

CUDA 5 and Beyond

Mark Harris
Chief Technologist, GPU Computing

CUDA By the Numbers:

>375,000,000

CUDA-Capable GPUs

>1,000,000

Toolkit Downloads

>120,000

Active Developers

>500

Universities Teaching CUDA

CUDA By the Numbers:

1 download

every 60 seconds

Inspiration for CUDA

CUDA Spotlights



CUDA Fellows



Takayuki Aoki
Tokyo Tech

Lorena Barba
Boston University

Mike Giles
Oxford
University

Scott LeGrand
Amazon

PJ Narayanan
IIIT, Hyderabad

Dan Negrut
University of
Wisconsin

John Stone
UIUC

Manuel Ujaldon
University of
Malaga

Ross Walker
SDSC

The Soul of CUDA

The Platform for High Performance
Parallel Computing

Accessible High
Performance

Enable Computing
Ecosystem

An abstract, high-resolution geometric pattern composed of many small, interconnected cubes or squares. The colors are primarily vibrant green and cyan, with some blue and purple accents, creating a shimmering, crystalline effect. The pattern is set against a dark background, making the bright colors stand out.

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

Dynamic Parallelism

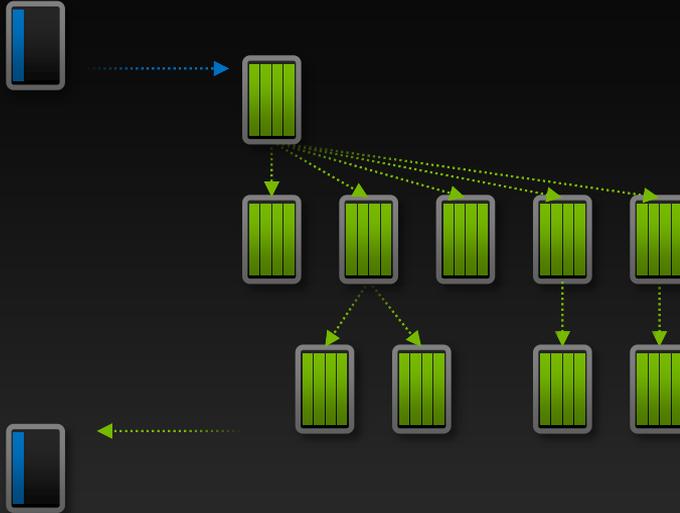
CPU

Fermi GPU



CPU

Kepler GPU



What is CUDA Dynamic Parallelism?

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

- Dynamically
- Simultaneously
- Independently

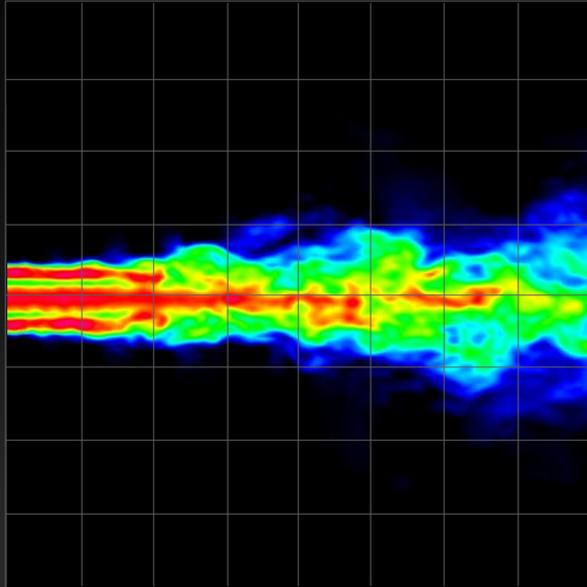


Fermi: Only CPU can generate GPU work

Kepler: GPU can generate work for itself

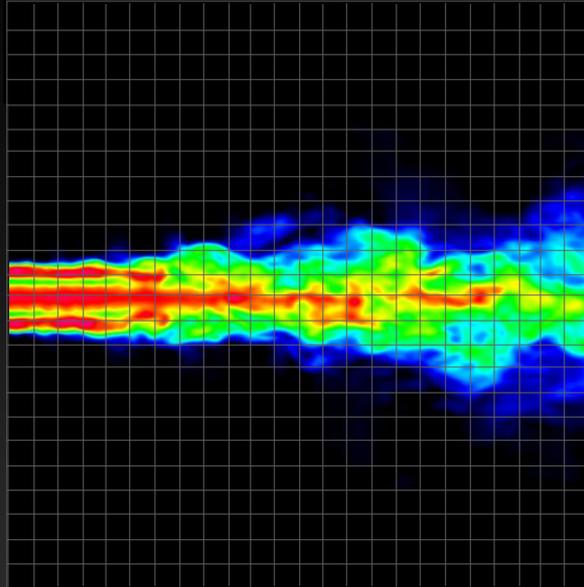
Dynamic Work Generation

Coarse grid



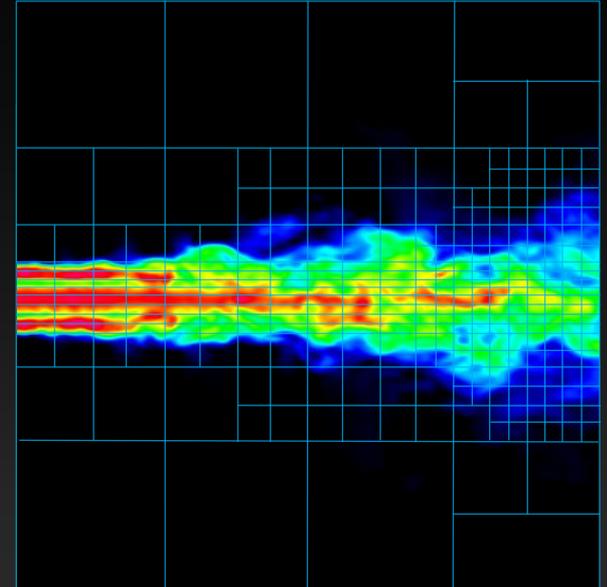
Higher Performance
Lower Accuracy

Fine grid



Lower Performance
Higher Accuracy

Dynamic grid

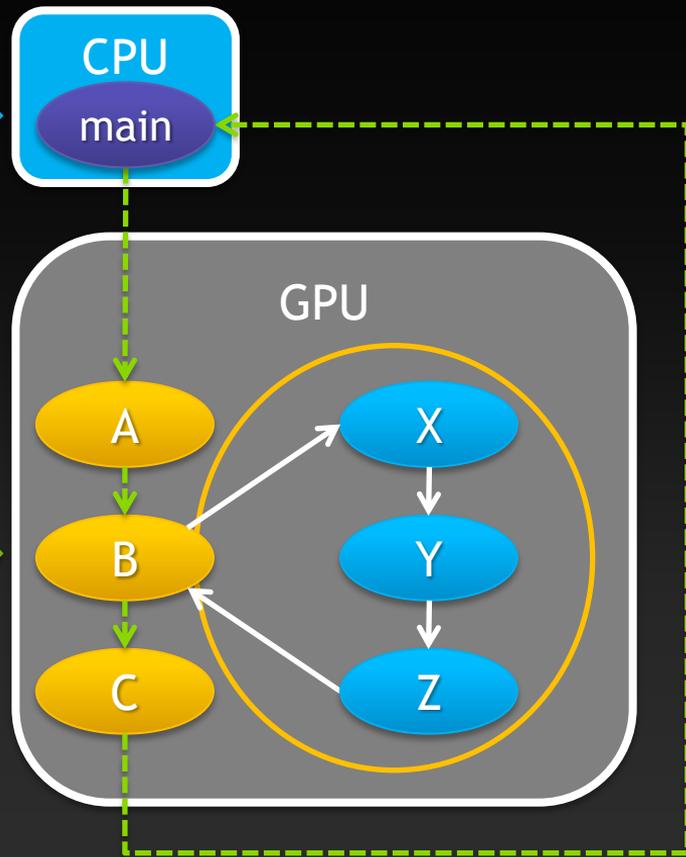


Target performance where
accuracy is required

Familiar Syntax and Programming Model

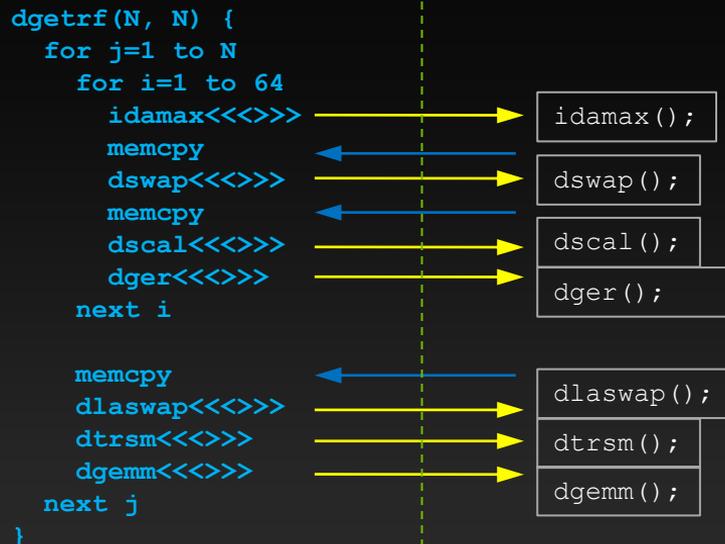
```
int main() {  
    float *data;  
    setup(data);  
  
    A <<< ... >>> (data);  
    B <<< ... >>> (data);  
    C <<< ... >>> (data);  
  
    cudaDeviceSynchronize();  
    return 0;  
}
```

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



Simpler Code: LU Example

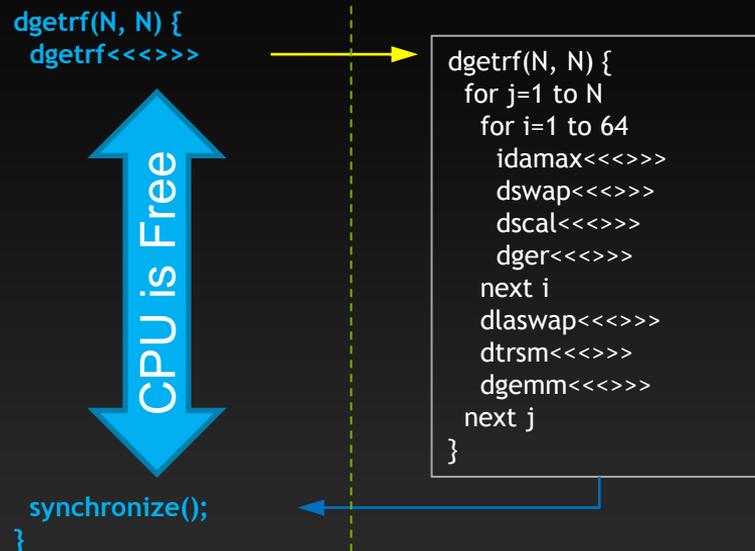
LU decomposition (Fermi)



CPU Code

GPU Code

LU decomposition (Kepler)



CPU Code

GPU Code

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
Widrow L. M., Dubinski J., 2005,
Astrophysical Journal, 631 838

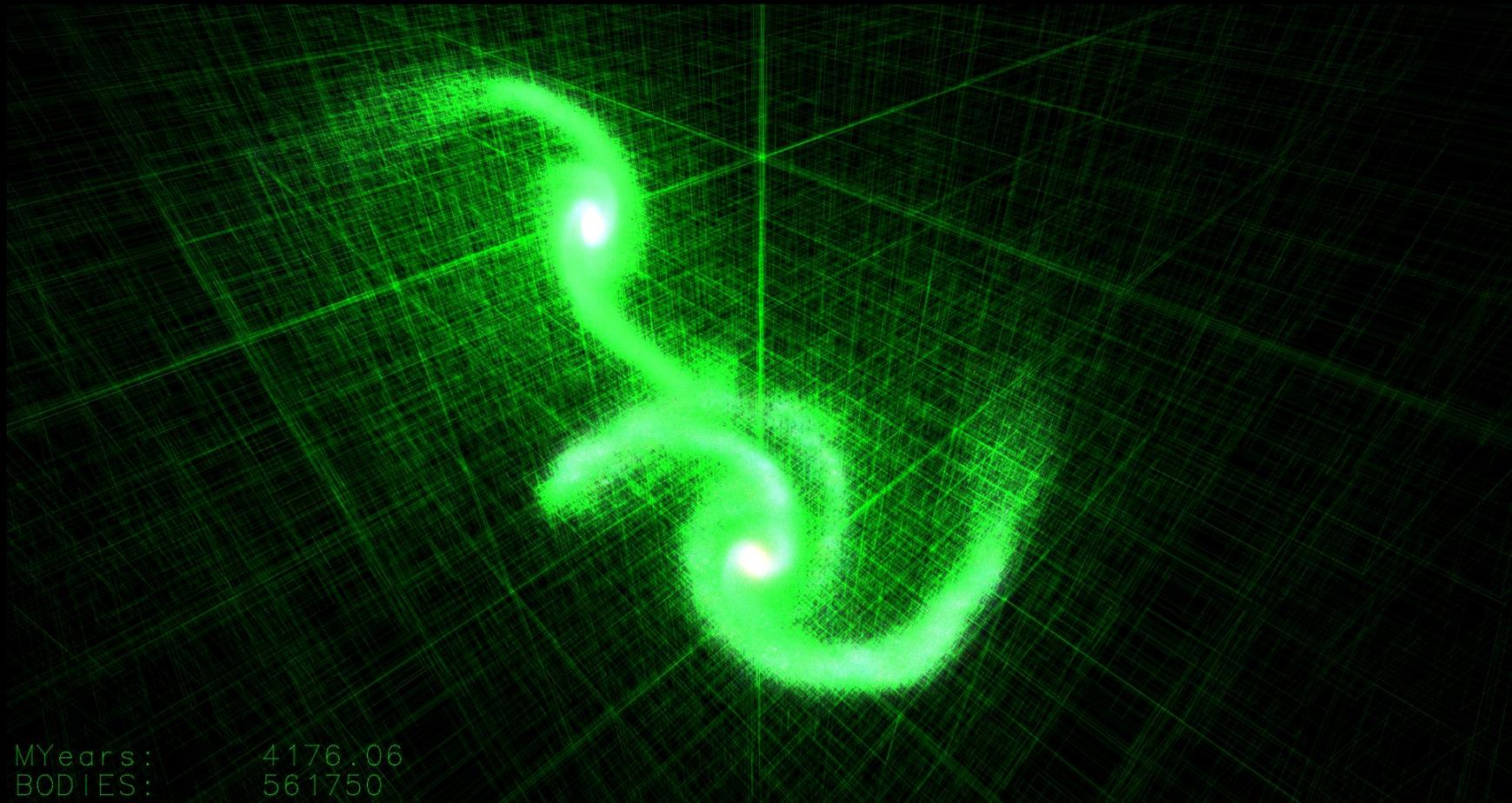


Mapping Compute to the Problem

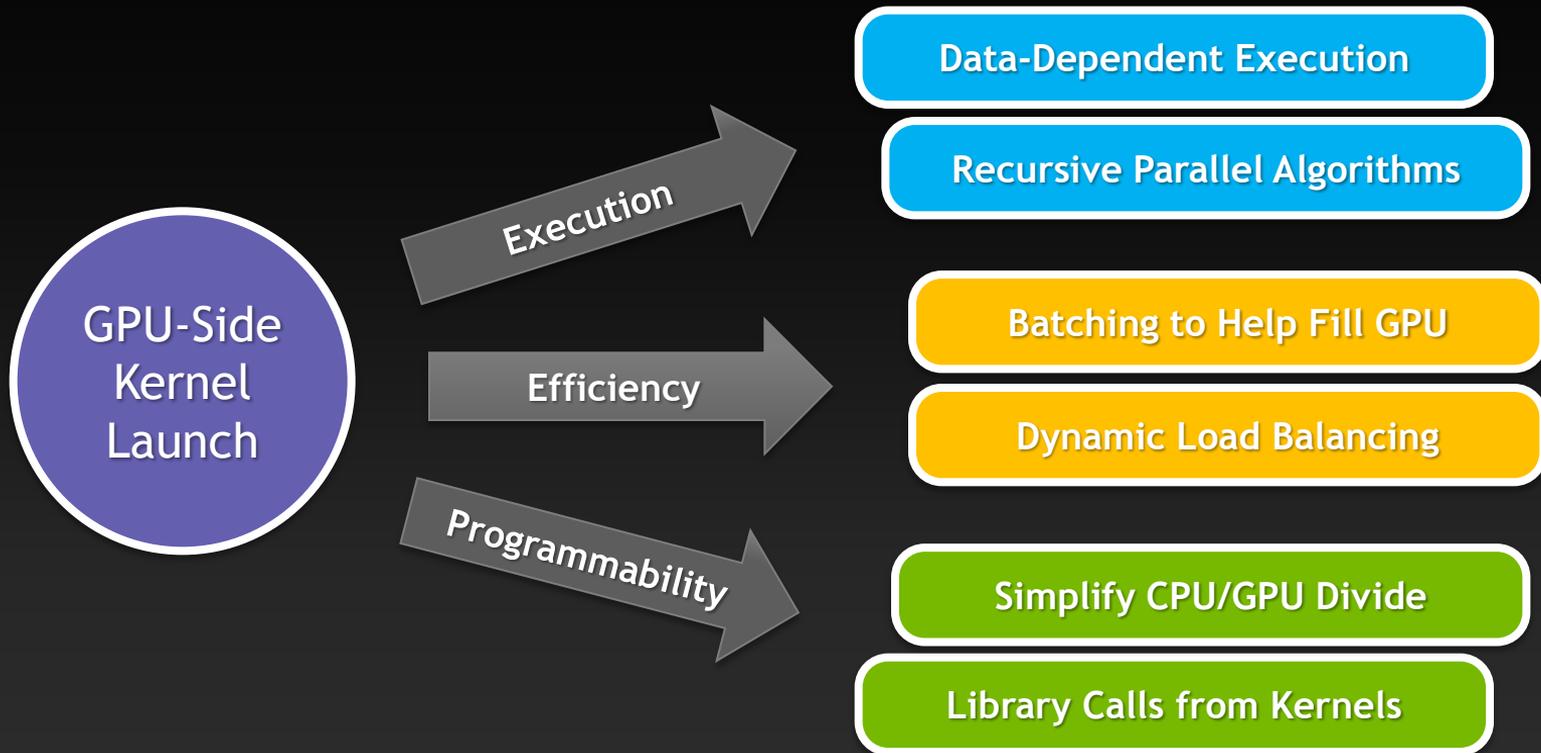


MYears: 4176.06
BODIES: 561750

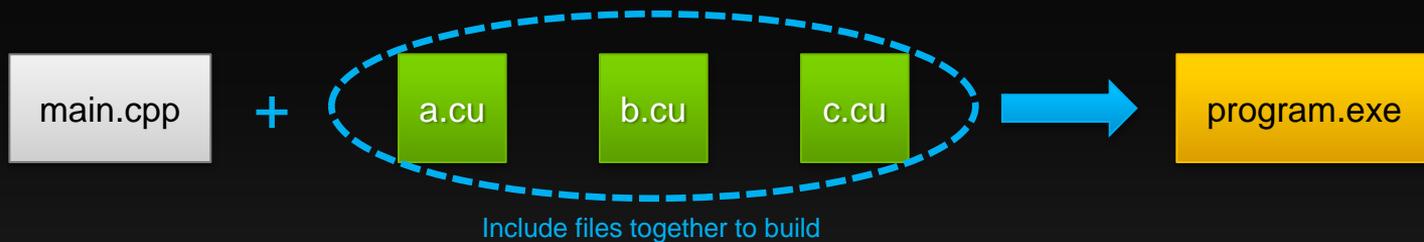
Mapping Compute to the Problem



CUDA Dynamic Parallelism

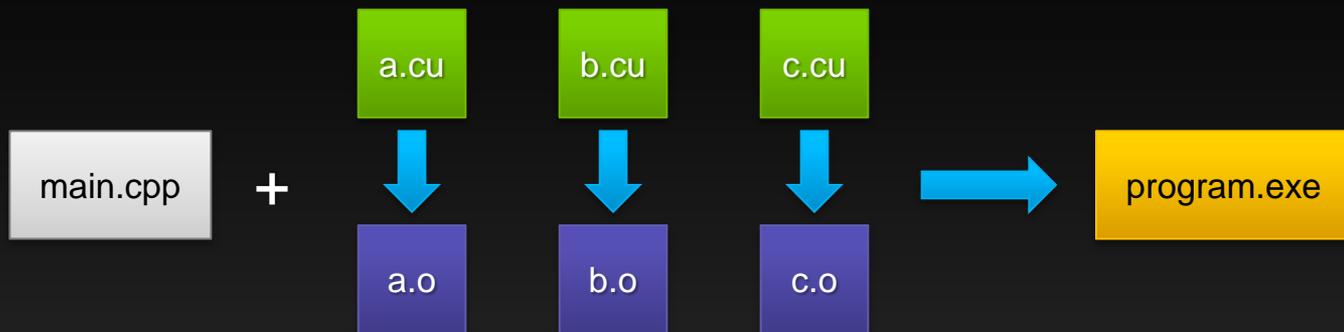


CUDA 4: Whole-Program Compilation & Linking



CUDA 4 required single source file for a single kernel
No linking external device code

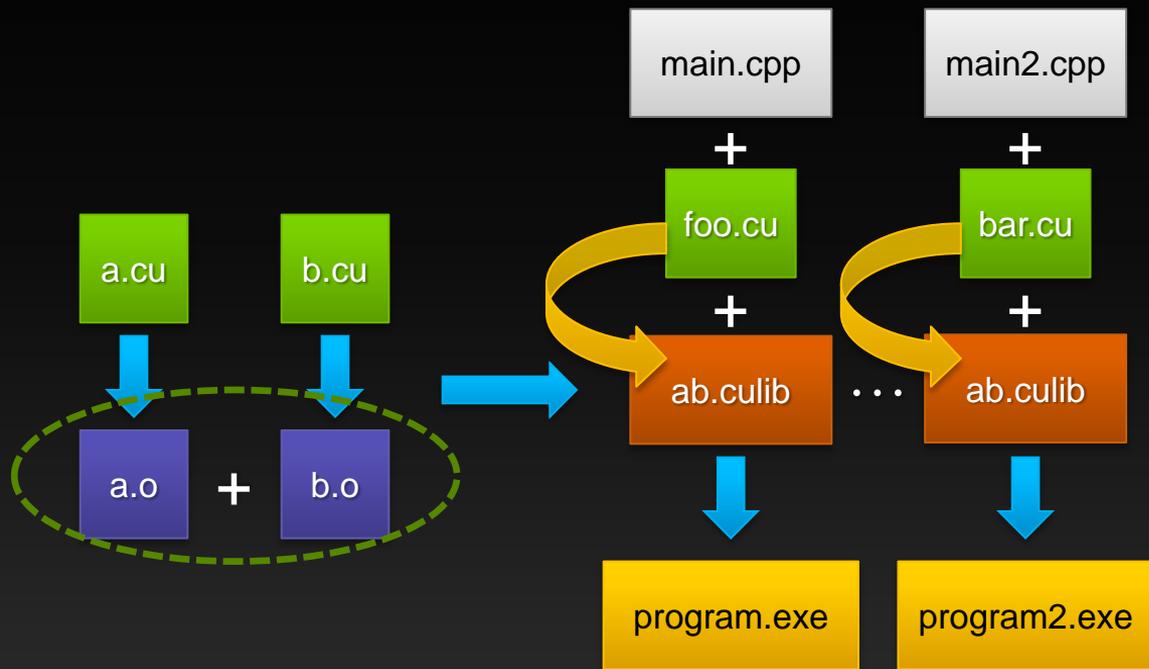
CUDA 5: Separate Compilation & Linking



Separate compilation allows building independent object files

CUDA 5 can link multiple object files into one program

CUDA 5: Separate Compilation & Linking



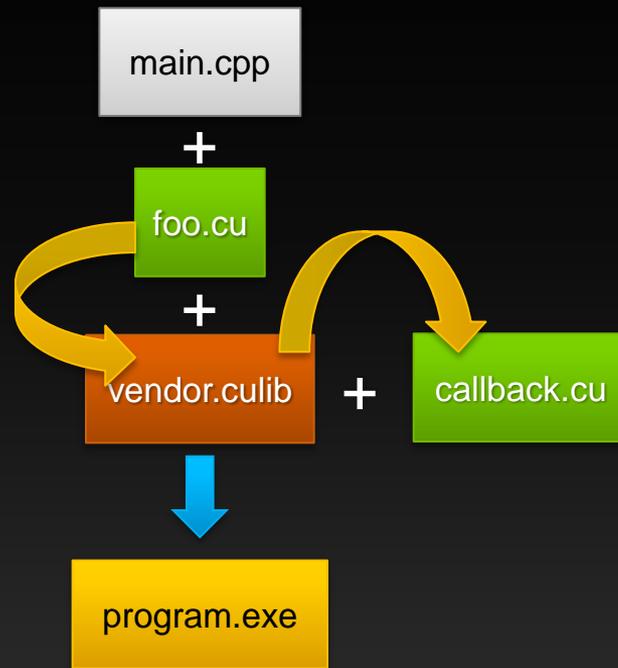
Can also combine object files into static libraries

Link and externally call *device* code

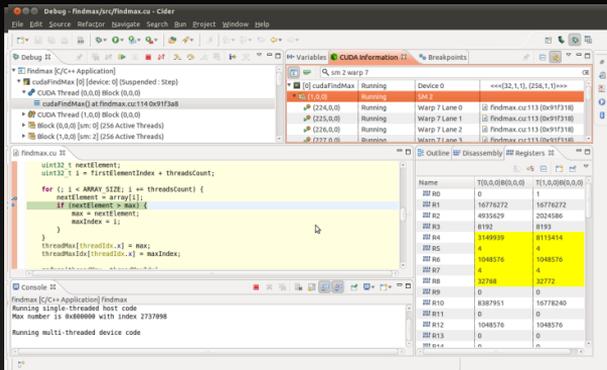
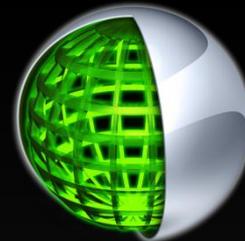
Facilitates code reuse, reduces compile time

CUDA 5: Separate Compilation & Linking

Enables closed-source device libraries to call user-defined device callback functions

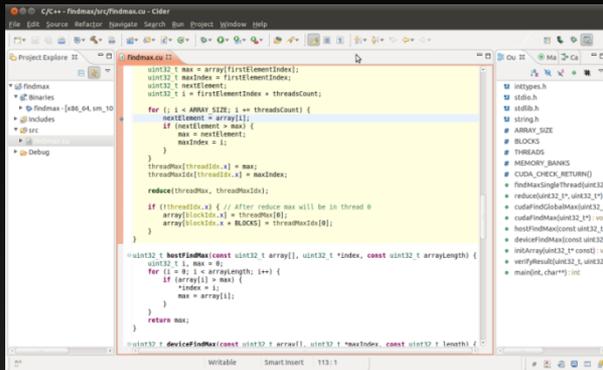


NVIDIA® Nsight™, Eclipse Edition



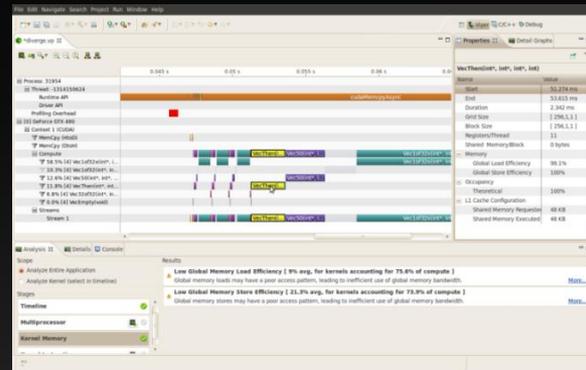
CUDA-Aware Editor

- 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

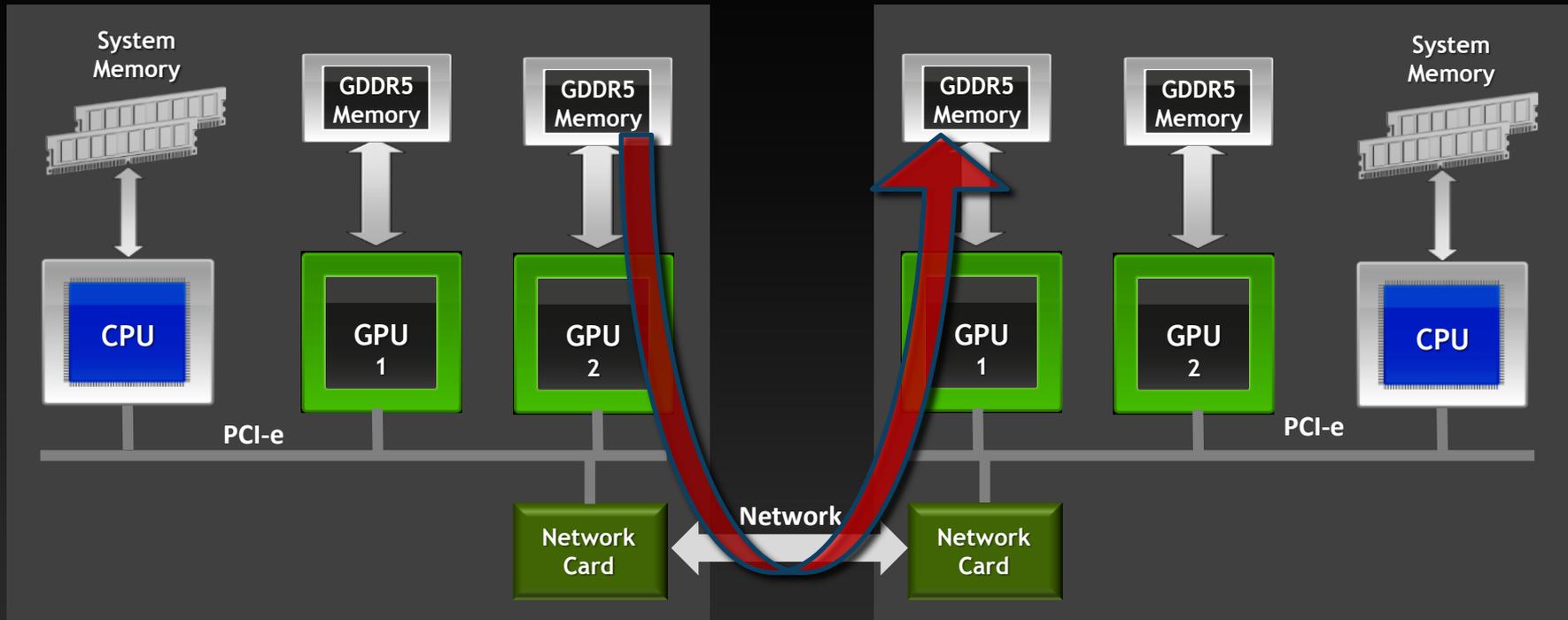


Nsight Profiler

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

Available for Linux and Mac OS

NVIDIA GPUDirect™ now supports RDMA



Server 1

Server 2

RDMA: Remote Direct Memory Access between any GPUs in your *cluster*

Other talks on CUDA 5 and Kepler

- S0235 - “Compiling CUDA and Other Languages for GPUs” (LLVM)
 - Vinod Grover and Yuan Lin, Wed 10am
- S0338 - “New Features in the CUDA Programming Model”
 - Stephen Jones, 10am Thursday
- S0642 - “Inside Kepler”
 - Stephen Jones and Lars Nyland, Wed 2pm
- S0419A - “Optimizing Application Performance with CUDA Profiling Tools”
 - David Goodwin, 9am Tuesday

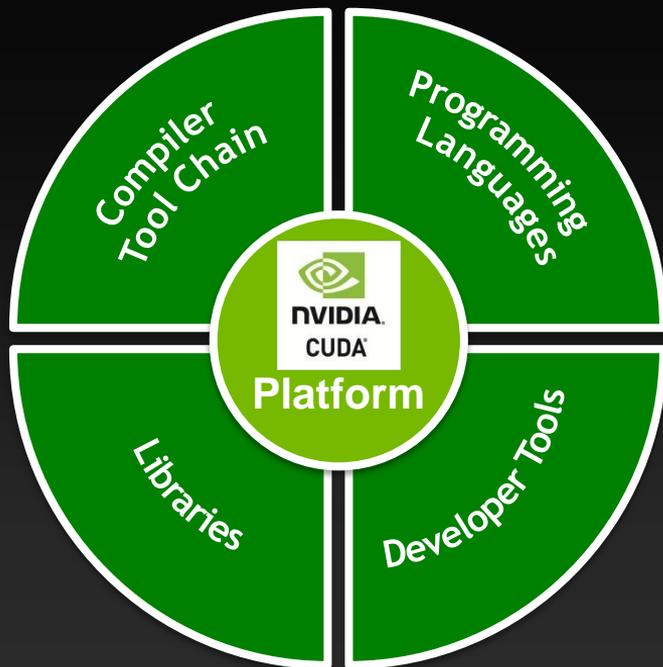
Try out CUDA 5

- **CUDA 5.0 Preview (alpha)**
 - Become a registered developer and download CUDA 5.0 preview
<http://developer.nvidia.com/user/register>
 - 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
- **CUDA 5.0 Release Candidate**
 - Later in 2012 (TBD)
 - Full support for all CUDA 5.0 features

The background features a complex, abstract pattern of overlapping, semi-transparent grid lines in shades of green, cyan, and blue. These lines form a curved, cylindrical shape that recedes into the distance, creating a sense of depth and motion. The overall effect is reminiscent of a digital or data visualization, possibly representing a GPU architecture or a complex data set.

Beyond CUDA 5

Platform for Parallel Computing



The CUDA Platform is a foundation that supports a diverse parallel computing ecosystem.

Diversity of Programming Languages

ohloh

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

C

C#

C++

C/C++ ChaiScript ClassicBasic 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

XSL Transformation

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

<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

```
transform(rows, Y,
```

```
 [=](uint2 row) {
```

Dynamic
Parallel
Launches

```
return inner_product(  
    slice(nonZeros, row.x, row.y),  
    gather(x, slice(col, row.x, row.y)));  
});
```

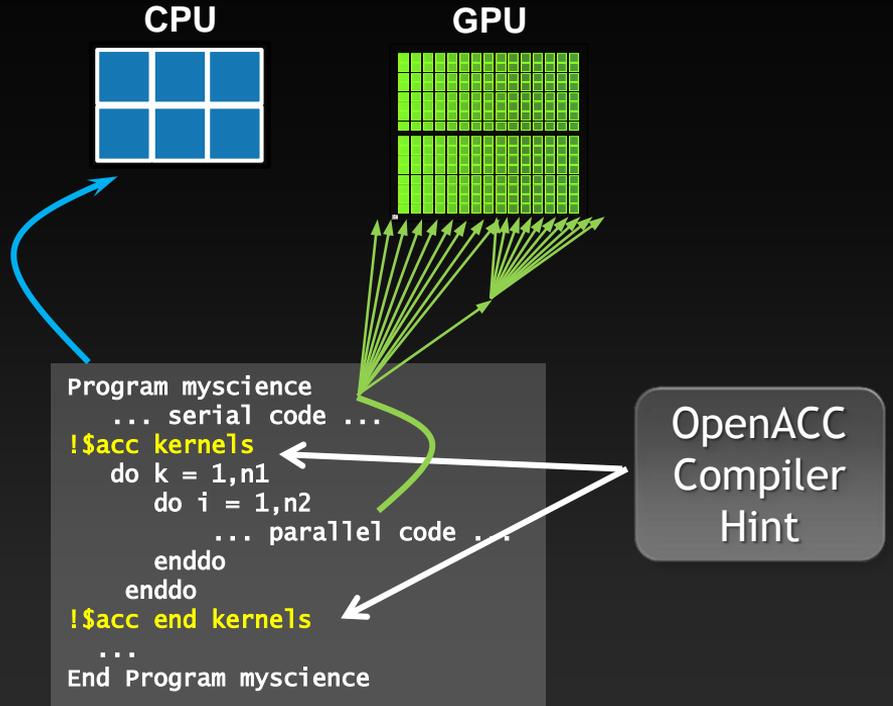
C++11
Lambda
Function

For each row of sparse matrix M

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

$$Y = \begin{bmatrix} 4 & -1 & \dots & 1 \\ -1 & 4 & & \\ \vdots & & \ddots & \vdots \\ 1 & & \dots & 4 & -1 \\ & & & -1 & 4 \end{bmatrix} X$$

OpenACC Directives



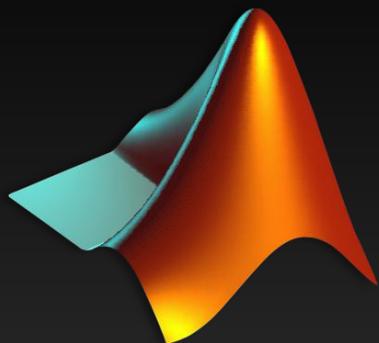
Your original
Fortran or C code

Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs &
multicore CPUs

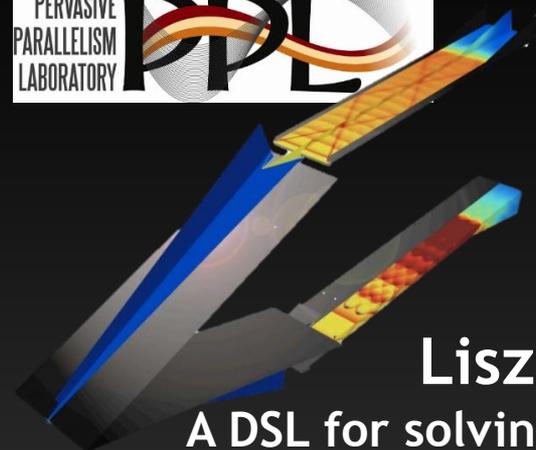
Domain-specific Languages



MATLAB



**R Statistical
Computing Language**



Liszt
A DSL for solving
mesh-based PDEs

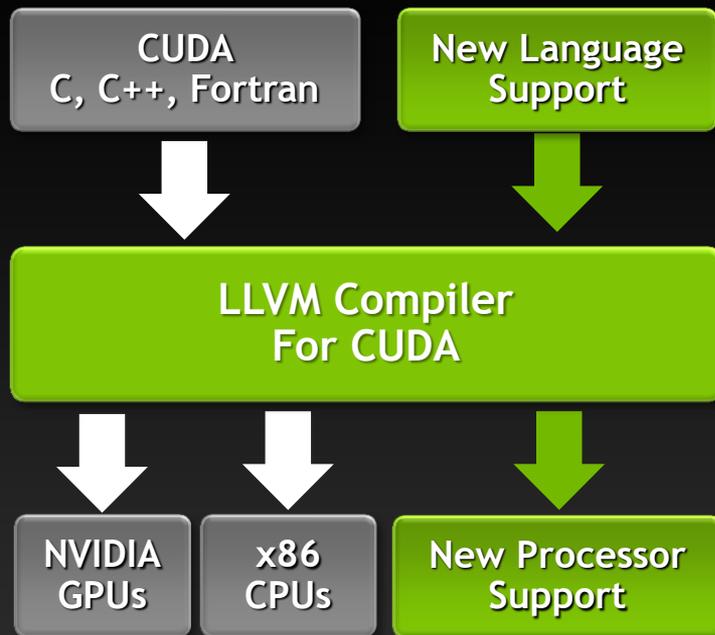
CUDA Compiler Contributed to Open Source LLVM

Developers want to build
front-ends for

Java, Python, R, DSLs

Target other processors like

ARM, FPGA, GPUs, x86

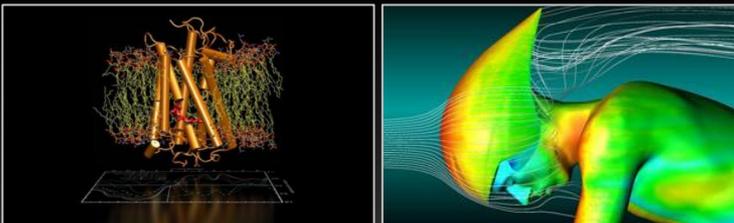


Building A Massively Parallel Future

- The Future is Heterogeneous
- Many solutions build a heterogeneous future
 - General-purpose Languages
 - Directives
 - DSLS

Education & Research

Domain Science



Languages and Compilers



Heterogeneous Architectures



CUDA on ARM

Computer Science

