Introduction to CUDA Programming

Lecture 4: libraries and tools

高性能计算机研究中心

CUDA libraries

Originally, NVIDIA planned to provide only one or two maths libraries, but over time these have steadily increased

- **CUBLAS**
  - basic linear algebra subroutines for dense matrices
  - includes matrix-vector and matrix-matrix product
  - significant input from Vasily Volkov at UC Berkeley; one routine contributed by Jonathan Hogg from RAL
  - with dynamic parallelism on Kepler, it is now possible to call CUBLAS routines from user kernels
  - 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

CUDA libraries

- **CUBLAS is a set of routines to be called by user host code:**
  - **helper routines:**
    - **memory allocation**
    - **data copying from CPU to GPU, and vice versa**
    - **error reporting**
  - **compute routines:**
    - **matrix-matrix product**
    - **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**

- simpleCUBLAS **example in SDK is a good example code**

CUDA libraries

- CUFFT
  - Fast Fourier Transform
  - 1D, 2D, 3D
  - significant input from Satoshi Matsuoka and others at Tokyo Institute of Technology
    www.voltaire.com/assets/files/Casestudies/titechcasestudyfinalforSC08.pdf
  - has almost all of the variations found in FFTW and other CPU libraries?
  - improved performance with Kepler hardware due to new warp shuffle instructions

CUDA libraries

- **Like CUBLAS, it is a set of routines called by user host code:**
  - **helper routines include “plan” construction**
  - **compute routines perform 1D, 2D, 3D FFTs**
  - **it supports doing a “batch” of independent transforms, e.g. applying 1D transform to a 3D dataset**
  - simpleCUFFT **example in SDK**

CUDA libraries

- CUSPARSE
  - various routines to work with sparse matrices
  - includes sparse matrix-vector and matrix-matrix products
  - could be used for iterative solution
  - also has solution of sparse triangular system
  - note: batched tridiagonal solver is in CUBLAS not CUSPARSE
  - contribution from István Reguly (Oxford)

CUDA libraries

- **CURAND**
  - **random number generation**
  - **XORWOW,** mrg32k3a, **Mersenne Twister and** Philox_4x32_10 **pseudo-random generators**
  - **Sobol quasi-random generator (with optimal scrambling)**
  - **uniform, Normal, log-Normal, Poisson outputs**

- **CUB**
  - **provides a collection of basic building blocks at three levels: device, thread block, warp**
  - **functions include sort, scan, reduction**
  - **Thrust uses CUB for CUDA version of key algorithms**

CUDA libraries

- NPP (NVIDIA Performance Primitives)
  - library for imaging and video processing
  - includes functions for filtering, JPEG decoding, etc.

- AmgX (originally named NVAMG)
  - library for algebraic multigrid
  - see presentation given at Univ. of Warwick: http://www2.warwick.ac.uk/fac/sci/dcs/research/pcav/linearsolvers/programme/eatonnvidia.pdf

    or at NVIDIA's 2013 GTC conference:

    http://on-demand.gputechconf.com/gtc/2013/presentations/S3579-High-Performance-Algebraic-Multigrid-Commercial-Apps.pdf
  - available from

    http://developer.nvidia.com/amgx

CUDA libraries

- Thrust
  - high-level C++ template library with an interface based on the C++ Standard Template Library (STL)
  - very different philosopy to other libraries; users write standard C++ code (no CUDA) but get the benefits of GPU parallelisation
  - also supports x86 execution
  - relies on C++ object-oriented programming; certain objects exist on the GPU, and operations involving them are implicitly performed on the GPU
  - I've not used it, but for some applications it can be very powerful – e.g. lots of built-in functions for operations like sort and scan
  - also simplifies memory management and data movement

Useful header files

- dbldbl.h **available from**
  https://gist.github.com/seibert/5914108
  Header file for double-double arithmetic for quad-precision (developed by NVIDIA, but published independently under the terms of the BSD license)

- cuda_complex.h **available from**
  https://github.com/jtravs/cudacomplex/blob/master/cudacomplex.hpp
  Header file for complex arithmetic – defines a class and overloaded arithmetic operations. (NVIDIA currently has no plans to adopt this)

- helper_math.h **available in CUDA SDK Defines operator-overloading operations for CUDA intrinsic vector datatypes such as** float4

Other libraries

- MAGMA
  - a new LAPACK for GPUs – higher level numericallinear algebra, layered on top of CUBLAS
  - open source – freely available
  - Jack Dongarra, Jim Demmel and others
- FLAME
  - similar, but being developed by Robert van de Geijn at UT Austin with various collaborators

Other libraries

- **NAG RNG GPU library**
  - mrg32k3a **and Mersenne Twister pseudo-random generators**
  - **Sobol quasi-random generator**
  - **uniform, Normal, exponential outputs**
  - **Brownian bridge for use with Sobol generator**
  - **www.nag.co.uk/numeric/GPUs/**

- **CULAtools**
  - **commercial library, with dense and sparse linear algebra capabilities**
  - **restricted single-precision subset free for academics**
  - **http://www.culatools.com/**

Other libraries

- ArrayFire from Accelereyes:
  - commercial software, but free for academics and single GPU use
  - 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

- NVIDIA maintains a webpage with links to a variety of CUDA libraries and other tools:

  developer.nvidia.com/cuda-tools-ecosystem

The 7 dwarfs

- Phil Colella, 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
- key algorithmic kernels in many scientific computing applications
- very helpful to focus attention on HPC challenges and development of libraries and problem-solving environments/frameworks.

The 7 dwarfs

- dense linear algebra
- sparse linear algebra
- spectral methods
- N-body methods
- structured grids
- unstructured grids
- Monte Carlo

Dense linear algebra

- CUBLAS
- MAGMA
- FLAME
- ArrayFire
- CULAtools

Sparse linear algebra

- iterative solvers:
  - some available in PetSc
  - others can be implemented using sparse matrix-vector multiplication from CUSPARSE
  - NVIDIA has AmgX, an algebraic multigrid library

- commercial direct solvers:
  - Access Analytics (ex-Boeing Computer Services)
  - ANSYS/Acceleware

- non-commercial direct solvers:
  - SuperLU project at University of Florida (Tim Davis)
    www.cise.ufl.edu/ davis/publications_files/qrgpu_paper.pdf
  - new project at RAL (Jennifer Scott & Jonathan Hogg)

Spectral methods

- CUFFT
  - library provided / maintained by NVIDIA
- nothing else needed?

N-body methods

- OpenMM
  - 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: onlinelibrary.wiley.com/doi/10.1002/nme.3240/pdf

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

Unstructured grids

- In addition to GPU implementations of specific codes in CFD community (e.g. Rainald Löhner at GMU / NRL) 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 has developed OP2, a general-purpose open-source framework based on a previous framework built on MPI

    (Case Study 3 on Friday)

- May be other work I'm not aware of

Monte Carlo

- NVIDIA CURAND library
- NAG GPU RNG library
- Accelereyes ArrayFire library
- some examples in CUDA SDK distribution
- nothing else needed except for more output distributions?

Tools

- **Debugging:**
  - cuda-memcheck

    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**

Tools

- Other languages:
  - FORTRAN: PGI (Portland Group) CUDA FORTRAN compiler with natural FORTRAN equivalent to CUDA C
  - 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
  - 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
    http://chimera.labs.oreilly.com/books/1230000000929/ch06.html

Tools

- **Other target platforms:**
  - **PGI CUDA FORTRAN and CUDA C compilers are able to generate x86 code for multicore CPUs – this can also produce AVX code for the latest CPU vector units (but I've not tested it)**
  - **CUDA→OpenCL translation for AMD GPUs Swan is a simple tool for porting some programs**

    http://gpgpu.org/2010/03/09/swan

Tools

- Integrated Development Environments (IDE):
  - Nsight Visual Studio edition – NVIDIA plug-in for Microsoft Visual Studio

    developer.nvidia.com/nvidia-nsight-visual-studio-edition
  - Nsight Eclipse edition – IDE for Linux systems

    developer.nvidia.com/nsight-eclipse-edition
  - these come with editor, debugger, profiler integration

Tools

- **Visual Profiler:**
  - **standalone NVIDIA software for Linux and Windows systems**
  - **uses hardware counters to collect a lot of useful information**
  - **I think only 1 SMX is instrumented – implicitly assumes the others are behaving similarly**
  - **lots of things can be measured, but a limited number of counters, so it runs the application multiple times if necessary to get full info**
  - **can also obtain instruction counts from command line:**

    nvprof --metrics “flops_sp, flops_dp" prac2

    do nvprof --help **for more info on other options**

Summary

- active work on all of the dwarfs
- in most cases, significant effort to develop general purpose libraries or frameworks, to enable users to get the benefits without being CUDA experts
- too much going on for one person (e.g. me) to keep track of it all
- NVIDIA maintains a webpage with links to CUDA tools/libraries:

  developer.nvidia.com/cuda-tools-ecosystem
- the existence of this eco-system is part of why I think CUDA will remain more used than OpenCL for HPC