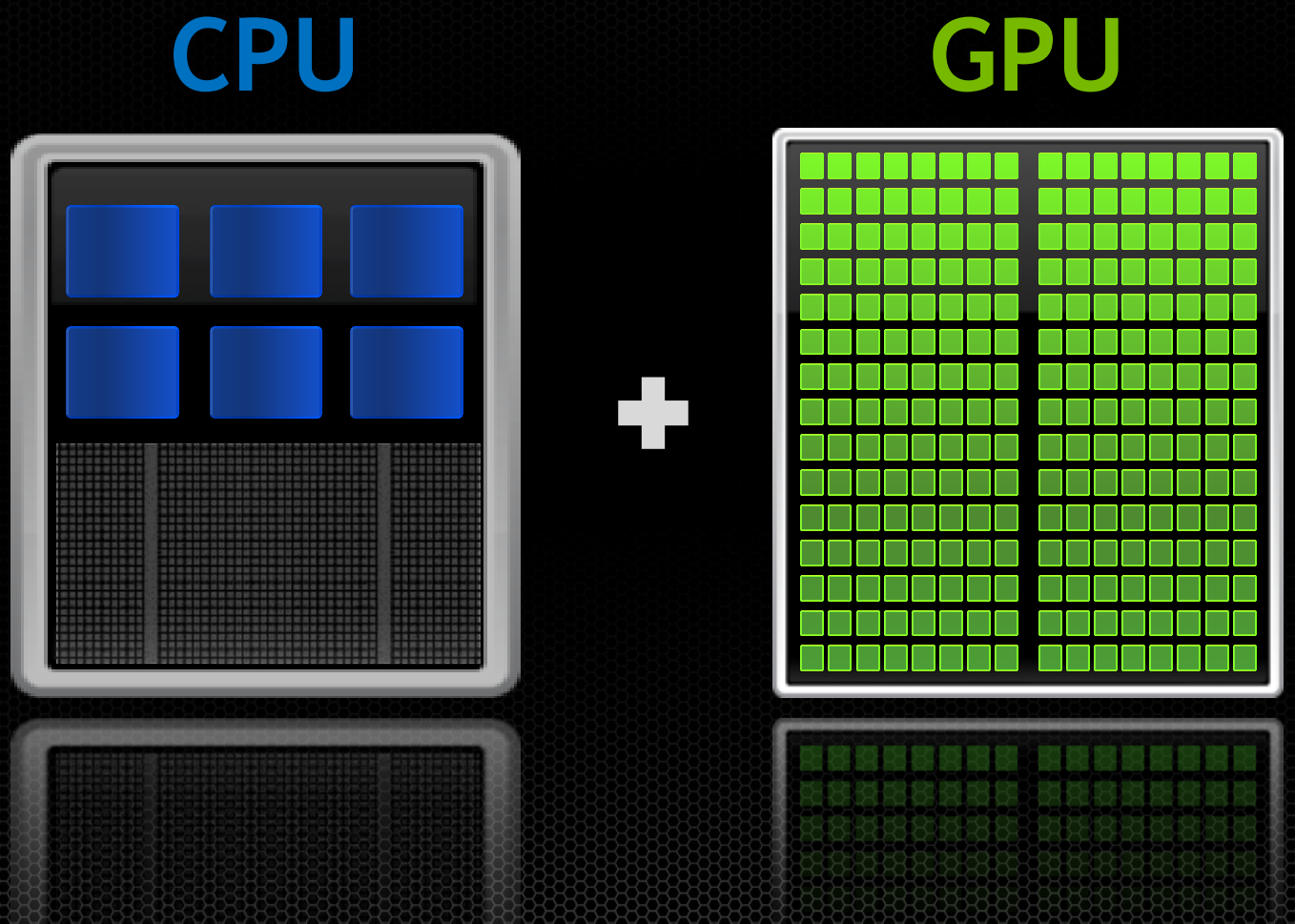
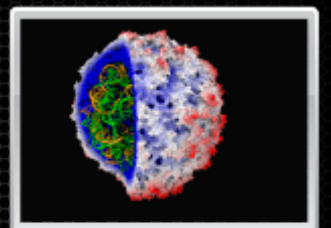

NVIDIA®
APPROACHES TO GPU COMPUTING
Libraries, OpenACC Directives, and Languages

Add GPUs: Accelerate Science Applications



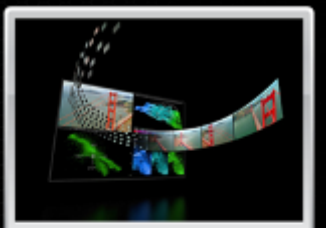
146X

Medical Imaging
U of Utah



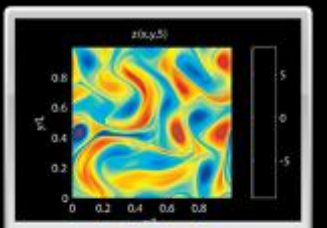
36X

Molecular Dynamics
U of Illinois, Urbana



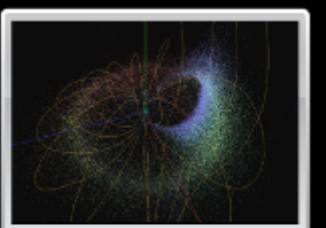
18X

Video Transcoding
Elemental Tech



50X

Matlab Computing
AccelerEyes



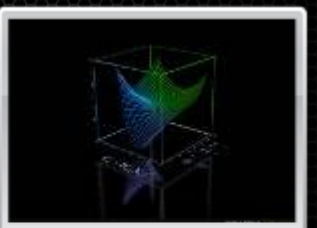
100X

Astrophysics
RIKEN

GPUs Accelerate Science

149X

Financial Simulation
Oxford



47X

Linear Algebra
Universidad Jaime

20X

3D Ultrasound
Techniscan

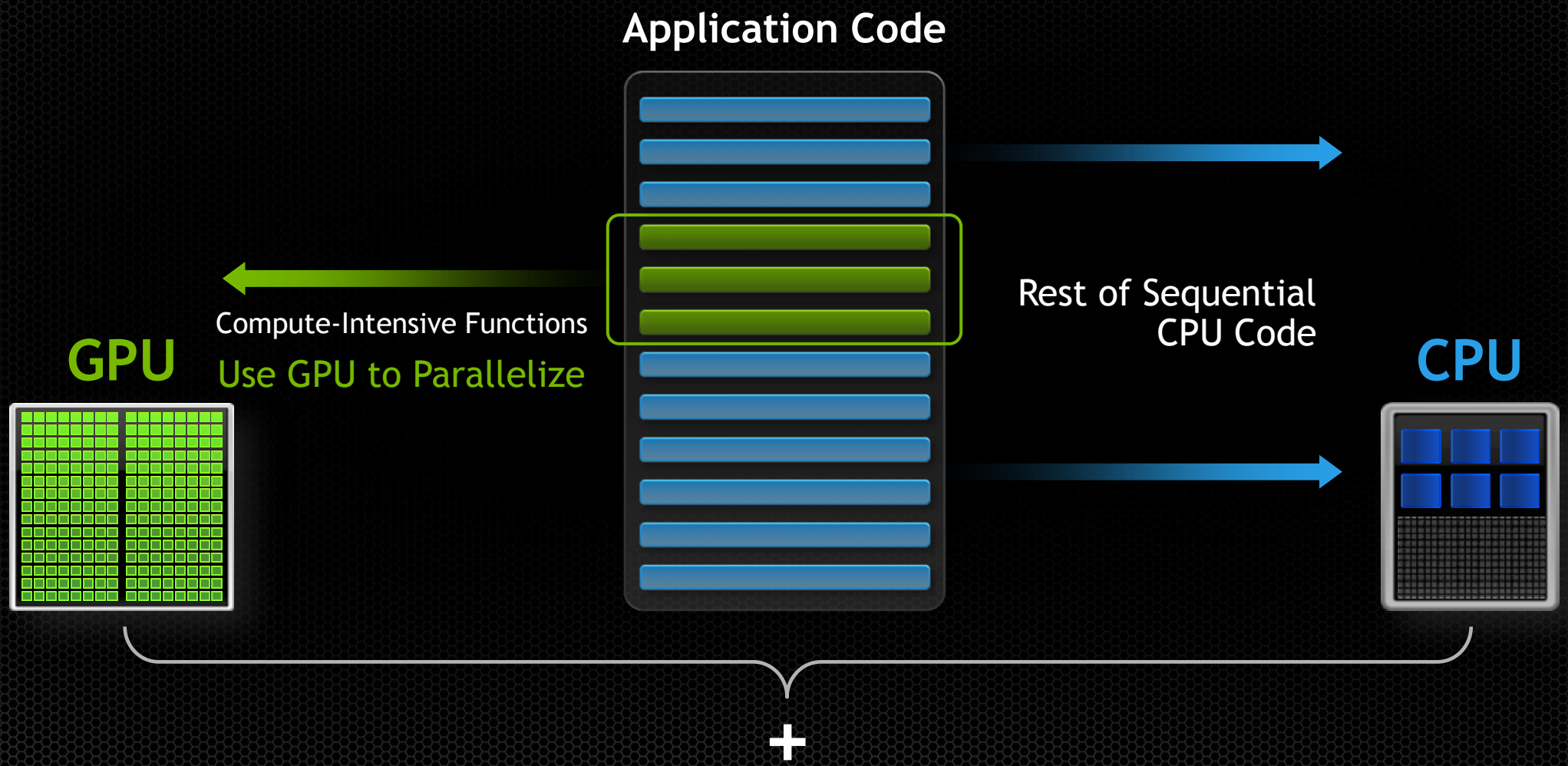
130X

Quantum Chemistry
U of Illinois, Urbana

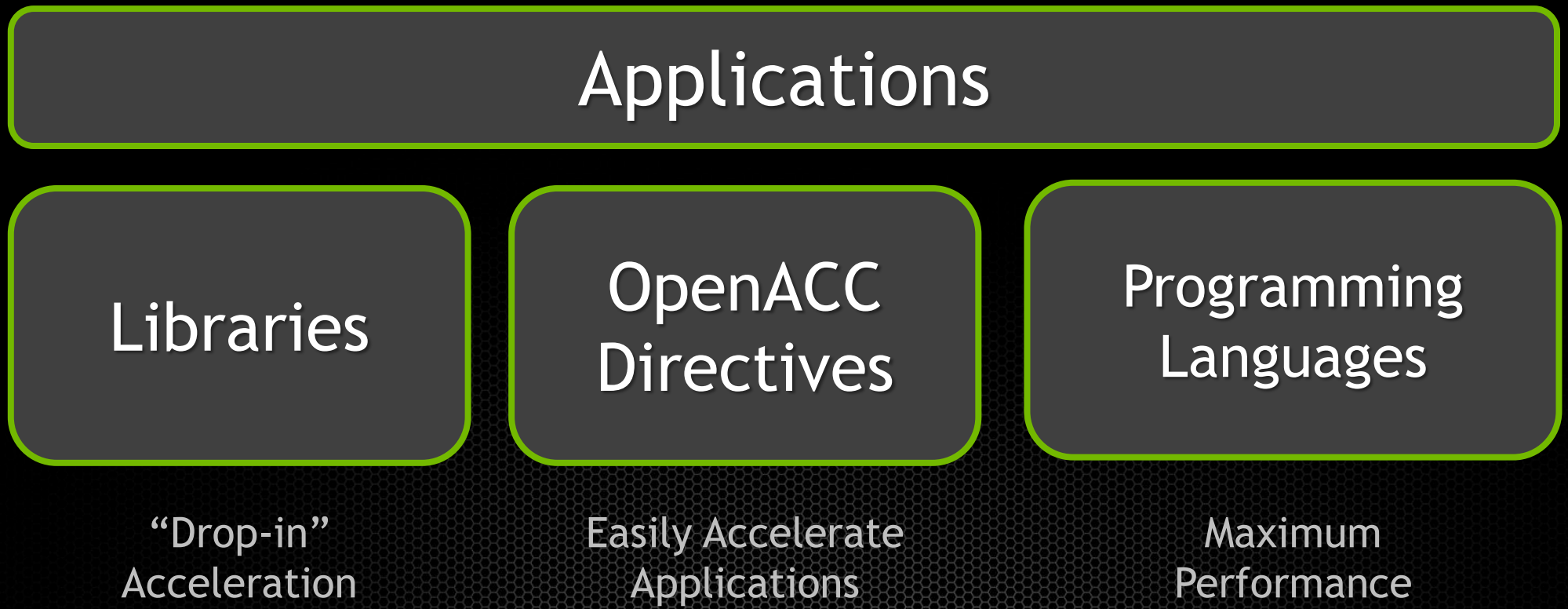
30X

Gene Sequencing
U of Maryland

Small Changes, Big Speed-up



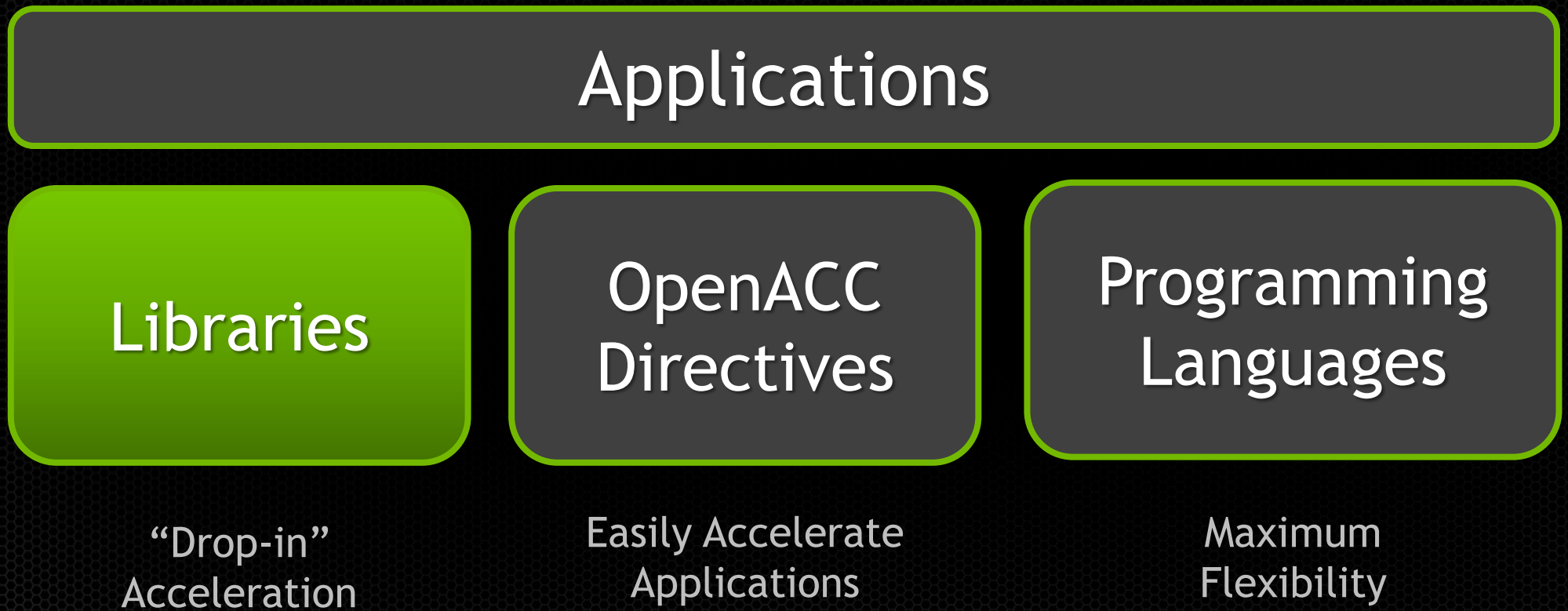
3 Ways to Accelerate Applications



Drop-in Acceleration

GPU-Accelerated Libraries

Three Ways to Accelerate Applications

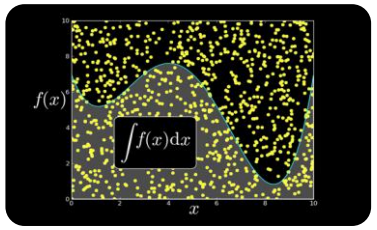


Easy, High-Quality Acceleration

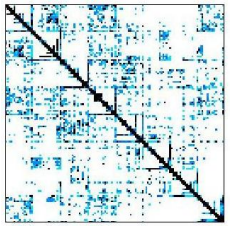
- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **"Drop-in":** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
- **Quality:** Libraries offer high-quality implementations of functions 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

GPU VSIPL

Vector Signal Image Processing

CULA tools

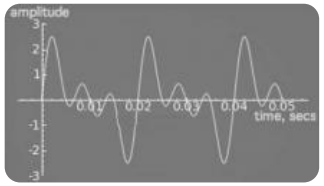
GPU Accelerated Linear Algebra

MAGMA

ICL UT

Matrix Algebra on GPU and Multicore

open source initiative



NVIDIA cuFFT

ROGUE WAVE SOFTWARE

IMSL Library

ArrayFire Matrix Computations

CUSP

Sparse Linear Algebra

open source initiative

Thrust

C++ STL Features for CUDA

open source initiative

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
```

Drop-In Acceleration (Step 1)

```
int N = 1 << 20;
```

```
// Perform SAXPY on 1M elements: y[]=a*x[]+y[]
saxpy(N, 2.0, d_x, 1, d_y, 1);
```

Drop-In Acceleration (Step 1)

```
int N = 1 << 20;
```

```
// Perform SAXPY on 1M elements: d_y[]=a*d_x[]+d_y[]
cublasSaxpy(N, 2.0, d_x, 1, d_y, 1);
```

Add “cublas” prefix and use device variables

Drop-In Acceleration (Step 2)

```
int N = 1 << 20;
cublasInit();
```

Initialize CUBLAS

```
// Perform SAXPY on 1M elements: d_y[]=a*d_x[]+d_y[]
cublasSaxpy(N, 2.0, d_x, 1, d_y, 1);
```

```
cublasShutdown();
```

Shut down CUBLAS

Drop-In Acceleration (Step 2)

```
int N = 1 << 20;
cublasInit();
cublasAlloc(N, sizeof(float), (void**)&d_x);
cublasAlloc(N, sizeof(float), (void*)&d_y);
```

Allocate device vectors

```
// Perform SAXPY on 1M elements: d_y[]=a*d_x[]+d_y[]
cublasSaxpy(N, 2.0, d_x, 1, d_y, 1);
```

```
cublasFree(d_x);
cublasFree(d_y);
cublasShutdown();
```

Deallocate device vectors

Drop-In Acceleration (Step 2)

```
int N = 1 << 20;
cublasInit();
cublasAlloc(N, sizeof(float), (void**)&d_x);
cublasAlloc(N, sizeof(float), (void*)&d_y);

cublasSetVector(N, sizeof(x[0]), x, 1, d_x, 1);
cublasSetVector(N, sizeof(y[0]), y, 1, d_y, 1);

// Perform SAXPY on 1M elements: d_y[]=a*d_x[]+d_y[]
cublasSaxpy(N, 2.0, d_x, 1, d_y, 1);

cublasGetVector(N, sizeof(y[0]), d_y, 1, y, 1);

cublasFree(d_x);
cublasFree(d_y);
cublasShutdown();
```

Transfer data to GPU

Read data back GPU

Explore the CUDA (Libraries) Ecosystem

- CUDA Tools and Ecosystem described in detail on NVIDIA Developer Zone:

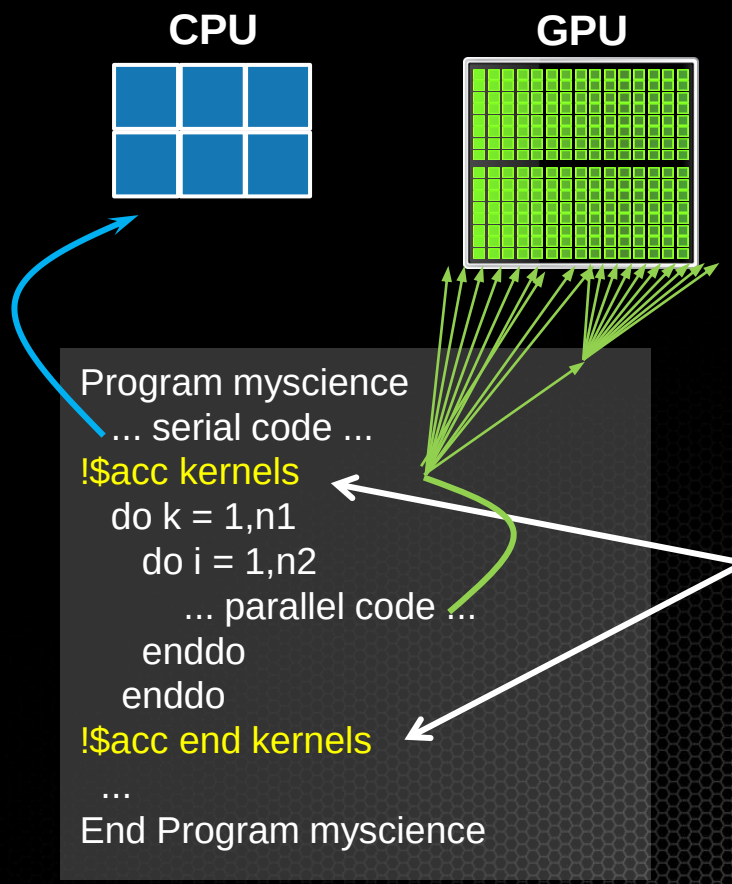
developer.nvidia.com/cuda-tools-ecosystem

Easily Accelerate Applications

GPU Computing with OpenACC Directives

3 Ways to Accelerate Applications

OpenACC Directives



Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs & multicore CPUs

OpenACC

Open Programming Standard for Parallel Computing

"OpenACC will enable programmers to easily develop portable applications that maximize the performance and power efficiency benefits of the hybrid CPU/GPU architecture of Titan."

--Buddy Bland, Titan Project Director, Oak Ridge National Lab

"OpenACC is a technically impressive initiative brought together by members of the OpenMP Working Group on Accelerators, as well as many others. We look forward to releasing a version of this proposal in the next release of OpenMP."

--Michael Wong, CEO OpenMP Directives Board

OpenACC Standard

NVIDIA

CRAY THE SUPERCOMPUTER COMPANY

PGI

CAPS

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

Two Basic Steps to Get Started

- **Step 1:** Annotate source code with directives:

```
!$acc data copy(util1,util2,util3) copyin(ip,scp2,scp2i)
  !$acc parallel loop
  …
  !$acc end parallel
!$acc end data
```

- **Step 2:** Compile & run:

```
pgf90 -ta=nvidia -Minfo=accel file.f
```

OpenACC Directives Example

```
!$acc data copy(A,Anew)
 iter=0
 do while ( err > tol .and. iter < iter_max )

  iter = iter +1
  err=0._fp_kind

!$acc kernels
   do j=1,m
    do i=1,n
     Anew(i,j) = .25_fp_kind *( A(i+1,j  ) + A(i-1,j  ) &
                               +A(i  ,j-1) + A(i  ,j+1))
     err = max( err, Anew(i,j)-A(i,j))
    end do
   end do
!$acc end kernels
   IF(mod(iter,100)==0 .or. iter == 1)    print *, iter, err
   A= Anew

 end do
!$acc end data
```

Copy arrays into GPU memory within data region

Parallelize code inside region

Close off parallel region

Close off data region, copy data back

Directives: Easy & Powerful

Real-Time Object Detection

Global Manufacturer of Navigation Systems

5x in 40 Hours

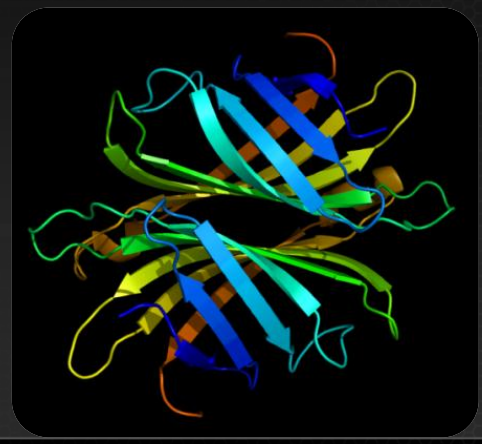
Valuation of Stock Portfolios using Monte Carlo

Global Technology Consulting Company

2x in 4 Hours

Interaction of Solvents and Biomolecules

University of Texas at San Antonio



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!

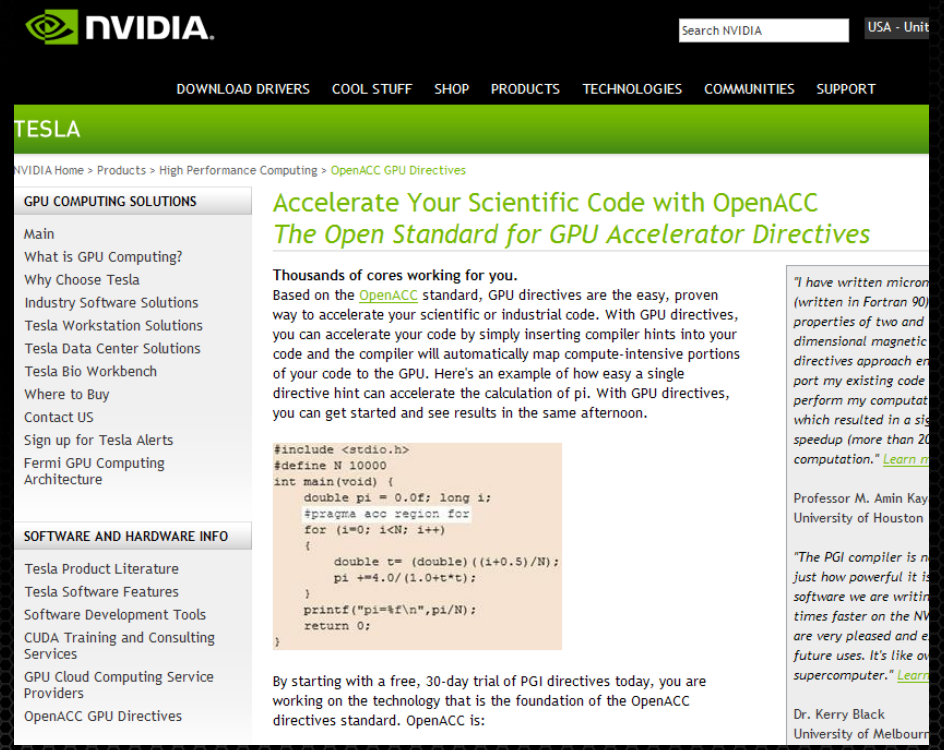
Free trial license to PGI Accelerator

Tools for quick ramp

www.nvidia.com/gpudirectives

NVIDIA

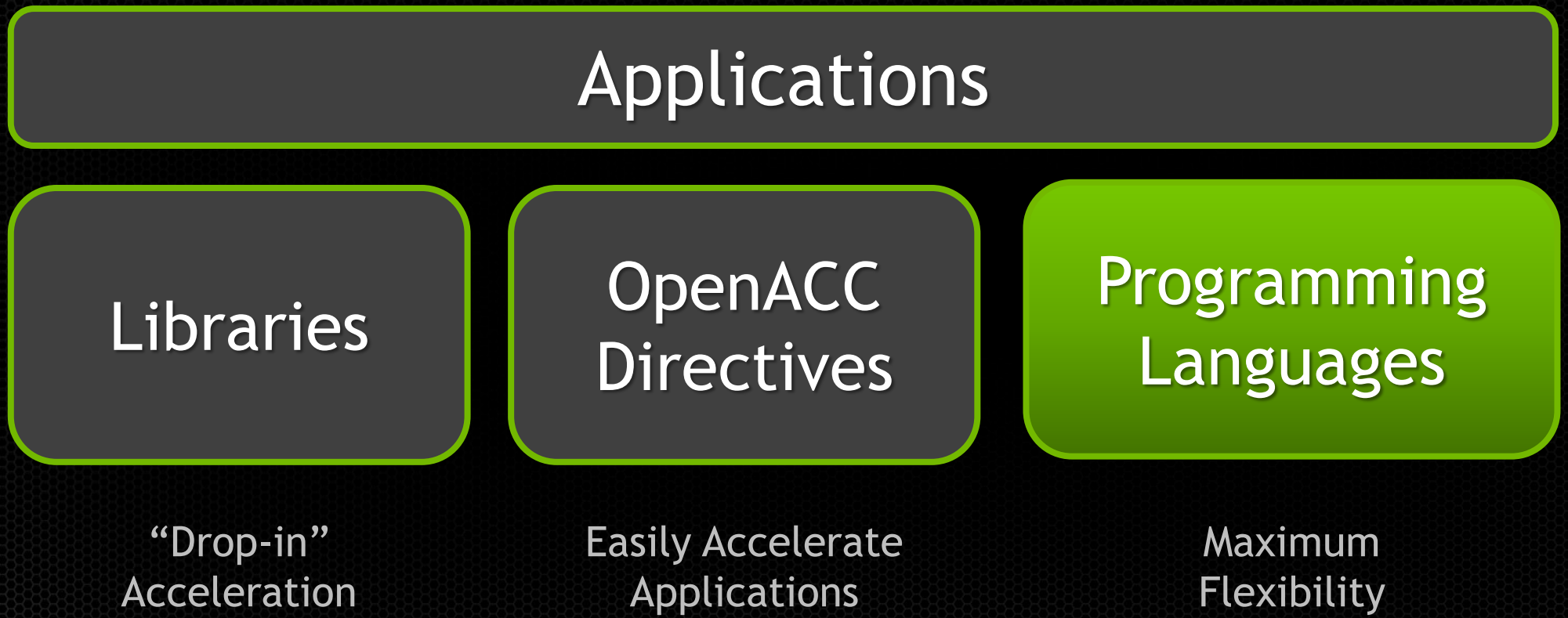
PGI®



Maximum Flexibility

Programming Languages for GPU Computing

3 Ways to Accelerate Applications



GPU Programming Languages

Numerical analytics	MATLAB, Mathematica, LabVIEW
Fortran	OpenACC, CUDA Fortran
C	OpenACC, CUDA C
C++	Thrust, CUDA C++
Python	PyCUDA, NumPy, Numba
C#	GPU.NET

CUDA C

Standard C Code

```
void saxpy_serial(int n,
                  float a,
                  float *x,
                  float *y)
{

  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_serial(4096*256, 2.0, x, y);
```

Parallel C Code

```
__global__
void saxpy_parallel(int n,
                    float a,
                    float *x,
                    float *y)
{
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_parallel<<<4096,256>>>(n,2.0,x,y);
```

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

CUDA C++: Develop Generic Parallel Code

CUDA C++ features enable sophisticated and flexible applications and middleware

- Class hierarchies
- __device__ methods
- Templates
- Operator overloading
- Functors (function objects)
- Device-side new/delete
- More…

```
template <typename T>
struct Functor {
  __device__ Functor(_a) : a(_a) {}
  __device__ T operator(T x) { return a*x; }
  T a;
}

template <typename T, typename Oper>
__global__ void kernel(T *output, int n) {
  Oper op(3.7);
  output = new T[n]; // dynamic allocation
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n)
    output[i] = op(i);  // apply functor
}
```

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

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());
```

http://developer.nvidia.com/thrust or http://thrust.googlecode.com

CUDA Fortran

- Program GPU using Fortran
  - Key language for HPC
- Simple language extensions
  - Kernel functions
  - Thread / block IDs
  - Device & data management
  - Parallel loop directives
- Familiar syntax
  - Use allocate, deallocate
  - Copy CPU-to-GPU with assignment (=)

```
module mymodule contains
  attributes(global) subroutine saxpy(n,a,x,y)
    real :: x(:), y(:), a,
    integer n, i
    attributes(value) :: a, n
    i = threadIdx%x+(blockIdx%x-1)*blockDim%x
    if (i<=n) y(i) = a*x(i) + y(i);
  end subroutine saxpy
end module mymodule

program main
  use cudafor; use mymodule
  real, device :: x_d(2**20), y_d(2**20)
  x_d = 1.0; y_d = 2.0
  call saxpy<<<4096,256>>>(2**20,3.0,x_d,y_d,)
  y = y_d
  write(*,*) 'max error=', maxval(abs(y-5.0))
end program main
```

http://developer.nvidia.com/cuda-fortran

More Programming Languages

Python ▶ PyCUDA 

C# .NET ▶ GPU.NET tidepowerd

Numerical Analytics ▶ MATLAB The Language of Technical Computing Wolfram Mathematica 8

Get Started Today

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

Thrust C++ Template Library
http://developer.nvidia.com/thrust

CUDA Fortran
http://developer.nvidia.com/cuda-toolkit

PyCUDA (Python)
http://mathema.tician.de/software/pycuda

GPU.NET
http://tidepowerd.com

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

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

CUDA Registered Developer Program

All GPGPU developers should become NVIDIA Registered Developers

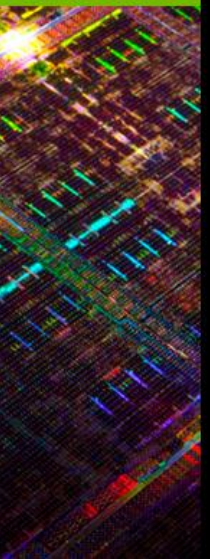
Benefits include:

- Early Access to Pre-Release Software
  - Beta software and libraries
  - CUDA 5.5 Release Candidate available now
- Submit & Track Issues and Bugs
  - Interact directly with NVIDIA QA engineers
- Benefits
  - Exclusive Q&A Webinars with NVIDIA Engineering
  - Exclusive deep dive CUDA training webinars
  - In-depth engineering presentations on pre-release software

Sign up Now: www.nvidia.com/ParallelDeveloper

GPU TECHNOLOGY CONFERENCE

GPU Technology Conference 2014
May 24-27 | San Jose, CA



The one event you can't afford to miss

- Learn about leading-edge advances in GPU computing
- Explore the research as well as the commercial applications
- Discover advances in computational visualization
- Take a deep dive into parallel programming

Ways to participate

- Speak – share your work and gain exposure as a thought leader
- Register – learn from the experts and network with your peers
- Exhibit/Sponsor – promote your company as a key player in the GPU ecosystem

www.gputechconf.com

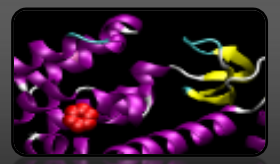
GPU Accelerated Applications

OIL & GAS

Schlumberger

Chevron

EDU/RESEARCH



OAK RIDGE National Laboratory

Chinese Academy Of Sciences

GOVERNMENT

Air Force Research Laboratory

NASA

MANUFACTURING

ANSYS

SIMULIA

GPUs

Central To Computing

DATA ANALYTICS

synerscope

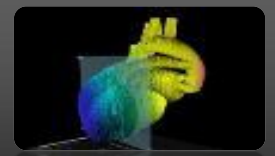
FUZZY LOGIX

MEDIA & ENTMNT.

SONY PICTURES

FOX

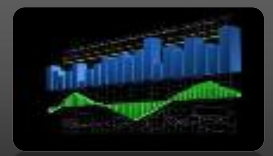
LIFE SCIENCES



BECKMAN COULTER

life technologies™

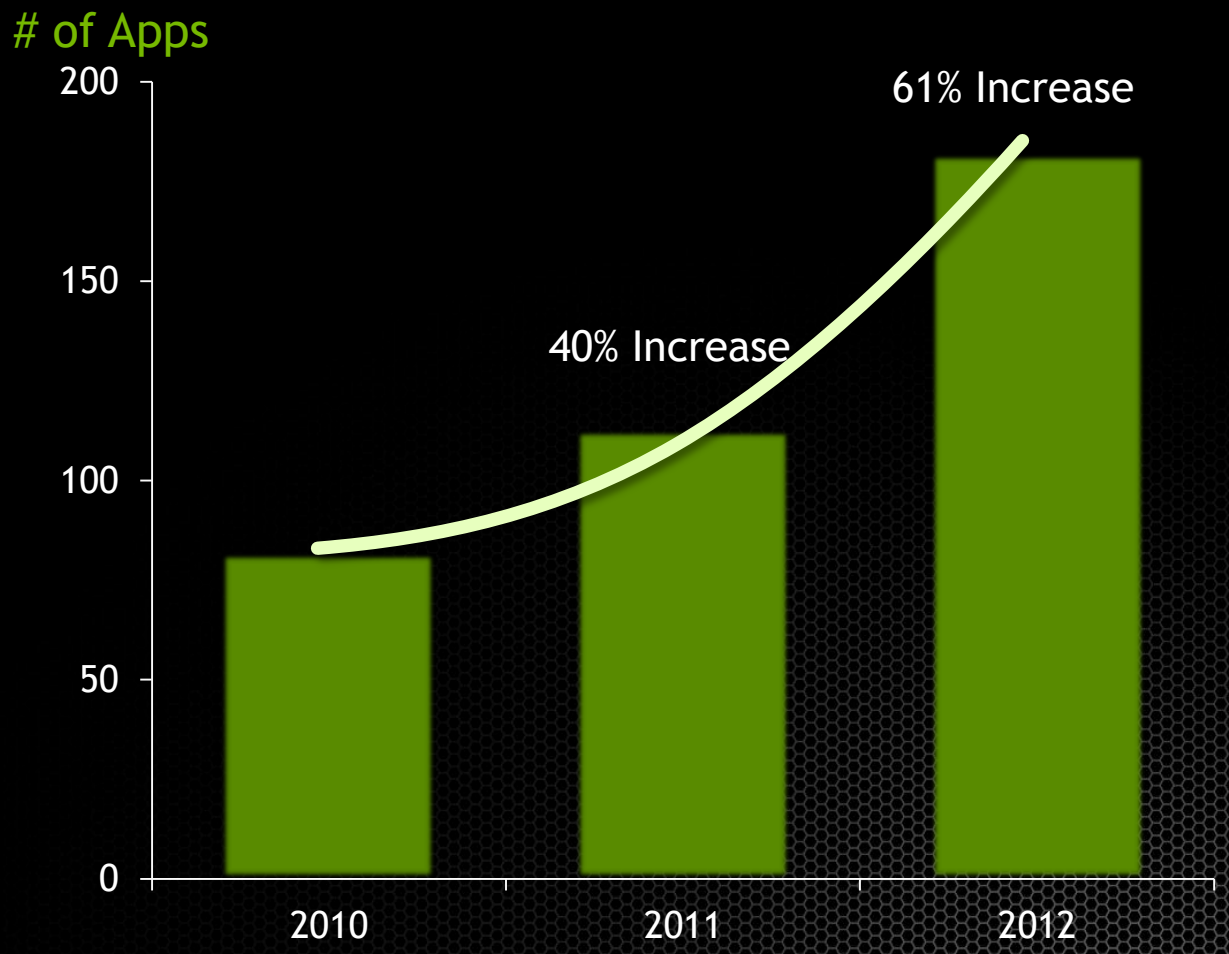
FINANCE



Bloomberg

JPMorgan

Explosive Growth of GPU Accelerated Apps



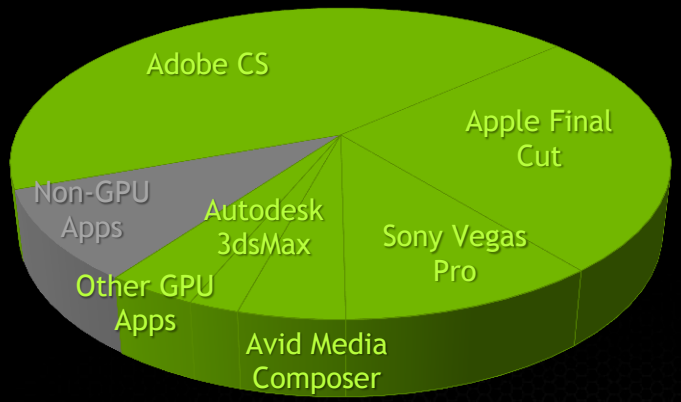
Top Scientific Apps

Computational Chemistry	AMBER CHARMM GROMACS	LAMMPS NAMD DL_POLY
Material Science	QMCPACK Quantum Espresso GAMESS-US	Gaussian NWChem VASP
Climate & Weather	COSMO GEOS-5	CAM-SE NIM WRF
Physics	Chroma Denovo GTC	GTS ENZO MILC
CAE	ANSYS Mechanical MSC Nastran SIMULIA Abaqus	ANSYS Fluent OpenFOAM LS-DYNA

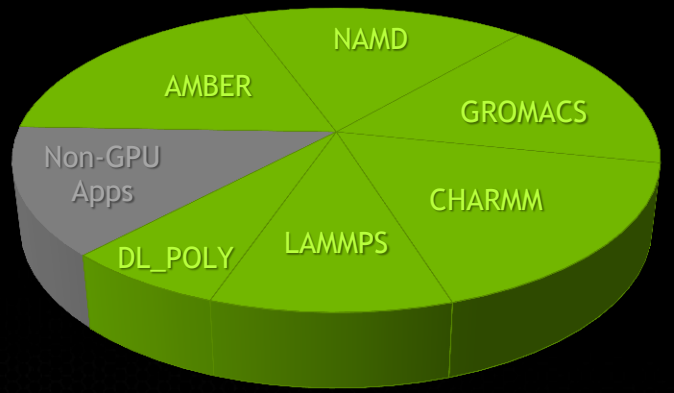
Accelerated, In Development

Top Applications Now with Built-in GPU Support

Digital Content Creation

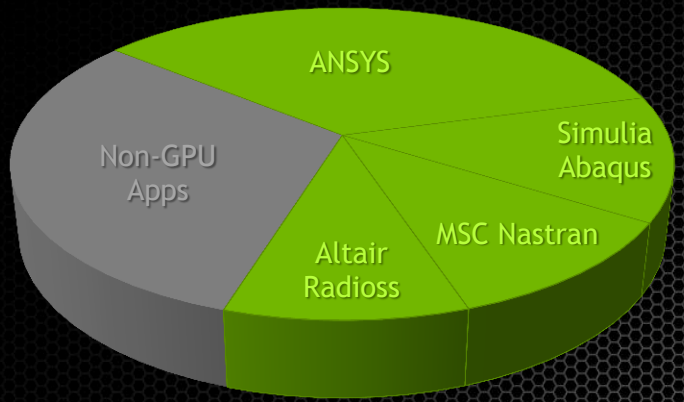


Molecular Dynamics

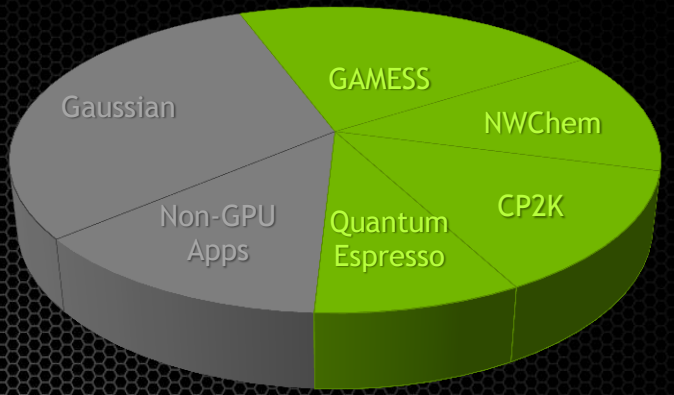


Application Market Share by Segment

Computer-Aided Engineering



Quantum Chemistry



OpenACC Accelerates Science

Weather Prediction

Swiss National Weather Agency

COSMO (Physics)

4.2x

Chemistry Research

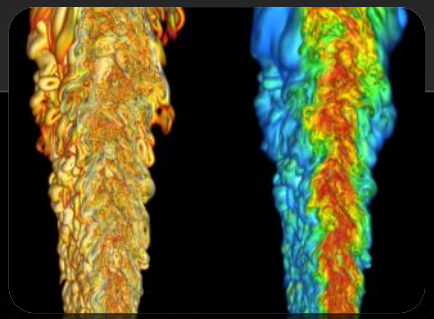
Blue Waters @ NCSA

GAMESS CCSD

3.1x

Fuel Efficiency

National Renewable Energy Lab

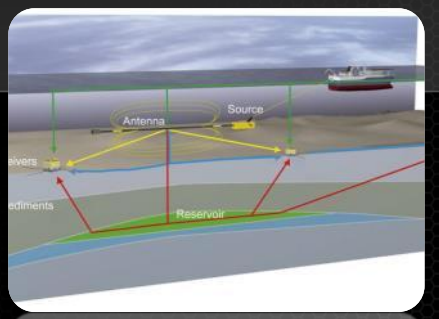


S3D

2.2x

Oil Exploration

EMGS



ELAN

3.2x

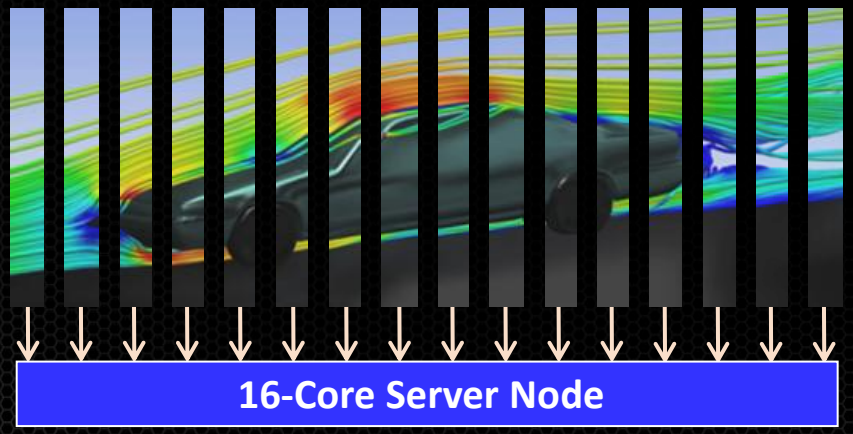
Performance Speed-up:
2x AMD6274 CPUs vs 1 AMD6274 CPU + 1 Tesla K20X

ANSYS Fluent 14.5 Multi-GPU Demonstration

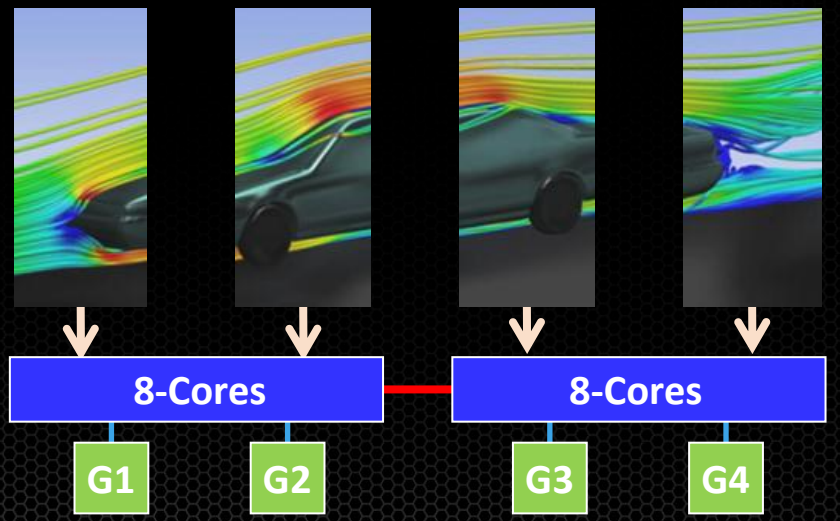
Multi-GPU Acceleration of a 16-Core ANSYS Fluent Simulation of External Aero

Xeon E5-2667 CPUs + Tesla K20X GPUs

CPU Configuration



CPU + GPU Configuration




NVIDIA
POPULAR GPU-ACCELERATED APPLICATIONS
CONTENTS
02 Research: Higher Education and Supercomputing
COMPUTATIONAL CHEMISTRY AND BIOLOGY
NUMERICAL ANALYTICS
PHYSICS
WEATHER AND CLIMATE FORECASTING
06 Defense and Intelligence
07 Computational Finance
08 Manufacturing: CAD and CAE
COMPUTER AIDED DESIGN
COMPUTATIONAL FLUID DYNAMICS
COMPUTATIONAL STRUCTURAL MECHANICS
ELECTRONIC DESIGN AUTOMATION
10 Media and Entertainment
ANIMATION, MODELING AND RENDERING
COLOR CORRECTION AND GRAIN MANAGEMENT
COMPOSITING, FINISHING AND EFFECTS
EDITING
ENCODING AND DIGITAL DISTRIBUTION
ON-AIR GRAPHICS
ON-SET, REVIEW AND STEREO TOOLS
SIMULATION
WEATHER GRAPHICS
14 Oil and Gas
Research: Higher Education and Supercomputing
COMPUTATIONAL CHEMISTRY AND BIOLOGY
Bioinformatics
BarraCUDA Sequence mapping software Alignment of short sequencing reads 6-10x T 2075, 2090, K10, K20, K20X Yes Available now Version 0.6.2
CUDASW++ Open source software for Smith-Waterman protein database searches on GPUs Parallel search of Smith-Waterman database 10-50x T 2075, 2090, K10, K20, K20X Yes Available now Version 2.0.8
CUSHAW Parallelized short read aligner Parallel, accurate long read aligner - gapped alignments to large genomes 10x T 2075, 2090, K10, K20, K20X Yes Available now Version 1.0.40
GPU-BLAST Local search with fast k-tuple heuristic Protein alignment according to blastp, multi cpu threads 3-4x T 2075, 2090, K10, K20, K20X Single only Available now Version 2.2.26
GPU-HMMER Parallelized local and global search with profile Hidden Markov models Parallel local and global search of Hidden Markov Models 60-100x T 2075, 2090, K10, K20, K20X Yes Available now Version 2.3.2
mCUDA-MEME Ultrafast scalable motif discovery algorithm based on MEME Scalable motif discovery algorithm based on MEME 4-10x T 2075, 2090, K10, K20, K20X Yes Available now Version 3.0.12
SeqNFind A GPU Accelerated Sequence Analysis Toolset Reference assembly, blast, smith-waterman, hmm, de novo assembly 400x T 2075, 2090, K10, K20, K20X Yes Available now
UGENE Opensource Smith-Waterman for SSE/CUDA, Suffix array based repeats finder and dotplot Fast short read alignment 6-8x T 2075, 2090, K10, K20, K20X Yes Available now Version 1.11
WideLM Fits numerous linear models to a fixed design and response Parallel linear regression on multiple similarly-shaped models 150x T 2075, 2090, K10, K20, K20X Yes Available now Version 0.1-1
Molecular Dynamics
Abalone Models molecular dynamics of biopolymers for simulations of proteins, DNA and ligands Simulations (on 1060 GPU) 4-29x T 2075, 2090, K10, K20, K20X Single Only Available now Version 1.8.48
ACEMD GPU simulation of molecular mechanics force fields, implicit and explicit solvent Written for use on GPUs 160 ns/day GPU version only T 2075, 2090, K10, K20, K20X Yes Available now
AMBER Suite of programs to simulate molecular dynamics on biomolecule PMEMD: explicit and implicit solvent 89.44 ns/day JAC NVE T 2075, 2090, K10, K20, K20X Yes Available now Version 12 + bugfix9
DL-POLY Simulate macromolecules, polymers, ionic systems, etc on a distributed memory parallel computer Two-body forces, Link-cell pairs, Ewald SPME forces, Shake VV 4x T 2075, 2090, K10, K20, K20X Yes Available now, Version4.0 Source only
CHARMM MD package to simulate molecular dynamics on biomolecule. Implicit (5x), Explicit (2x) Solvent via OpenMM TBD T 2075, 2090, K10, K20, K20X Yes In Development 04/12
GROMACS Simulation of biochemical molecules with complicated bond interactions Implicit (5x), Explicit(2x) solvent 165 ns/Day DHFR T 2075, 2090, K10, K20, K20X Single only Available now Version 4.6 in 04/12
HOOMD-Blue Particle dynamics package written grounds up for GPUs Written for GPUs 2x T 2075, 2090, K10, K20, K20X Yes Available now
LAMMPS Classical molecular dynamics package Lennard-Jones, Morse, Buckingham, CHARMM, Tabulated, Course grain SDK, Anisotropic Gay-Bern, RE-squared, "Hybrid" combinations 3-18x T 2075, 2090, K10, K20, K20X Yes Available now
AMD Designed for high-performance simulation of large molecular systems 100M atom capable 6.44 ns/days STMV 585x 2050s T 2075, 2090, K10, K20, K20X Yes Available now, Version 2.9
enMM Library and application for molecular dynamics for HPC with GPUs Implicit and explicit solvent, custom forces Implicit: 127-213 ns/day; Explic 18-55 ns/ DHFR T 2075, 2090, K10, K20, K20X Yes Available now Version 4.1.1
207 GPU-Accelerated Applications
www.nvidia.com/appscatalog