

# Introduction to the CUDA Platform

by NVIDIA Corporation

Source of presentation slides:

<https://developer.nvidia.com/cuda-education>

# CUDA Parallel Computing Platform

[www.nvidia.com/getcuda](http://www.nvidia.com/getcuda)

Programming  
Approaches

Libraries

“Drop-in”  
Acceleration

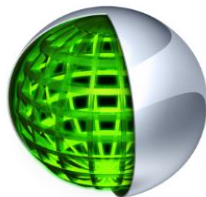
OpenACC  
Directives

Easily Accelerate  
Apps

Programming  
Languages

Maximum Flexibility

Development  
Environment



Nsight IDE  
Linux, Mac and Windows  
GPU Debugging and  
Profiling

CUDA-GDB  
debugger  
NVIDIA Visual  
Profiler

Open Compiler  
Tool Chain



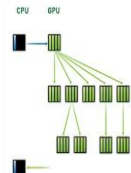
Enables compiling new languages to CUDA  
platform, and CUDA languages to other  
architectures

Hardware  
Capabilities

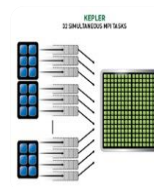
SMX



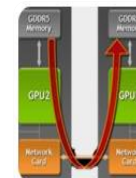
Dynamic  
Parallelism



HyperQ



GPUDirect



# 3 Ways to Accelerate Applications

Applications

Libraries

“Drop-in”  
Acceleration

OpenACC  
Directives

Easily Accelerate  
Applications

Programming  
Languages

Maximum  
Flexibility

# 3 Ways to Accelerate Applications

Applications

Libraries

“Drop-in”  
Acceleration

OpenACC  
Directives

Easily Accelerate  
Applications

Programming  
Languages

Maximum  
Flexibility

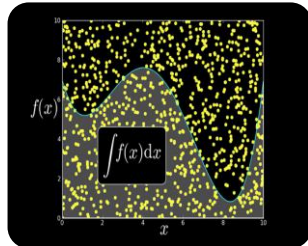
# Libraries: Easy, High-Quality Acceleration

- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **“Drop-in”:** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
- **Quality:** Libraries offer high-quality implementations of functions encountered in a broad range of applications
- **Performance:** NVIDIA libraries are tuned by experts

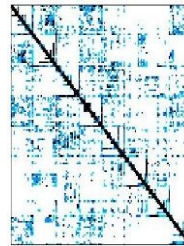
# Some GPU-accelerated Libraries



NVIDIA cuBLAS



NVIDIA cuRAND



NVIDIA cuSPARSE



NVIDIA NPP



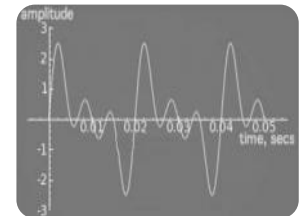
Vector Signal  
Image Processing



GPU Accelerated  
Linear Algebra



Matrix Algebra  
on GPU and  
Multicore



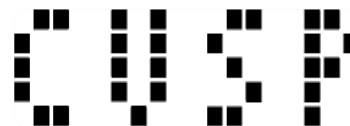
NVIDIA cuFFT



IMSL Library



ArrayFire Matrix  
Computations



Sparse Linear  
Algebra




C++ STL  
Features for  
CUDA



# 3 Steps to CUDA-accelerated application

- **Step 1:** Substitute library calls with equivalent CUDA library calls

`saxpy ( ... )`            `cublasSaxpy ( ... )`

- **Step 2:** Manage data locality

- with CUDA:      `cudaMalloc()`, `cudaMemcpy()`, etc.
- with CUBLAS:      `cublasAlloc()`, `cublasSetVector()`, etc.

- **Step 3:** Rebuild and link the CUDA-accelerated library

`nvcc myobj.o -l cublas`

# Explore the CUDA (Libraries) Ecosystem

- CUDA Tools and Ecosystem described in detail on NVIDIA Developer Zone:  
[developer.nvidia.com/cuda-tools-ecosystem](http://developer.nvidia.com/cuda-tools-ecosystem)

The screenshot displays the NVIDIA Developer Zone website. The top navigation bar includes the NVIDIA logo, 'DEVELOPER ZONE', and links for Log In, Feedback, and New Account. Below the navigation bar, the 'GPU-Accelerated Libraries' section is highlighted. It features a grid of library cards, each with an icon, title, and brief description. The libraries shown include NVIDIA cuFFT, NVIDIA cuBLAS, CULA|tools, MAGMA, IMSL Fortran Numerical Library, NVIDIA cuSPARSE, CUSP, AccelerEyes ArrayFire, NVIDIA cuRAND, NVIDIA NPP, NVIDIA CUDA Math Library, and Thrust. A sidebar on the right contains 'QUICKLINKS' (The NVIDIA Registered Developer Program, Registered Developers Website, NVDeveloper (old site), CUDA Newsletter, CUDA Downloads, CUDA GPUs, Get Started - Parallel Computing, CUDA Spotlights, CUDA Tools & Ecosystem) and 'FEATURED ARTICLES' (Introducing NVIDIA NSIGHT VISUAL STUDIO EDITION 2.2, WITH LOCAL SINGLE GPU CUDA DEBUGGING!). The bottom of the sidebar shows 'LATEST NEWS' with articles like 'OpenACC Compiler For \$199' and 'Introducing NVIDIA NSIGHT Visual Studio Edition 2.2, With Local Single GPU CUDA Debugging!'.

**GPU-Accelerated Libraries**

Adding GPU-acceleration to your application can be as easy as simply calling a library function. Check out the extensive list of high performance GPU-accelerated libraries below. If you would like other libraries added to this list please [contact us](#).

**NVIDIA cuFFT**  
NVIDIA CUDA Fast Fourier Transform Library (cuFFT) provides a simple interface for computing FFTs up to 10x faster, without having to develop your own custom GPU FFT implementation.

**NVIDIA cuBLAS**  
NVIDIA CUDA BLAS Library (cuBLAS) is a GPU-accelerated version of the complete standard BLAS library that delivers 6x to 17x faster performance than the latest MKL BLAS.

**CULA|tools**  
**CULA Tools**  
GPU-accelerated linear algebra library by EM Photonics, that utilizes CUDA to dramatically improve the computation speed of sophisticated mathematics.

**MAGMA**  
A collection of next gen linear algebra routines. Designed for heterogeneous GPU-based architectures. Supports current LAPACK and BLAS standards.

**IMSL Fortran Numerical Library**  
Developed by RogueWave, a comprehensive set of mathematical and statistical functions that offload work to GPUs.

**NVIDIA cuSPARSE**  
NVIDIA CUDA Sparse (cuSPARSE) Matric library provides a collection of basic linear algebra subroutines used for sparse matrices that delivers over 8x performance boost.

**CUSP**  
**NVIDIA CUSP**  
A GPU accelerated Open Source C++ library of generic parallel algorithms for sparse linear algebra and graph computations. Provides an easy to use high-level interface.

**AccelerEyes ArrayFire**  
**AccelerEyes ArrayFire**  
Comprehensive GPU function library, including functions for math, signal and image processing, statistics, and more. Interfaces for C, C++, Fortran, and Python.

**NVIDIA cuRAND**  
The CUDA Random Number Generation library performs high quality GPU-accelerated random number generation (RNG) over 8x faster than typical CPU only code.

**NVIDIA NPP**  
NVIDIA Performance Primitives is a GPU accelerated library with a very large collection of 1000s of image processing functions.

**NVIDIA CUDA Math Library**  
An industry proven, highly accurate collection of standard mathematical functions, providing high performance for GPU applications.

**Thrust**  
A powerful, open source library of parallel algorithms and data structures. Perform GPU-accelerated sort, scan, transform, and reductions.

**QUICKLINKS**  
The NVIDIA Registered Developer Program  
Registered Developers Website  
NVDeveloper (old site)  
CUDA Newsletter  
CUDA Downloads  
CUDA GPUs  
Get Started - Parallel Computing  
CUDA Spotlights  
CUDA Tools & Ecosystem

**FEATURED ARTICLES**  
**INTRODUCING NVIDIA NSIGHT VISUAL STUDIO EDITION 2.2, WITH LOCAL SINGLE GPU CUDA DEBUGGING!**

**LATEST NEWS**  
OpenACC Compiler For \$199  
Introducing NVIDIA NSIGHT Visual Studio Edition 2.2, With Local Single GPU CUDA Debugging!  
CUDA Spotlight: Lorena Barba, Boston University  
Stanford To Host CUDA On Campus Day, April 13, 2012  
CUDA Spotlight:



# 3 Ways to Accelerate Applications

Applications

Libraries

“Drop-in”  
Acceleration

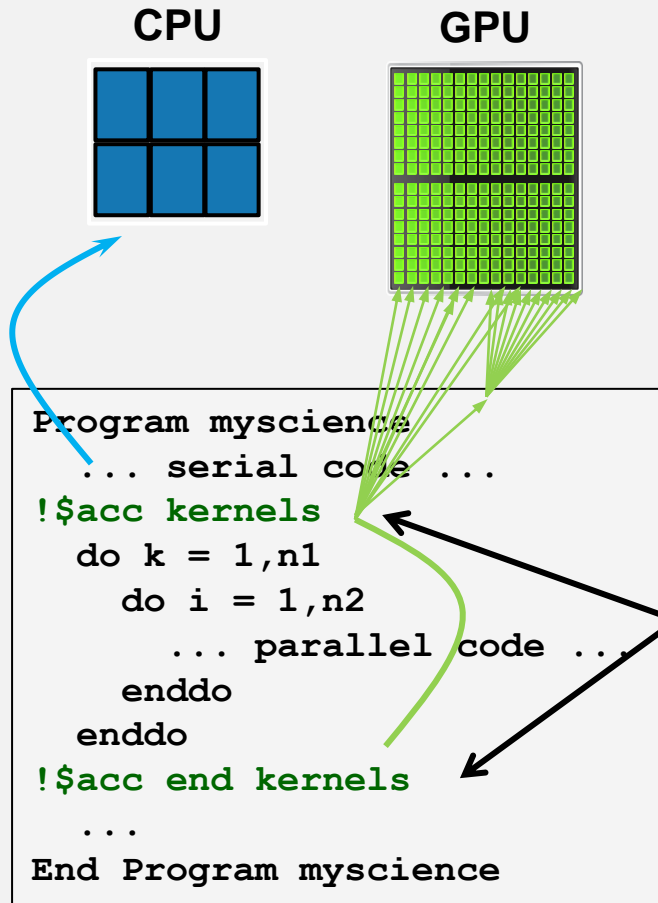
OpenACC  
Directives

Easily Accelerate  
Applications

Programming  
Languages

Maximum  
Flexibility

# OpenACC Directives



Your original  
Fortran or C  
code

Simple Compiler hints

Compiler Parallelizes  
code

Works on many-core  
GPUs & multicore CPUs

# OpenACC



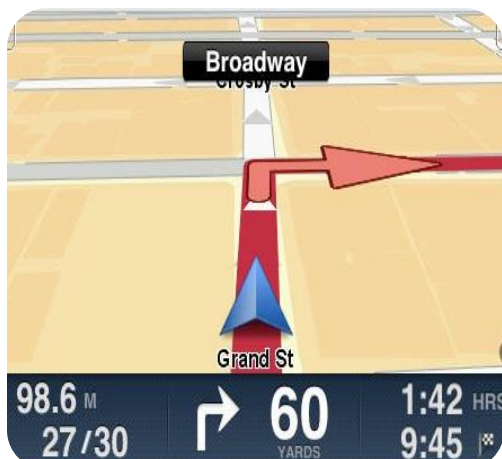
## The Standard for GPU Directives

- **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

# Directives: Easy & Powerful

## Real-Time Object Detection

Global Manufacturer of Navigation Systems



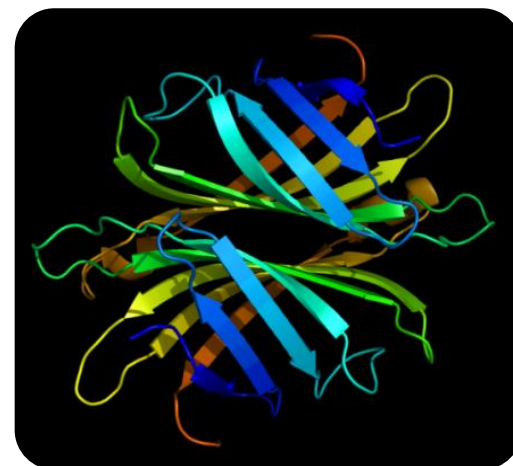
## Valuation of Stock Portfolios using Monte Carlo

Global Technology Consulting Company



## Interaction of Solvents and Biomolecules

University of Texas at San Antonio



**5x in 40 Hours** **2x in 4 Hours** **5x in 8 Hours**

“Optimizing code with directives is quite easy, especially compared to CPU threads or writing CUDA kernels. The most important thing is avoiding restructuring of existing code for production applications.”

-- Developer at the Global Manufacturer of Navigation Systems

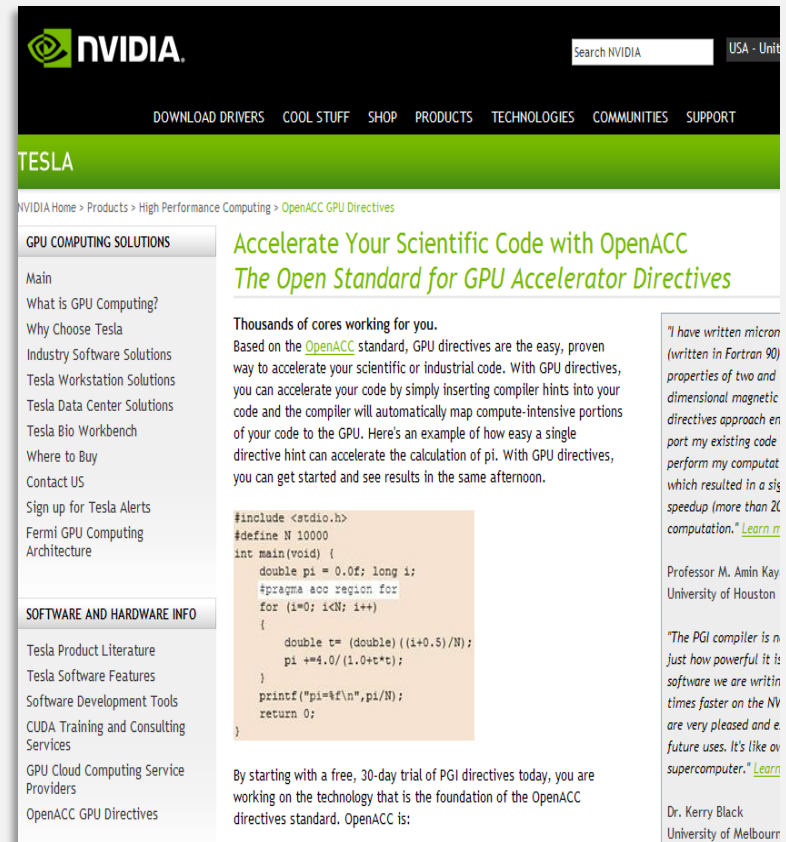
# Start Now with OpenACC Directives

Sign up for a **free trial** of  
the directives compiler  
now!

Free trial license to PGI  
Accelerator

Tools for quick ramp

[www.nvidia.com/gpudirectives](http://www.nvidia.com/gpudirectives)



The screenshot shows the NVIDIA website's 'OpenACC GPU Directives' page. The header includes the NVIDIA logo, a search bar, and navigation links like 'DOWNLOAD DRIVERS', 'COOL STUFF', 'SHOP', 'PRODUCTS', 'TECHNOLOGIES', 'COMMUNITIES', and 'SUPPORT'. A green banner below the header reads 'TESLA'. The main content area is titled 'Accelerate Your Scientific Code with OpenACC' and 'The Open Standard for GPU Accelerator Directives'. It features a sidebar with 'GPU COMPUTING SOLUTIONS' and 'SOFTWARE AND HARDWARE INFO' categories. The main text explains the benefits of OpenACC, provides a code example for calculating pi, and includes testimonials from Professor M. Amin Kay and Dr. Kerry Black. A footer note mentions a free 30-day trial of the PGI directives compiler.

**GPU COMPUTING SOLUTIONS**

- Main
- What is GPU Computing?
- Why Choose Tesla
- Industry Software Solutions
- Tesla Workstation Solutions
- Tesla Data Center Solutions
- Tesla Bio Workbench
- Where to Buy
- Contact US
- Sign up for Tesla Alerts
- Fermi GPU Computing Architecture

**SOFTWARE AND HARDWARE INFO**

- Tesla Product Literature
- Tesla Software Features
- Software Development Tools
- CUDA Training and Consulting Services
- GPU Cloud Computing Service Providers
- OpenACC GPU Directives

## Accelerate Your Scientific Code with OpenACC

### The Open Standard for GPU Accelerator Directives

Thousands of cores working for you.

Based on the [OpenACC](#) standard, GPU directives are the easy, proven way to accelerate your scientific or industrial code. With GPU directives, you can accelerate your code by simply inserting compiler hints into your code and the compiler will automatically map compute-intensive portions of your code to the GPU. Here's an example of how easy a single directive hint can accelerate the calculation of pi. With GPU directives, you can get started and see results in the same afternoon.

```
#include <stdio.h>
#define N 10000
int main(void) {
    double pi = 0.0f; long i;
    #pragma acc region for
    for (i=0; i<N; i++)
    {
        double t = (double) ((i+0.5)/N);
        pi +=4.0/(1.0+t*t);
    }
    printf("pi=%f\n",pi/N);
    return 0;
}
```

By starting with a free, 30-day trial of PGI directives today, you are working on the technology that is the foundation of the OpenACC directives standard. OpenACC is:

*"I have written micron (written in Fortran 90) properties of two and dimensional magnetic directives approach en port my existing code perform my comput which resulted in a sig speedup (more than 2x computation)." [Learn more](#)*

Professor M. Amin Kay  
University of Houston

*"The PGI compiler is n just how powerful it is software we are writin times faster on the NV are very pleased and e future uses. It's like ov supercomputer." [Learn more](#)*

Dr. Kerry Black  
University of Melbourne

© NVIDIA 2013

# 3 Ways to Accelerate Applications

Applications

Libraries

“Drop-in”  
Acceleration

OpenACC  
Directives

Easily Accelerate  
Applications

Programming  
Languages

Maximum  
Flexibility

# GPU Programming Languages

**Numerical analytics** ▶

MATLAB, Mathematica, LabVIEW

**Fortran** ▶

OpenACC, CUDA Fortran

**C** ▶

OpenACC, CUDA C

**C++** ▶

Thrust, CUDA C++

**Python** ▶

PyCUDA, Copperhead

**F#** ▶

Alea.cuBase

# Rapid Parallel C++ Development



- Resembles C++ STL
- 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
- Open source

```
// 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());
```



# Learn More

These languages are supported on all CUDA-capable GPUs.  
You might already have a CUDA-capable GPU in your laptop or desktop PC!

CUDA C/C++

<http://developer.nvidia.com/cuda-toolkit>

GPU.NET

<http://tidepowerd.com>

Thrust C++ Template Library

<http://developer.nvidia.com/thrust>

MATLAB

<http://www.mathworks.com/discovery/matlab-gpu.html>

CUDA Fortran

<http://developer.nvidia.com/cuda-toolkit>

Mathematica

<http://www.wolfram.com/mathematica/new-in-8/cuda-and-opencl-support/>

PyCUDA (Python)

<http://mathematician.de/software/pycuda>

# Getting Started

- Download CUDA Toolkit & SDK: [www.nvidia.com/getcuda](http://www.nvidia.com/getcuda)
- Nsight IDE (Eclipse or Visual Studio): [www.nvidia.com/nsight](http://www.nvidia.com/nsight)
- Programming Guide/Best Practices:
  - [docs.nvidia.com](http://docs.nvidia.com)
- Questions:
  - NVIDIA Developer forums: [devtalk.nvidia.com](http://devtalk.nvidia.com)
  - Search or ask on: [www.stackoverflow.com/tags/cuda](http://www.stackoverflow.com/tags/cuda)
- General: [www.nvidia.com/cudazone](http://www.nvidia.com/cudazone)