The Soul of CUDA

The Platform for High Performance Parallel Computing

Accessible High Performance

Enable Computing Ecosystem
Introducing CUDA 5
CUDA 5
Application Acceleration Made Easier

Dynamic Parallelism
Spawn new parallel work from within GPU code on GK110

GPU Object Linking
Libraries and plug-ins for GPU code

New Nsight™ Eclipse Edition
Develop, Debug, and Optimize... All in one tool!

GPUDirect™
RDMA between GPUs and PCIe devices
What is CUDA Dynamic Parallelism?

The ability for any GPU thread to launch a parallel GPU kernel

- Dynamically
- Simultaneously
- Independently

Fermi: Only CPU can generate GPU work

Kepler: GPU can generate work for itself

© 2012 NVIDIA
Dynamic Work Generation

Coarse grid

Higher Performance
Lower Accuracy

Fine grid

Lower Performance
Higher Accuracy

Dynamic grid

Target performance where accuracy is required
Familiar Syntax and Programming Model

```c
int main() {
    float *data;
    setup(data);
    A <<< ... >>> (data);
    B <<< ... >>> (data);
    C <<< ... >>> (data);
    cudaDeviceSynchronize();
    return 0;
}

__global__ void B(float *data) {
    do_stuff(data);
    X <<< ... >>> (data);
    Y <<< ... >>> (data);
    Z <<< ... >>> (data);
    cudaDeviceSynchronize();
    do_more_stuff(data);
}```
Simpler Code: LU Example

LU decomposition (Fermi)

\[
dgetrf(N, N) \{
\text{for } j=1 \text{ to } N \\
\quad \text{for } i=1 \text{ to } 64 \\
\quad \quad \text{idamax}<<<>>> \\
\quad \quad \text{memcpy} \\
\quad \quad \text{dswap}<<<>>> \\
\quad \quad \text{memcpy} \\
\quad \quad \text{dscal}<<<>>> \\
\quad \quad \text{dger}<<<>>> \\
\quad \quad \text{next } i \\
\quad \text{memcpy} \\
\quad \text{dlaswap}<<<>>> \\
\quad \text{dtrsm}<<<>>> \\
\quad \text{dgemm}<<<>>> \\
\text{next } j \\
\}
\]

LU decomposition (Kepler)

\[
dgetrf(N, N) \{ \\
\quad \text{idamax}<<<>>> \\
\quad \text{dswap}<<<>>> \\
\quad \text{dscal}<<<>>> \\
\quad \text{dger}<<<>>> \\
\quad \text{dlaswap}<<<>>> \\
\quad \text{dtrsm}<<<>>> \\
\quad \text{dgemm}<<<>>> \\
\quad \text{synchronize}(); \\
\text{next } i \\
\quad \text{dgetrf}<<<>>> \\
\text{next } j \\
\}
\]

CPU Code

GPU Code

© 2012 NVIDIA
Bonsai GPU Tree-Code

Journal of Computational Physics, 231:2825-2839, April 2012

- Jeroen Bédorf, Simon Portegies Zwart
  - Leiden Observatory, The Netherlands
- Evghenii Gaburov
  - CIERA @ Northwestern U.
  - SARA, The Netherlands

Galaxies generated with:
Galatics
Mapping Compute to the Problem
CUDA 4: Whole-Program Compilation & Linking

CUDA 4 required single source file for a single kernel
No linking external device code
CUDA 5: Separate Compilation & Linking

Separate compilation allows building independent object files

CUDA 5 can link multiple object files into one program
CUDA 5: Separate Compilation & Linking

Can also combine object files into static libraries
Link and externally call device code
Facilitates code reuse, reduces compile time

© 2012 NVIDIA
CUDA 5: Separate Compilation & Linking

Enables closed-source device libraries to call user-defined device callback functions
NVIDIA® Nsight™ Eclipse Edition

CUDA-Aware Editor
- Automated CPU to GPU code refactoring
- Semantic highlighting of CUDA code
- Integrated code samples & docs

Nsight Debugger
- Simultaneously debug of CPU and GPU
- Inspect variables across CUDA threads
- Use breakpoints & single-step debugging

Nsight Profiler
- Quickly identifies performance issues
- Integrated expert system
- Source line correlation

Available for Linux and Mac OS
CUDA on Mac!
NVIDIA GPUDirect™ now supports RDMA

Server 1
RDMA: Remote Direct Memory Access between any GPUs in your *cluster*
## GK10x vs. GK110 Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>GK10x</th>
<th>GK110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyper-Q</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic Parallelism</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>WARP shuffle instructions</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Read-only Cache</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Bigger Register File (65536 x 32-bit)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Faster Atomics</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RDMA (Tesla-only)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Double L2 size and speed</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Improved ECC performance</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Try out CUDA 5

CUDA 5.0 Preview (alpha)
- Become a registered developer and download CUDA 5.0 preview
- Use GPU linking and NSIGHT EE—both work with Fermi & GK104
- Peruse early documentation and header files for GK110 features
  - SM 3.5 support and Dynamic Parallelism
- Provide feedback to NVIDIA via CUDA Forums and
  [CUDA_RegDev@nvidia.com](mailto:CUDA_RegDev@nvidia.com)

CUDA 5.0 Release Candidate
- Later in 2012 (TBD)
- Full support for all CUDA 5.0 features

© 2012 NVIDIA
Beyond CUDA 5
Platform for Parallel Computing

The CUDA Platform is a foundation that supports a diverse parallel computing ecosystem.
Diversity of Programming Languages

ActionScript  Ada  Assembly  Autoconf  Automake  AWK  BlitzMax  Boo  Brainfuck  Brainfuck++  C

C++  C/C++  ChaiScript  Classic Basic  ClearSilver  Clojure  CMake  CoffeeScript  CSS  CUDA  D  DCL  DOS

batch  script  Dylan  Ebuild  eC  Eiffel  Emacs  Lisp  Erlang  Exheres  F#  Factor  forth  Fortran  (Fixed-format)  Fortran

(Free-format)  Go  Groovy  Haml  Haskell  HaXe  HTML  IDL/PV-WAVE/GDL  Jam  Java

JavaScript  Limbo  Lisp  Logtalk  Lua  Make  Matlab  MetaFont  MetaPost  Modula-2  Modula-3  MXML  Nix

NSIS  Oberon  Objective-C  Objective-Caml  Objective-J  Octave  OpenGL  Shading  Pascal  Perl  PHP  Pike

Prolog  Puppet  Python  QML  R  Racket  REBOL  Rexx  Ruby  Scala  Scheme  Scilab  shell script  Smalltalk  SQL

Stratego  Structured Basic  Tcl  TeX/LaTeX  Vala  VHDL  Vim Script  Visual Basic  XAML  XML  XML Schema

XSL  Transformation

http://www.ohloh.net
Rapid Parallel C++ Development

- Resembles C++ STL
- Open source
- High-level interface
  - Enhances developer productivity
  - Enables performance portability between GPUs and multicore CPUs
- Flexible
  - CUDA, OpenMP, and TBB backends
  - Extensible and customizable
  - Integrates with existing software

// generate 32M random numbers on host
thrust::host_vector<int> h_vec(32 << 20);
thrust::generate(h_vec.begin(), h_vec.end(), rand);

// transfer data to device (GPU)
thrust::device_vector<int> d_vec = h_vec;

// sort data on device
thrust::sort(d_vec.begin(), d_vec.end());

// transfer data back to host
thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());

Expressive Parallel Programming

```c++
transform(rows, Y,
    [=](uint2 row) {
        return inner_product(
            slice(nonZeros, row.x, row.y),
            gather(X, slice(col, row.x, row.y)));
    });
```

For each row of sparse matrix $M$

$Y[i] = \text{inner product of non-zero elements of row with corresponding elements of } X$

$Y = \begin{bmatrix}
4 & -1 & \cdots & 1 \\
-1 & 4 & \cdots & 1 \\
\vdots & \ddots & \vdots & \vdots \\
1 & \cdots & 4 & -1
\end{bmatrix}$
OpenACC Directives

Program myscience
... serial code ...
!$acc kernels
  do k = 1,n1
    do i = 1,n2
      ... parallel code ...
    enddo
  enddo
!$acc end kernels
... End Program myscience

Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs & multicore CPUs

Your original Fortran or C code
OpenACC
The Standard for GPU Directives

Easy: Directives are the easy path to accelerate compute intensive applications

Open: OpenACC is an open GPU directives standard, making GPU programming straightforward and portable across parallel and multi-core processors

Powerful: GPU Directives allow complete access to the massive parallel power of a GPU
High-level, with low-level access

- Compiler directives to specify parallel regions in C, C++, Fortran
  - OpenACC compilers offload parallel regions from host to accelerator
  - Portable across OSes, host CPUs, accelerators, and compilers

- Create high-level heterogeneous programs
  - Without explicit accelerator initialization,
  - Without explicit data or program transfers between host and accelerator

- Programming model allows programmers to start simple
  - Enhance with additional guidance for compiler on loop mappings, data location, and other performance details

- Compatible with other GPU languages and libraries
  - Interoperate between CUDA C/Fortran and GPU libraries
  - e.g. CUFFT, CUBLAS, CUSPARSE, etc.
Domain-specific Languages

MATLAB

R Statistical Computing Language

Liszt
A DSL for solving mesh-based PDEs
CUDA Compiler Contributed to Open Source LLVM

Developers want to build front-ends for Java, Python, R, DSLs

Target other processors like ARM, FPGA, GPUs, x86
The Future is Heterogeneous

Many solutions build a heterogeneous future
- General-purpose Languages
- Directives
- DSLS
mebersole@nvidia.com
or
cudahamster@nvidia.com

Twitter: @CudaHamster

Mark Ebersole