg from RAL.
- It is possible to call cuBLAS routines from user kernels – device API (might have been removed from CUDA 10).
- Some support for a single routine call to do a “batch” of smaller matrix-matrix multiplications.
- Also support for using CUDA streams to do a large number of small tasks concurrently.



cuBLAS

GPU-accelerated standard BLAS library

cuBLAS Library

cuBLAS is used by calling routines from your host code.

Helper routines:

- Memory allocation.
- Data copying from CPU to GPU, and vice versa.
- Error reporting.

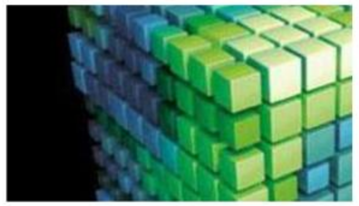
Compute routines:

- matrix-matrix and matrix-vector product.
- **Warning!** Some calls are asynchronous, i.e. the call starts the operation but the host code then continues before it has completed!

CUDA 10 provides a new lightweight library dedicated to GEneral Matrix-to-matrix Multiply (GEMM), called cuBLASLt.

The `simpleCUBLAS` example in SDK is a good example code.

The cuBLASxt library extends cuBLAS to multiple GPUs.



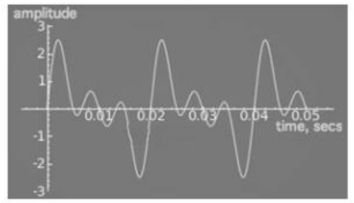
cuBLAS

GPU-accelerated standard BLAS library

cuFFT Library

cuFFT is a GPU accelerated library that provides Fast Fourier Transforms.

- Provides 1D, 2D and 3D FFTs.
- Significant input from Satoshi Matsuoka and others at Tokyo Institute of Technology.
- Has almost all of the variations found in FFTW and other CPU libraries.
- Includes the cuFFTW library, a porting tool, to enable users of FFTW to start using GPUs with minimal effort.
- Plans for device level functionality. *If this is something of interest ask Karel – he has already produced shared memory device level FFTs for our projects.*



cuFFT

GPU-accelerated library for Fast Fourier Transforms

cuFFT Library

cuFFT is used exactly like cuBLAS - it has a set of routines called by host code:

Helper routines include “plan” construction.

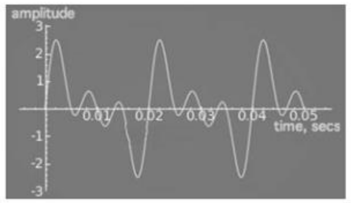
Compute routines perform 1D, 2D, 3D FFTs:

- `cufftExecC2C()` - complex-to-complex.
- `cufftExecR2C()` - real-to-complex.
- `cufftExecC2R()` - complex-to-real inverse transform.

(double precision routines have different function calls, e.g. `cufftExecZ2Z()`)

It supports doing a “batch” of independent transforms, e.g. applying 1D transform to a 3D dataset

The `simpleCUFFT` example in SDK is a good starting point.



cuFFT

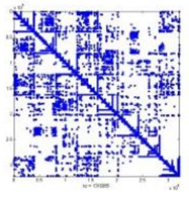
GPU-accelerated library for Fast Fourier Transforms

https://docs.nvidia.com/cuda/cufft/index.html#introduction

cuSPARSE Library

cuSPARSE is a GPU accelerated library that provides various routines to work with sparse matrices.

- Includes sparse matrix-vector and matrix-matrix products.
- Can be used for iterative solution (but see cuSOLVER for an easy life).
- Also has solution of sparse triangular system
- Note: batched tridiagonal solver is in cuBLAS not cuSPARSE
- Contribution from István Reguly (Oxford)



cuSPARSE

GPU-accelerated BLAS for sparse matrices

cuRAND Library

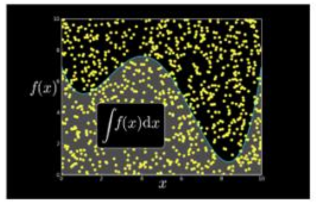
cuRAND is a GPU accelerated library for random number generation.

It has many different algorithms for pseudorandom and quasi-random number generation.

Pseudo: XORWOW, mrg32k3a, Mersenne Twister and Philox 4x32_10
Quasi: SOBOL and Scrambled SOBOL

Uniform, Normal, log-Normal and Poisson outputs

This library also includes device level routines for RNG within user kernels.



cuRAND

GPU-accelerated random number generation (RNG)

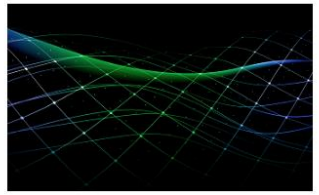
cuSOLVER

cuSOLVER brings together cuSPARSE and cuBLAS.

Has solvers for dense and sparse systems.

Key LAPACK dense solvers, 3 – 6x faster than MKL.

Sparse direct solvers, 2–14x faster than CPU equivalents.



cuSOLVER

Dense and sparse direct solvers for Computer Vision, CFD, Computational Chemistry, and Linear Optimization applications

Other notable libraries

CUB (CUDA Unbound): https://nvlabs.github.io/cub/

- Provides a collection of basic building blocks at three levels: device, thread block, warp.
- Functions include sort, scan and reduction.
- Thrust uses CUB for CUDA versions of key algorithms.

http://on-demand.gputechconf.com/gtc/2014/presentations/S4566-cub-collective-software-primitives.pdf

AmgX

GPU accelerated linear solvers for simulations and implicit unstructured methods

AmgX (originally named NVAMG): http://developer.nvidia.com/amgx

- Library for algebraic multigrid

Other notable libraries

cuDNN

- Library for Deep Neural Networks
- Some parts developed by Jeremy Appleyard (NVIDIA) working in Oxford

nvGraph

- Page Rank, Single Source Shortest Path, Single Source Widest Path

NPP (NVIDIA Performance Primitives)

- Library for imaging and video processing
- Includes functions for filtering, JPEG decoding, etc.

CUDA Video Decoder API…

cuDNN

GPU-accelerated library of primitives for deep neural networks

nvGRAPH

GPU-accelerated library for graph analytics

NVIDIA Performance Primitives

GPU-accelerated library for image and signal processing

Thrust

Thrust is a high-level C++ template library with an interface based on the C++ Standard Template Library (STL).

Thrust has a very different philosophy to other libraries - users write standard C++ code (*no CUDA*) but get the benefits of GPU acceleration.

Thrust relies on C++ object-oriented programming – certain objects exist on the GPU, and operations involving them are implicitly performed on the GPU.

It has lots of built-in functions for operations like sort and scan.

It also simplifies memory management and data movement.

Thrust

GPU-accelerated library of parallel algorithms and data structures

Kokkos

Kokkos is another high-level C++ template library, similar to Thrust.

It has been developed in the US DoE Labs, so there is considerable investment in both capabilities and on-going software maintenance.

Could be worth investigating if you are considering using Thrust in your projects.

For more information see
https://github.com/kokkos/kokkos/wiki
https://trilinos.org/packages/kokkos/

MAGMA

MAGMA (Matrix Algebra on GPU and Multicore Architectures) has been available for a few years (See nice SC17 handout: http://www.icl.utk.edu/files/print/2017/magma-sc17.pdf)

- LAPACK for GPUs – higher level numerical linear algebra, layered on top of cuBLAS.
- Open source – freely available.
- Developed by Jack Dongarra, Jim Demmel and others.

ArrayFire

Originally a commercial software (from Accelereyes), but is now open source.

- Supports both CUDA and OpenCL execution.
- C, C++ and Fortran interfaces.
- Wide range of functionality including linear algebra, image and signal processing, random number generation, sorting…

www.accelereyes.com/products/arrayfire

A final word on libraries

NVIDIA maintains webpages with links to a variety of CUDA libraries:

www.developer.nvidia.com/gpu-accelerated-libraries

and other tools:

www.developer.nvidia.com/tools-ecosystem

The seven dwarfs

Phil Colella a senior researcher at Lawrence Berkeley National Laboratory, talked about "*7 dwarfs*" of numerical computation in 2004.

Expanded to 13 by a group of UC Berkeley professors in a 2006 report: "A View from Berkeley" www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

These 13 dwarfs define key algorithmic kernels in many scientific computing applications.

They have been very helpful to focus attention on HPC challenges and development of libraries and problem-solving environments/frameworks.

The seven dwarfs

1. Dense linear algebra
2. Sparse linear algebra
3. Spectral methods
4. N-body methods
5. Structured grids
6. Unstructured grids
7. Monte Carlo

1. Dense Linear Algebra

Many tools available, some from NVIDIA, some third party:

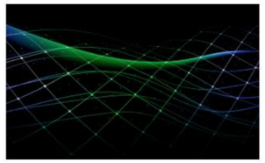
- cuBLAS
- cuSOLVER
- MAGMA
- ArrayFire

CUTLASS, an NVIDIA tool for Fast Linear Algebra in CUDA C++ might also be worth a look if you can't use the above libraries for any reason.

https://devblogs.nvidia.com/cutlass-linear-algebra-cuda/

cuBLAS

GPU-accelerated standard BLAS library



cuSOLVER

Dense and sparse direct solvers for Computer Vision, CFD, Computational Chemistry, and Linear Optimization applications

MAGMA

{} ARRAYFIRE

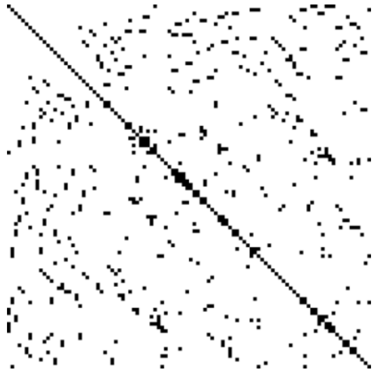
2. Sparse Linear Algebra

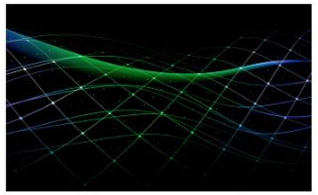
Iterative solvers

- Some available in PetSc - https://www.mcs.anl.gov/petsc/
- Others can be implemented using sparse matrix-vector multiplication from cuSPARSE.
- NVIDIA has AmgX, an algebraic multigrid library.

Direct solvers

- NVIDIA's cuSOLVER.
- SuperLU project at University of Florida (Tim Davis) www.cise.ufl.edu/davis/publications_files/qrgpu_paper.pdf
- Project at RAL (Jennifer Scott & Jonathan Hogg) https://epubs.stfc.ac.uk/work/12189719





cuSOLVER

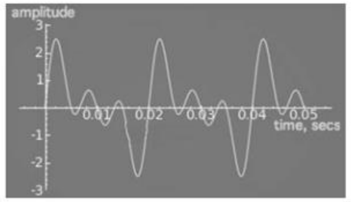
Dense and sparse direct solvers for Computer Vision, CFD, Computational Chemistry, and Linear Optimization applications

3. Spectral methods

cuFFT

Library provided / maintained by NVIDIA

For those interested in FFTs on GPUs – ask karel…



cuFFT

GPU-accelerated library for Fast Fourier Transforms

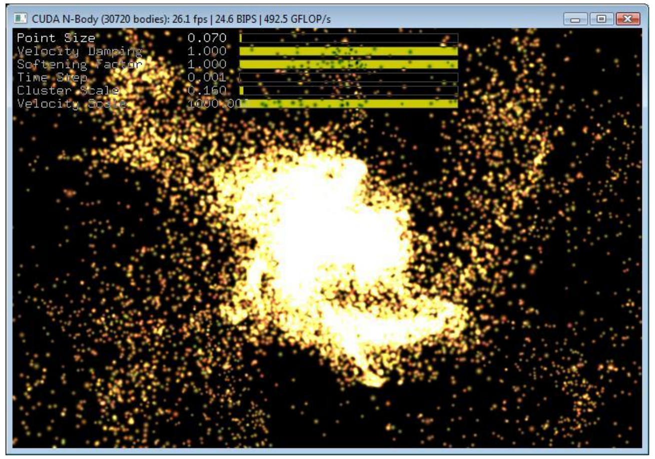
4. N-Body methods

OpenMM:
- http://openmm.org/
- open source package to support molecular modelling, developed at Stanford.

Fast multipole methods:
- ExaFMM by Yokota and Barba: http://www.bu.edu/exafmm/

- FMM2D by Holm, Engblom, Goude, Holmgren: http://user.it.uu.se/~stefane/freeware

- Software by Takahashi, Cecka, Fong, Darve: http://onlinelibrary.wiley.com/doi/10.1002/nme.3240/pdf


CUDA N-Body (30720 bodies): 26.1 fps | 24.6 BIPS | 492.5 GFLOP/s

```
Point Size          0.070
Velocity Damping    1.000
Softening Factor    1.000
Time Step           0.001
Cluster Scale       0.160
Velocity Scale      1000.000
```

https://docs.nvidia.com/cuda/cuda-samples/index.html#cuda-n-body-simulation

https://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/nbody/doc/nbody_gems3_ch31.pdf

5. Structured grids

Lots of people have developed one-off applications.

No great need for a library for single block codes (though possible improvements from “tiling”?).

Multi-block codes could benefit from a general-purpose library, mainly for MPI communication.

Oxford OPS project has developed a high-level open-source framework for multi-block codes,
using GPUs for code execution and MPI for distributed-memory message-passing.

All implementation details are hidden from “users”, so they don’t have to know about GPU/MPI programming.

For those interested – ask Istvan/Mike…

6. Unstructured grids

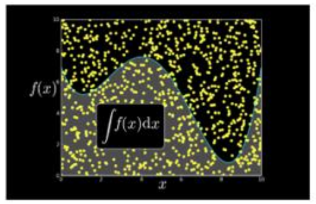
In addition to GPU implementations of specific codes there are projects to create high-level solutions which others can use for their application codes:

- Alonso, Darve and others (Stanford).
- Oxford / Imperial College project developed OP2, a general-purpose open-source framework based on a previous framework built on MPI.

May be other work, again ask Istvan/Mike…

7. Monte Carlo methods

- NVIDIA cuRAND library.
- ArrayFire library.
- Some examples in CUDA SDK distribution.
- Nothing else needed except for more output distributions?



cuRAND

GPU-accelerated random number generation (RNG)

ArrayFire

Tools - Debugging

`cuda-memcheck`

A command line tool that detects array out-of-bounds errors, and mis-aligned device memory accesses – very useful because such errors can be tough to track down otherwise.

`cuda-memcheck --tool racecheck`

This checks for shared memory race conditions:

- Write-After-Write (WAW): two threads write data to the same memory location but the order is uncertain.
- Read-After-Write (RAW) and Write-After-Read (WAR): one thread writes and another reads, but the order is uncertain.

`cuda-memcheck --tool initcheck`

This detects the reading of uninitialised device memory.

CUDA-MEMCHECK

Identifies memory access errors in your GPU code and allows you to locate and resolve problems quickly. CUDA-MEMCHECK also reports runtime execution errors, identifying situations that could result in an "unspecified launch failure" error while your application is running.

Tools – Other Languages

FORTRAN: PGI (Portland Group) CUDA FORTRAN compiler with natural FORTRAN equivalent to CUDA C; also IBM FORTRAN XL for new DoE systems.

MATLAB: can call kernels directly, or use OOP like Thrust to define MATLAB objects which live on the GPU
http://www.oerc.ox.ac.uk/projects/cuda-centre-excellence/matlab-gpus

Mathematica: similar to MATLAB?

Python: http://mathema.tician.de/software/pycuda
https://store.continuum.io/cshop/accelerate/

R: http://www.fuzzyl.com/products/gpu-analytics/
http://cran.r-project.org/web/views/HighPerformanceComputing.html

Haskell: https://hackage.haskell.org/package/cuda
http://hackage.haskell.org/package/accelerate

Tools – Directive based

OpenACC - “More Science, Less Programming”:
This is like Thrust, it aims to hide CUDA programming by doing everything in the top-level CPU code.

A programmer takes standard C/C++/Fortran code and inserts pragmas saying what can be done in parallel and where data should be located: https://www.openacc.org/

OpenMP 4.0 is similar but newer:
This has been strongly pushed by Intel to accommodate Xeon Phi.
My feeling is, this is a good way to use a directives based approach, OpenMP has significant support and I believe GPU integration will mature over the coming years.
http://on-demand.gputechconf.com/gtc/2016/presentation/s6510-jeff-larkin-targeting-gpus-openmp.pdf

Tools

CUDA Toolkit

Provides a comprehensive environment for C/C++ developers building GPU-accelerated applications.

OpenACC

Directives for parallel computing, is a new open parallel programming standard designed to enable all scientific and technical programmers.

PGI Accelerator Fortran and C Compilers

Accelerate applications on GPU platforms by adding compiler directives to existing code.

The PGI CUDA C/C++ compiler for x86

Compile and optimize their CUDA applications to run on x86-based workstations, servers and clusters.

CUDA FORTRAN

Enjoy GPU acceleration directly from your Fortran program using CUDA Fortran from The Portland Group.

Anaconda Accelerate

Enables acceleration on your GPU or multi-core processor using Python.

PyCUDA

Gives you access to CUDA fuctionality from your Python code.

Altimesh Hybridizer™

An advanced productivity tool that generates vectorized C++ (AVX) and CUDA C code from .NET assemblies (MSIL) or Java archives (bytecode)

OpenCL™

OpenCL is a low-level API for GPU computing that can run on CUDA-powered GPUs.

Alea GPU

This is a novel approach to develop GPU applications on .NET, combining the CUDA with Microsoft's F#.

Tools - IDEs

Integrated Development Environments (IDE):

Nsight Visual Studio edition – NVIDIA plug-in for Microsoft Visual Studio
http://developer.nvidia.com/nvidia-nsight-visual-studio-edition

Nsight Eclipse edition – IDE for Linux systems
http://developer.nvidia.com/nsight-eclipse-edition

these come with editor, debugger, profiler integration

NVIDIA® Nsight™

The ultimate development platform for heterogeneous computing. Work with powerful debugging and profiling tools, optimize the performance of your CPU and GPU code. Find out about the Ecilipse Edition and the graphics debugging enabled Visual Studio Edition.

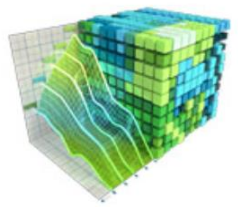
Tools - Profiling

NVIDIA Visual Profiler `nvprof`:

- This is a standalone piece of software for Linux and Windows systems.
- It uses hardware counters to collect a lot of useful information.
- Lots of things can be measured, but the limited number of counters means that, for some larger applications, it runs the application multiple to gather necessary information.
- It's also possible to use `nvprof` from the command line:

```
nvprof --metrics "flops sp,flops dp" prac2
```

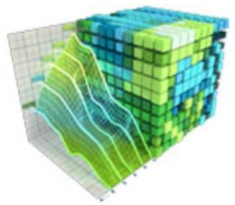
- This can be useful if you want to profile on a machine that you don't have a graphical interface to.
- Do `nvprof --help` for more info on other options.



NVIDIA Visual Profiler

This is a cross-platform performance profiling tool that delivers developers vital feedback for optimizing CUDA C/C++ applications. First introduced in 2008, Visual Profiler supports all CUDA capable NVIDIA GPUs shipped since 2006 on Linux, Mac OS X, and Windows.

Tools



NVIDIA Visual Profiler

This is a cross-platform performance profiling tool that delivers developers vital feedback for optimizing CUDA C/C++ applications. First introduced in 2008, Visual Profiler supports all CUDA capable NVIDIA GPUs shipped since 2006 on Linux, Mac OS X, and Windows.

NVIDIA® Nsight™

The ultimate development platform for heterogeneous computing. Work with powerful debugging and profiling tools, optimize the performance of your CPU and GPU code. Find out about the Ecilipse Edition and the graphics debugging enabled Visual Studio Edition.

CUDA-GDB

Delivers a seamless debugging experience that allows you to debug both the CPU and GPU portions of your application simultaneously. Use CUDA-GDB on Linux or MacOS, from the command line, DDD or EMACS.

CUDA-MEMCHECK

Identifies memory access errors in your GPU code and allows you to locate and resolve problems quickly. CUDA-MEMCHECK also reports runtime execution errors, identifying situations that could result in an "unspecified launch failure" error while your application is running.

What have we learnt?

In this lecture we've looked at the wide software ecosystem that now surrounds GPU computing and how that can be used to make your life as a programmer easier.

We've looked at directives based approaches and how these are useful.

Finally we've looked at tools that allow us to develop CUDA code in an easy and maintainable way.