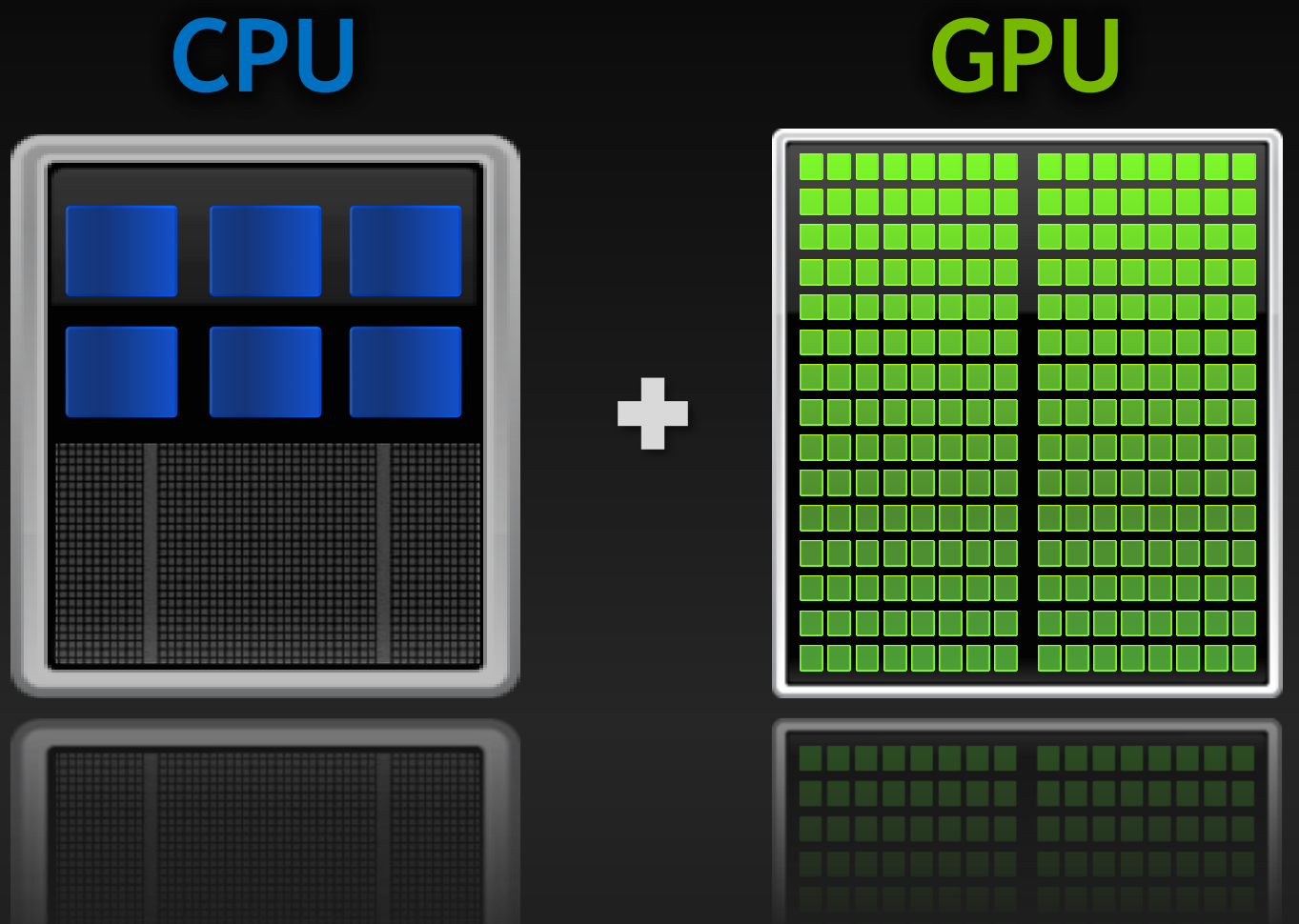
Introduction to GPU Computing

chandlerz@nvidia.com 周国峰

Wuhan University 2017/10/13

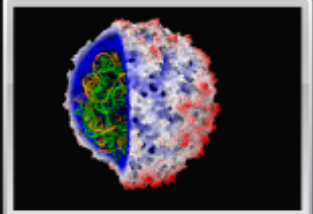
- GPU and Its Application
- 3 Ways to Develop Your GPU APP
- An Example to Show the Developments

Add GPUs: Accelerate Science Applications



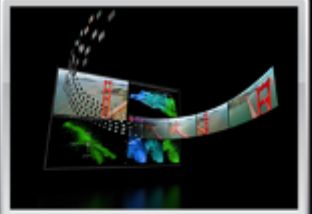
146X

Medical Imaging
U of Utah



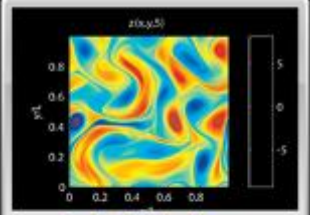
36X

Molecular Dynamics
U of Illinois, Urbana



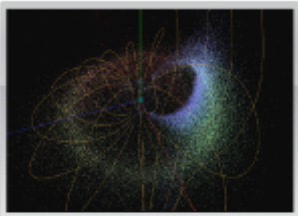
18X

Video Transcoding
Elemental Tech



50X

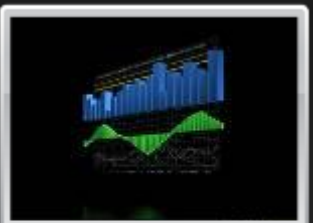
Matlab Computing
AccelerEyes



100X

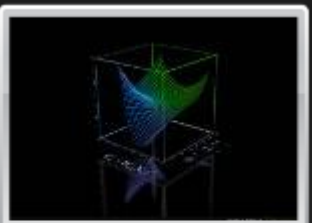
Astrophysics
RIKEN

GPUs Accelerate Science



149X

Financial Simulation
Oxford



47X

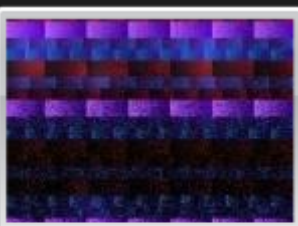
Linear Algebra
Universidad Jaime

20X

3D Ultrasound
Techniscan

130X

Quantum Chemistry
U of Illinois, Urbana



30X

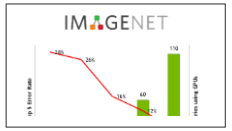
Gene Sequencing
U of Maryland

GPUs Accelerate Deep Learning

COMPUTER VISION

OBJECT DETECTION

IMAGE CLASSIFICATION



SPEECH & AUDIO

VOICE RECOGNITION

LANGUAGE TRANSLATION

NATURAL LANGUAGE PROCESSING

RECOMMENDATION ENGINES

SENTIMENT ANALYSIS

DEEP LEARNING FRAMEWORKS

Caffe

Chainer

DL4J
Deeplearning4j

Mocha.jl
julia

KERAS

Microsoft CNTK

mxnet

Purine

TensorFlow

torch

MatConvNet

MINERVA

OpenDeep

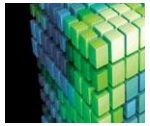
Pylearn2

theano

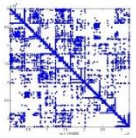
NVIDIA DEEP LEARNING SDK

cuDNN

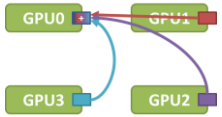
cuBLAS



cuSPARSE



NCCL



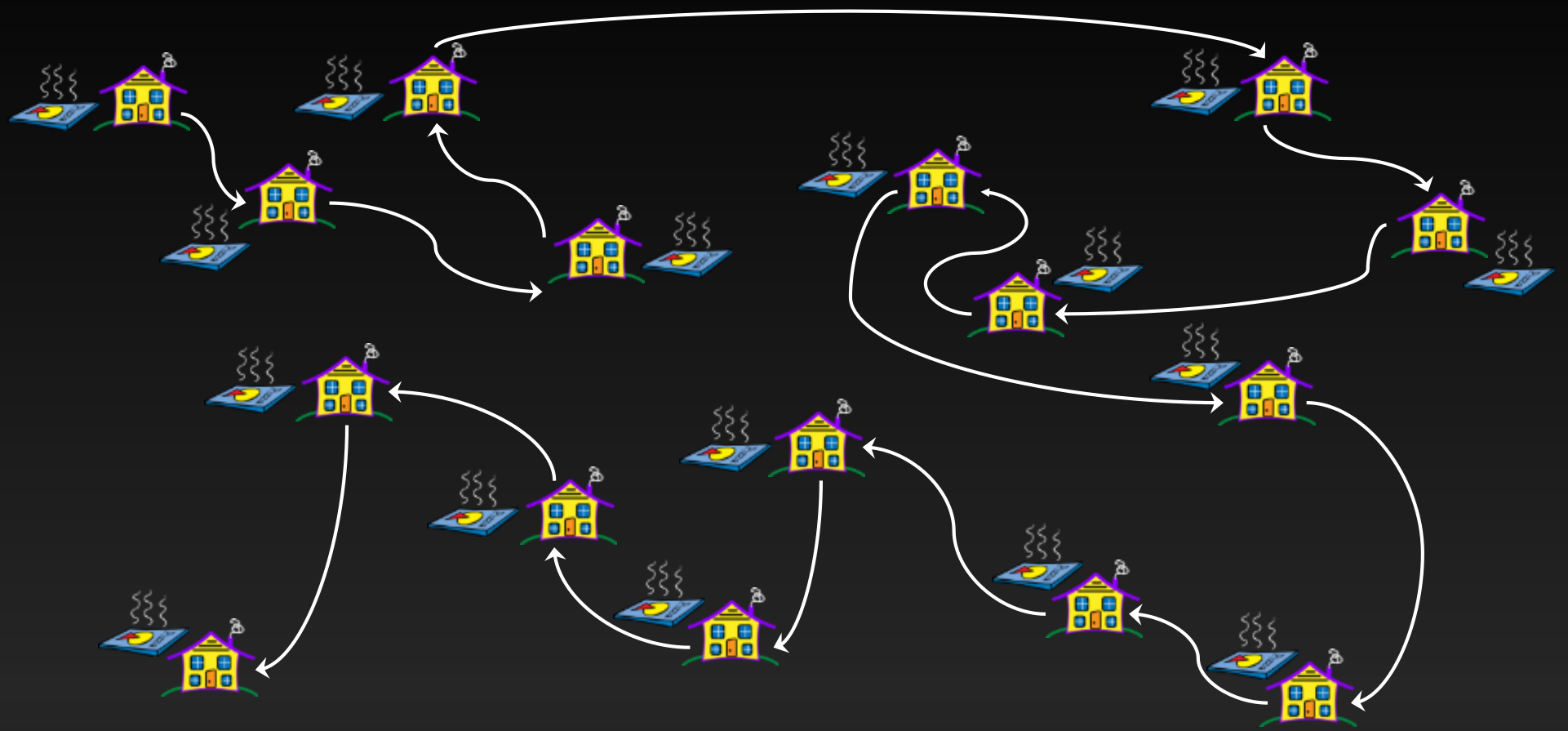
TensorRT

developer.nvidia.com/deep-learning-software

CPU Deliver Pizza

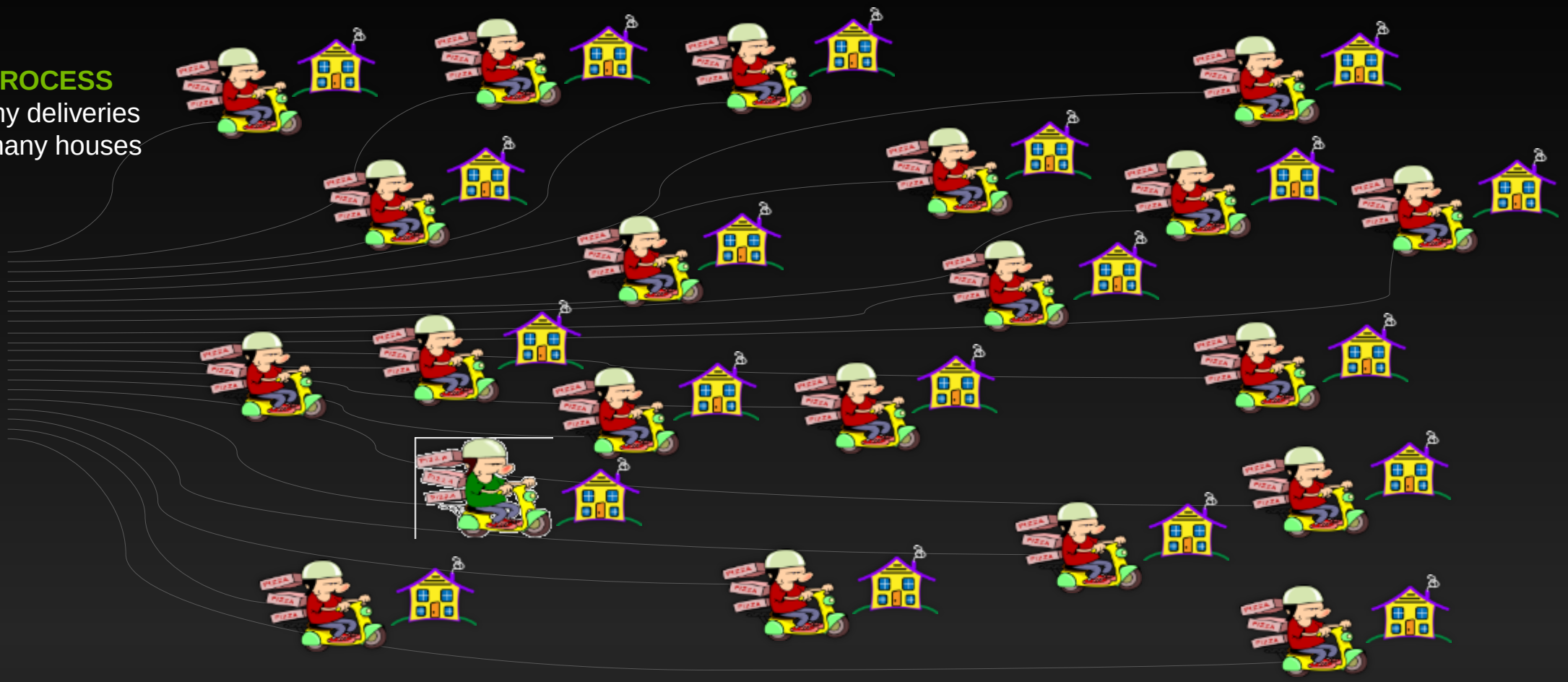
PROCESS

Delivery truck delivers one pizza and then moves to next house

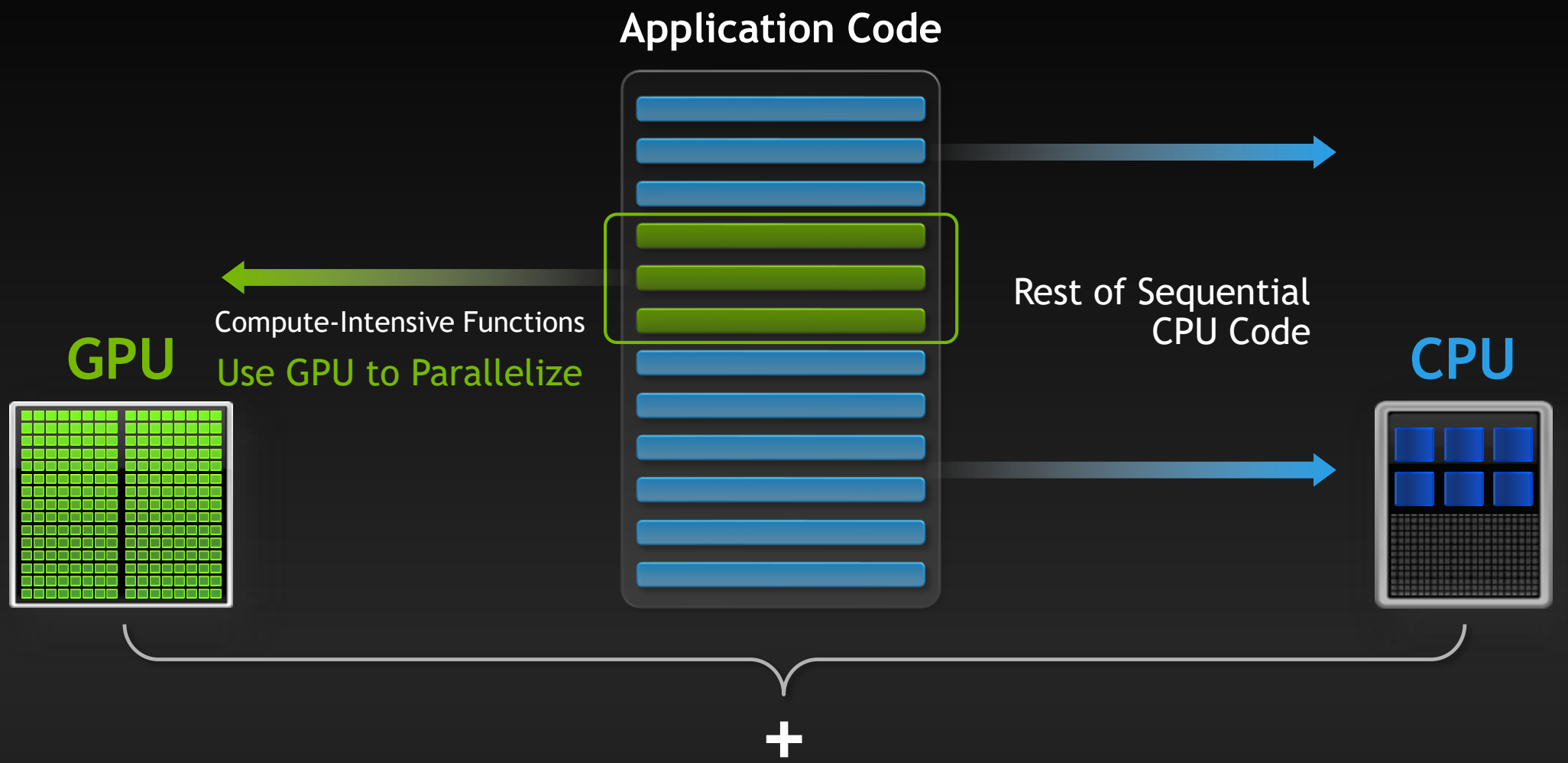


NVIDIA GPU Deliver Pizza

PROCESS
Many deliveries
to many houses



Small Changes, Big Speed-up



3 Ways to Accelerate Applications

Applications

Libraries	OpenACC Directives	Programming Languages
"Drop-in" Acceleration	Easily Accelerate Applications	Maximum Performance

3 Ways to Accelerate Applications

Applications

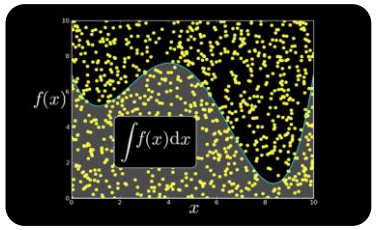
Libraries	OpenACC Directives	Programming Languages
"Drop-in" Acceleration	Easily Accelerate Applications	Maximum Flexibility

Libraries: Easy, High-Quality Acceleration

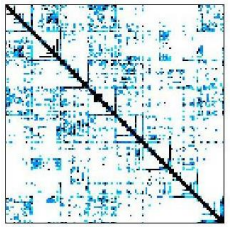
- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **"Drop-in":** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
- **Quality:** Libraries offer high-quality implementations of functions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

GPU-accelerated Libraries for HPC

NVIDIA cuBLAS



NVIDIA cuRAND



NVIDIA cuSPARSE

NVIDIA NPP

GPU VSIPL

Vector Signal Image Processing

CULA tools

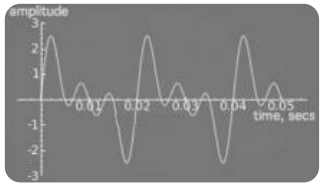
GPU Accelerated Linear Algebra

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

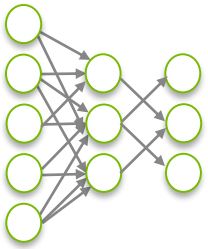
C++ STL Features for CUDA

open source initiative

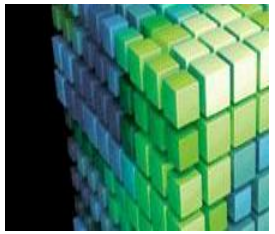
GPU-accelerated Libraries for Deep Learning

NVIDIA DEEP LEARNING SDK

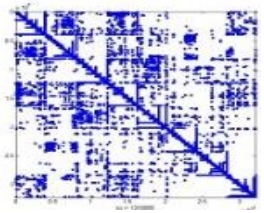
cuDNN



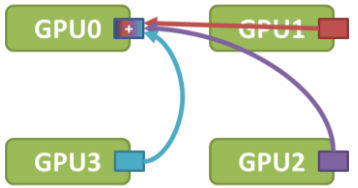
cuBLAS



cuSPARSE



NCCL



TensorRT

3 Ways to Accelerate Applications

Applications

Libraries

OpenACC Directives

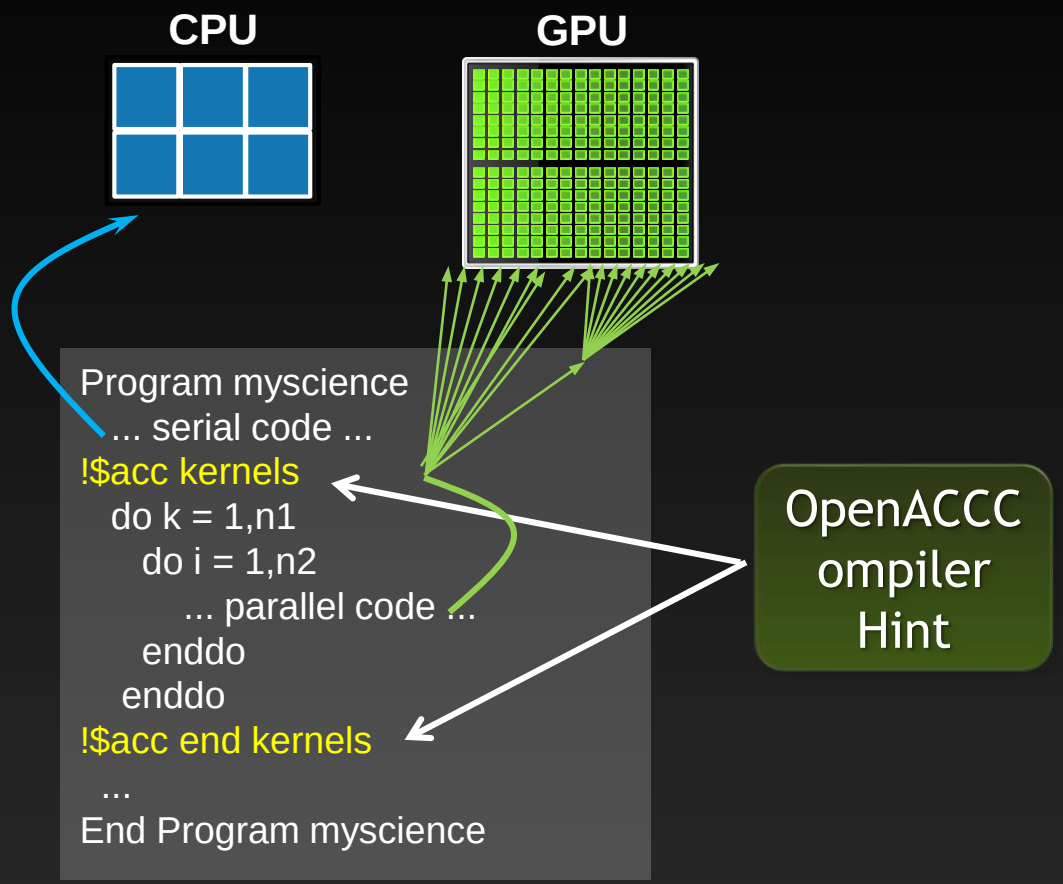
Programming Languages

"Drop-in" Acceleration

Easily Accelerate Applications

Maximum Flexibility

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

2 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

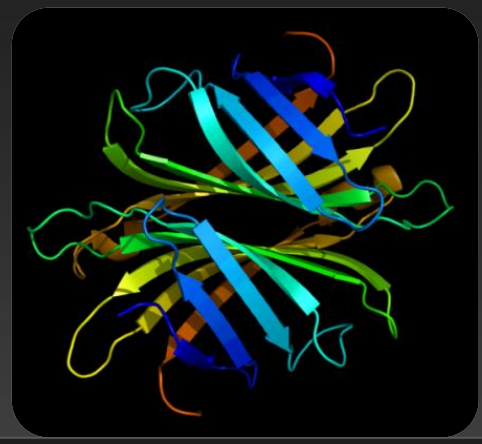
Valuation of Stock Portfolios using Monte Carlo

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*

3 Ways to Accelerate Applications

Applications

Libraries

OpenACC Directives

Programming Languages

"Drop-in" Acceleration

Easily Accelerate Applications

Maximum Flexibility

GPU Programming Languages

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

Single precision Alpha X Plus Y (SAXPY)

Part of Basic Linear Algebra Subroutines (BLAS) Library

$$z = \alpha x + y$$

x, y, z : vector

α : scalar

GPU SAXPY in multiple languages and libraries

A menagerie* of possibilities, not a tutorial

*technically, a *program chrestomathy*: http://en.wikipedia.org/wiki/Chrestomathy

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 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/


Thank you
developer.nvidia.com
NVIDIA