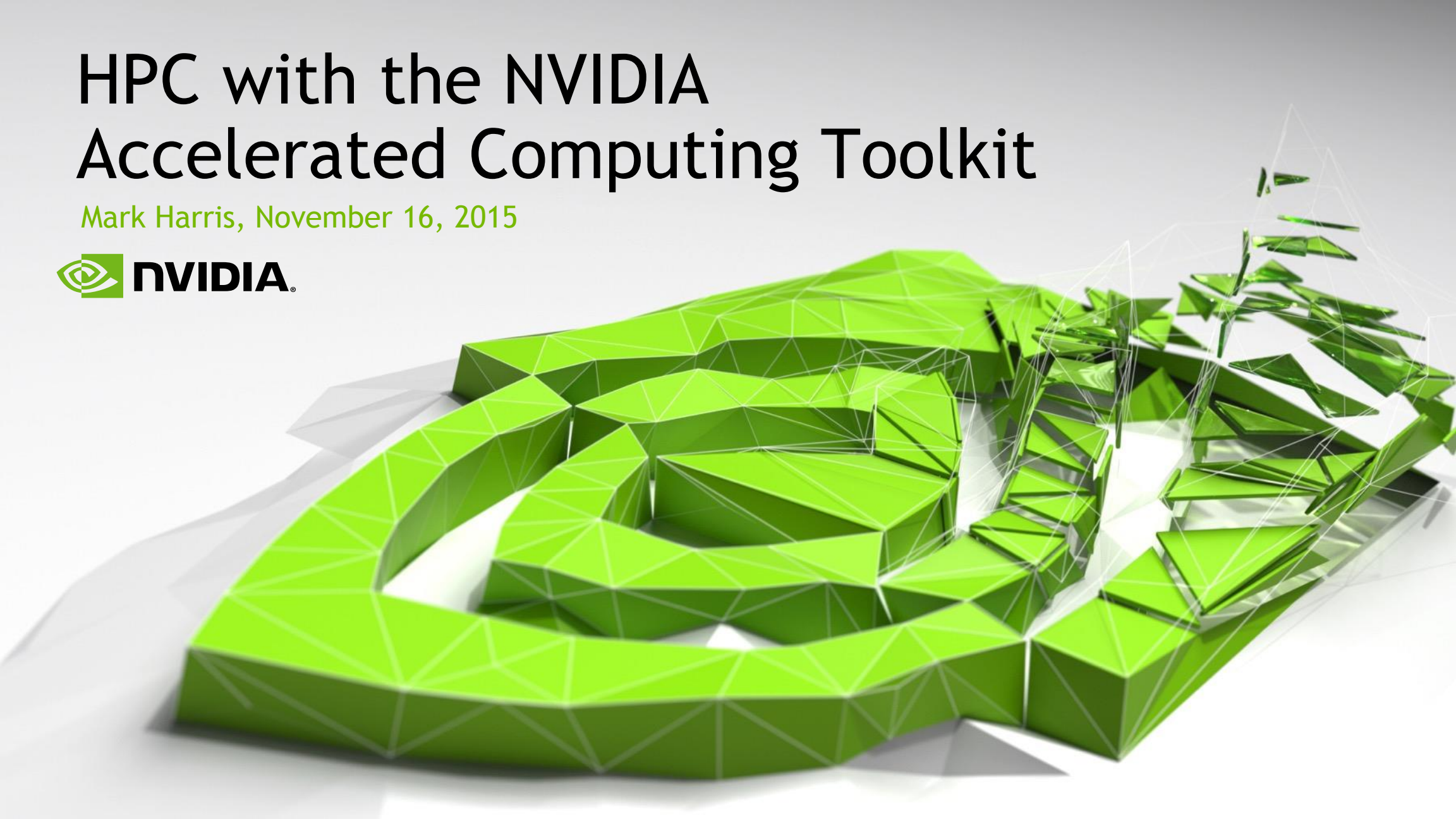
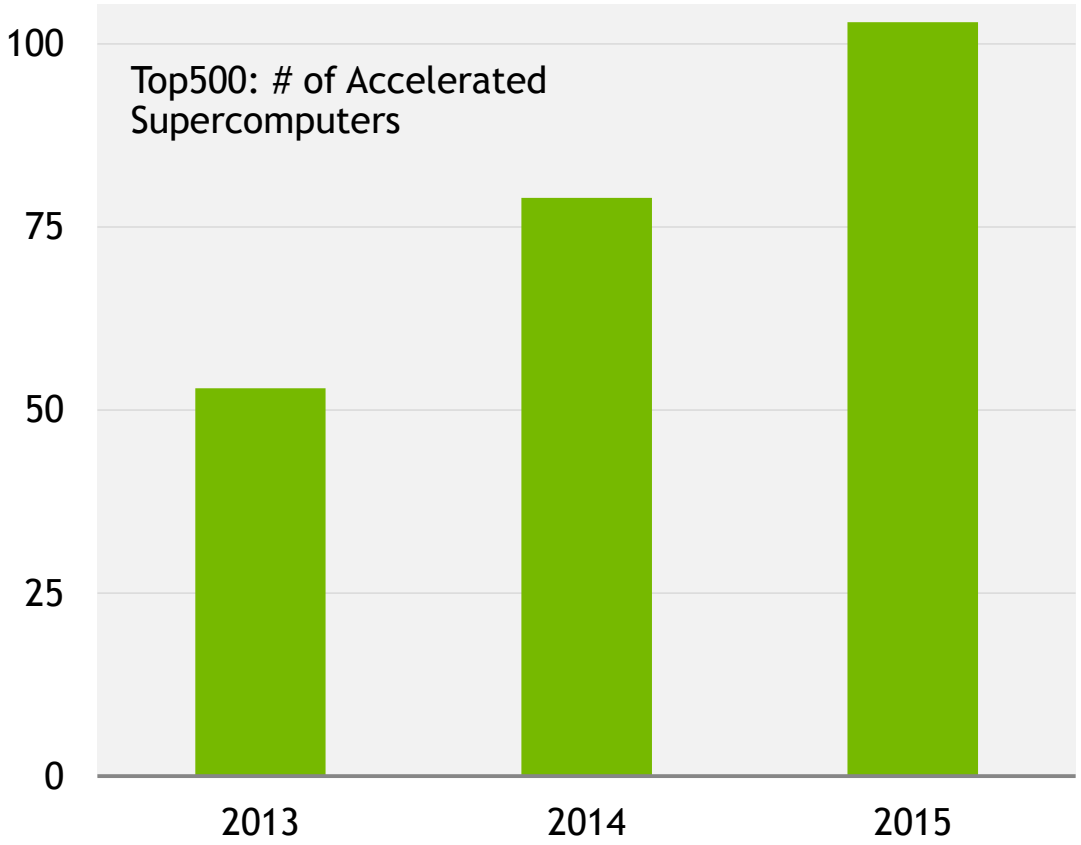
HPC with the NVIDIA Accelerated Computing Toolkit

Mark Harris, November 16, 2015



Accelerators Surge in World's Top Supercomputers



100+ accelerated systems now on Top500 list

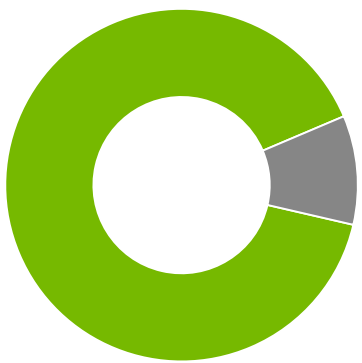
1/3 of total FLOPS powered by accelerators

NVIDIA Tesla GPUs sweep 23 of 24 new accelerated supercomputers

Tesla supercomputers growing at 50% CAGR over past five years

70% OF TOP HPC APPS NOW ACCELERATED

INTERSECT360 SURVEY OF TOP APPS



Top 10 HPC Apps

90% Accelerated

Top 50 HPC Apps

70% Accelerated

Intersect360, Nov 2015 "HPC Application Support for GPU Computing"

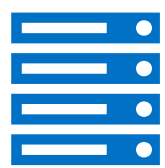
KEY MATERIALS APP NOW ACCELERATED

VASP
b-initio
ackage
imulation
ienna

Typically Consumes
10-25%
of HPC System

1 Dual K80 Server

1.3x



4 Dual CPU Servers

1.0x

Dual-socket Xeon E5-2690 v2 3GHz, Dual Tesla K80, FDR InfiniBand Dataset: NiAl-MD Blocked Davidson

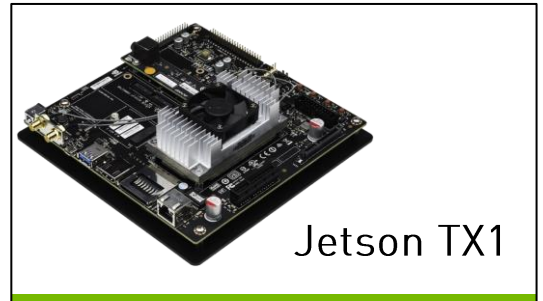
Accelerated Computing Available Everywhere

NVIDIA Accelerated Computing Toolkit

Supercomputer

Datacenter - Cloud

Workstation - Notebook



Embedded - Automotive

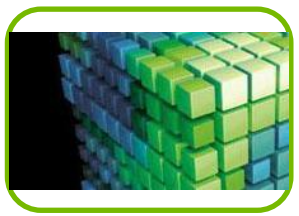
NVIDIA Accelerated Computing Toolkit

Everything You Need to Accelerate Your Applications

Libraries

FFTW

BLAS



DNN

cuDNN

Language Solutions

C/C++

NVIDIA CUDA

Fortran

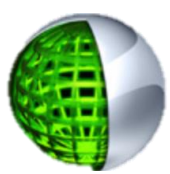
CUDA Fortran

OpenACC

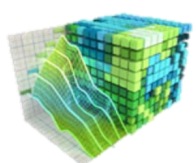
OpenACC
Directives for Accelerators

Tools

NSight IDE

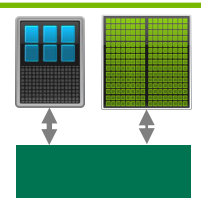


Visual Profiler

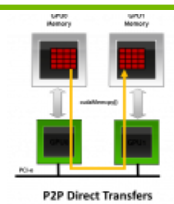


System Interfaces

Unified Memory



GPUDirect



Training

Hands-on Labs

Docs & Samples

developer.nvidia.com

Get Started Quickly

Machine Learning Tools

Design, Train and Deploy Deep Neural Networks

developer.nvidia.com/deep-learning

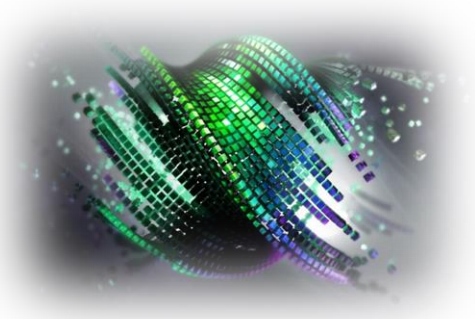
OpenACC Toolkit

OpenACC
Directives for Accelerators

Simple way to accelerate scientific code

developer.nvidia.com/openacc

CUDA Toolkit



High-Performance C/C++ Applications

developer.nvidia.com/cuda-toolkit

Why is Machine Learning Hot Now?

Big Data Availability

facebook | 350 millions images uploaded per day

Walmart | 2.5 Petabytes of customer data hourly

YouTube | 300 hours of video uploaded every minute

New Techniques

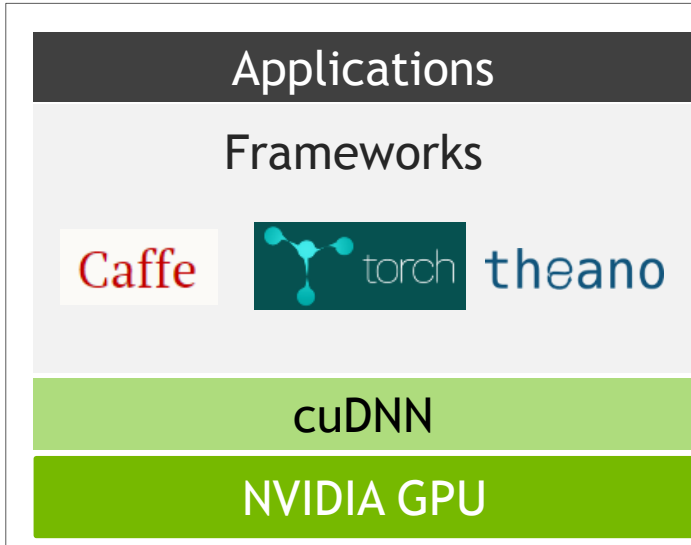
GPU Acceleration

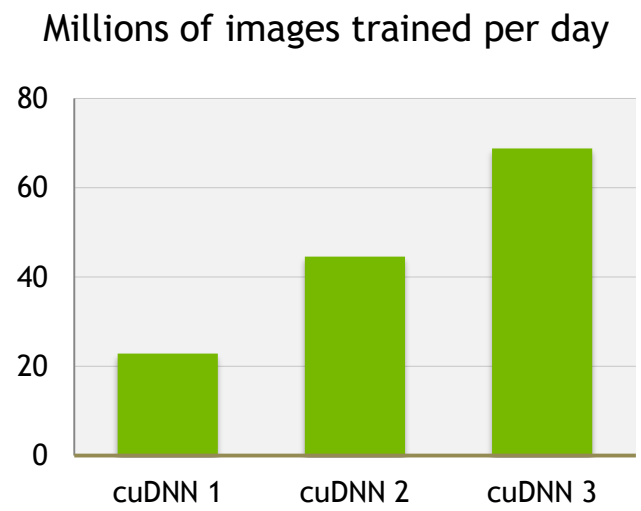
cuDNN

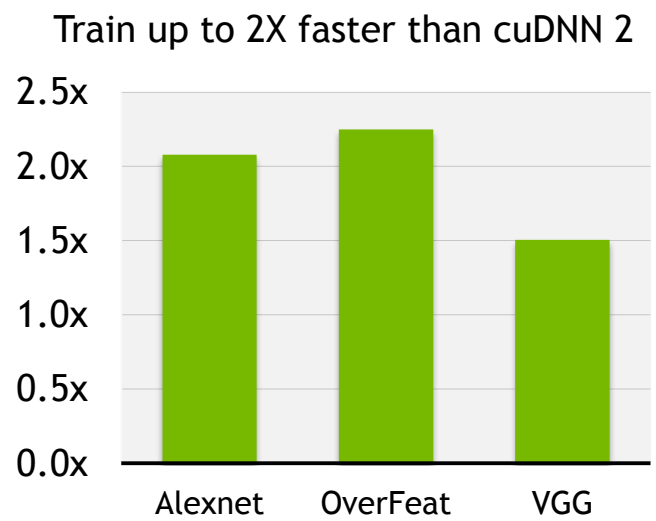
Deep Learning Primitives

Igniting
Artificial Intelligence

- Drop-in Acceleration for major Deep Learning frameworks: Caffe, Theano, Torch
- Up to 2x faster on AlexNet than baseline Caffe GPU implementation

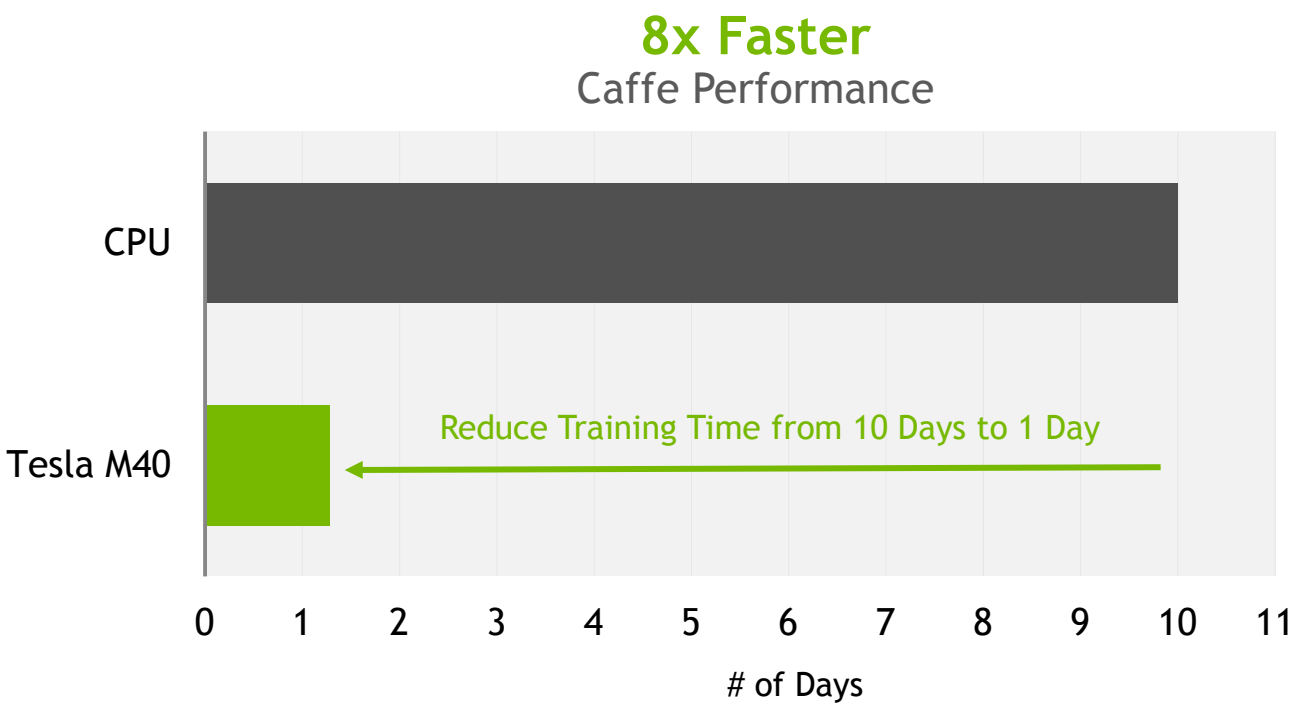






Tesla M40

World's Fastest Accelerator for Deep Learning



CUDA Cores	3072
Peak SP	7 TFLOPS
GDDR5 Memory	12 GB
Bandwidth	288 GB/s
Power	250W

Caffe Benchmark: AlexNet training throughput based on 20 iterations, CPU: E5-2680v3 @ 2.70GHz. 64GB System Memory, CentOS 6.2

NVIDIA® DIGITS™

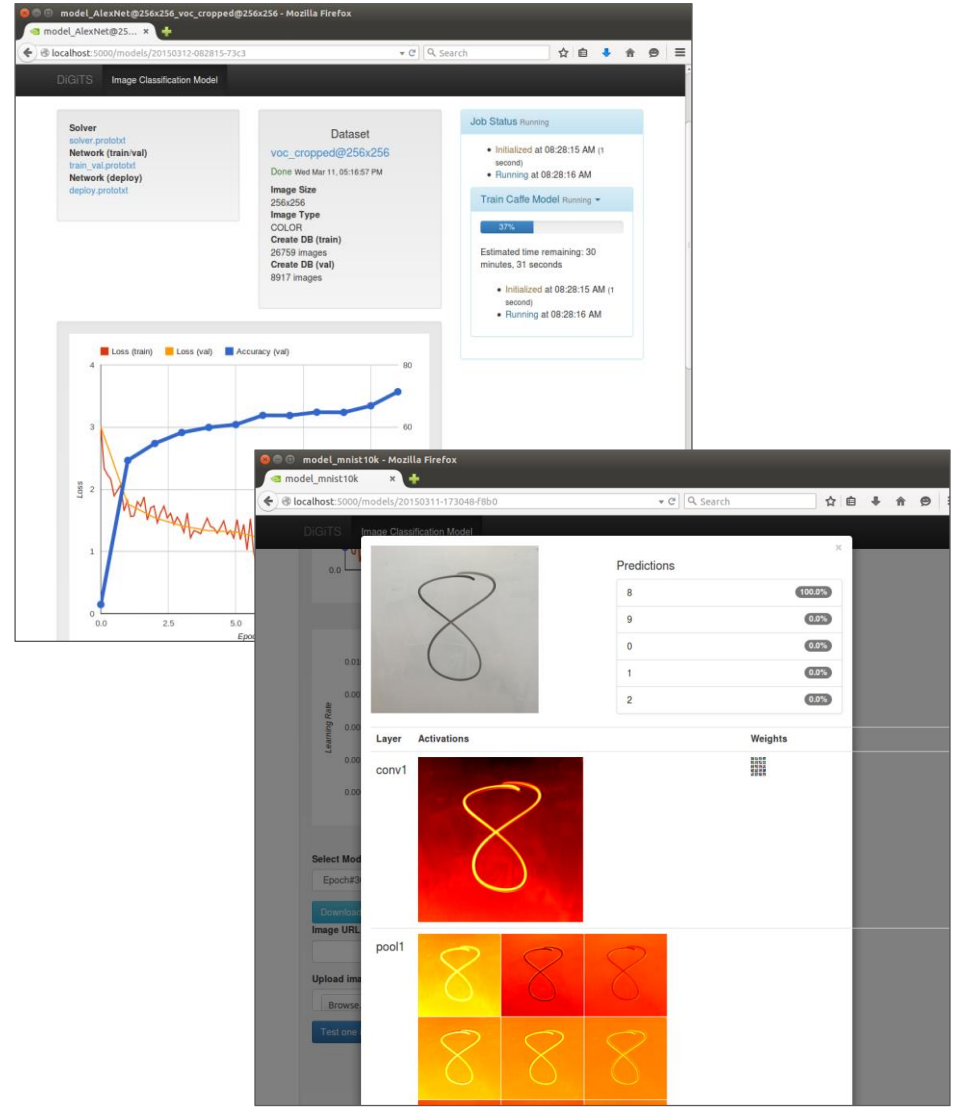
Interactive Deep Learning GPU Training System

Data Scientists and Researchers:

Quickly design the best deep neural network (DNN) for your data

Visually monitor DNN training quality in real-time

Manage training of many DNNs in parallel on multi-GPU systems



developer.nvidia.com/digits

OpenACC

Simple | Powerful | Portable

Fueling the Next Wave of Scientific Discoveries in HPC

```
main()
{
  <serial code>
  #pragma acc kernels
  //automatically run on GPU
  {
    <parallel code>
  }
}
```

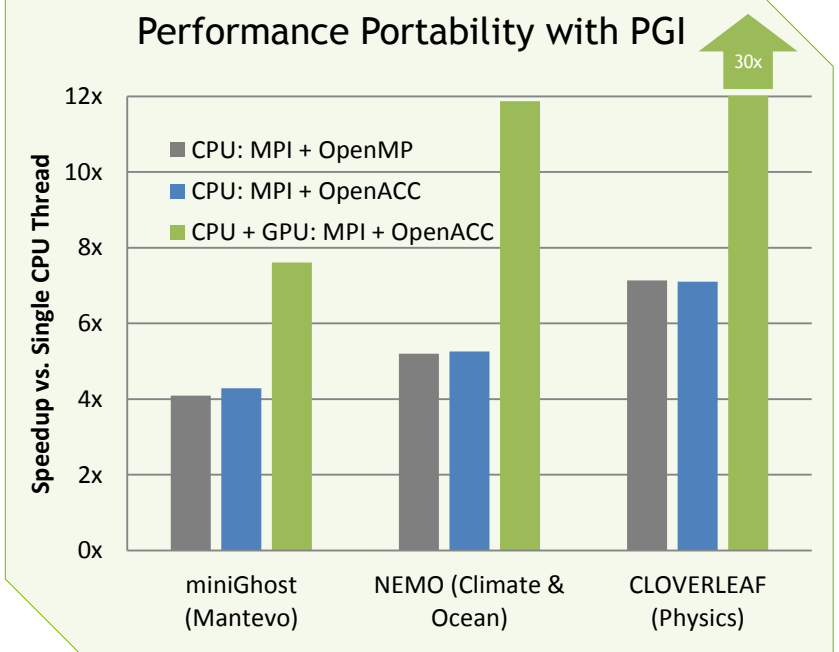
University of Illinois
PowerGrid- MRI Reconstruction

70x Speed-Up
2 Days of Effort

RIKEN Japan
NICAM- Climate Modeling

7-8x Speed-Up
5% of Code Modified

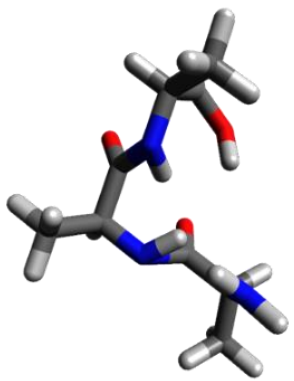
Performance Portability with PGI



http://www.cray.com/sites/default/files/resources/OpenACC_213462.12_OpenACC_Cosmo_CS_FNL.pdf
http://www.hpcwire.com/off-the-wire/first-round-of-2015-hackathons-gets-underway
http://on-demand.gputechconf.com/gtc/2015/presentation/S5297-Hisashi-Yashiro.pdf
http://www.openacc.org/content/experiences-porting-molecular-dynamics-code-gpus-cray-xk7

359.miniGhost: CPU: Intel Xeon E5-2698 v3, 2 sockets, 32-cores total, GPU: Tesla K80 (single GPU)
NEMO: Each socket CPU: Intel Xeon E5--2698 v3, 16 cores; GPU: NVIDIA K80 both GPUs
CLOVERLEAF: CPU: Dual socket Intel Xeon CPU E5-2690 v2, 20 cores total, GPU: Tesla K80 both GPUs

Large-scale Application for Calculating High-accuracy Molecular Energies



"*OpenACC makes GPU computing approachable for domain scientists. Initial OpenACC implementation required only minor effort, and more importantly, no modifications of our existing CPU implementation.*"

Janus Juul Eriksen, PhD Fellow
qLEAP Center for Theoretical Chemistry, Aarhus University

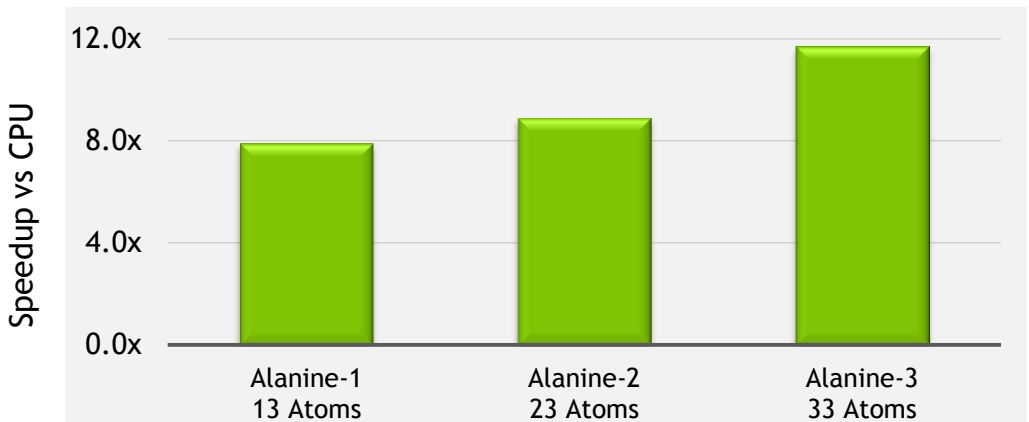
Minimal Effort

Lines of Code Modified	# of Weeks Required	# of Codes to Maintain
<100 Lines	1 Week	1 Source

Big Performance

LS-DALTON CCSD(T) Module

Benchmarked on Titan Supercomputer (AMD CPU vs Tesla K20X)



The NVIDIA OpenACC Toolkit

Free Toolkit Offers Simple & Powerful Path to Accelerated Computing

PGI Compiler
Free OpenACC compiler for academia

NVProf Profiler
Easily find where to add compiler directives

Code Samples
Learn from examples of real-world algorithms

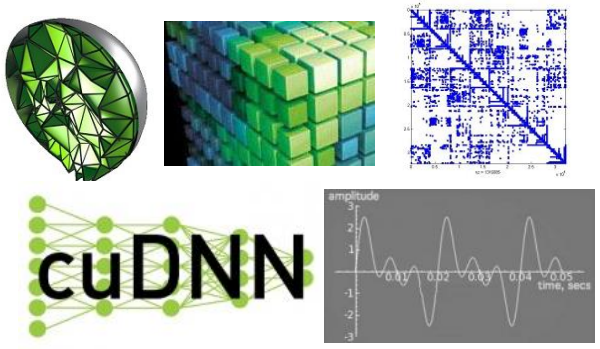
Documentation
Quick start guide, Best practices, Forums

Download at developer.nvidia.com/openacc

CUDA Toolkit

Flexibility – High Performance

LIBRARIES



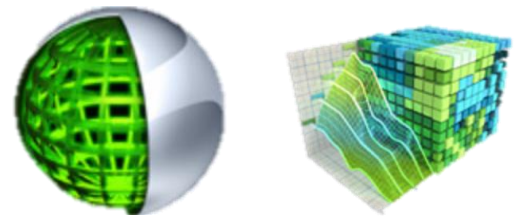
Drop-In Acceleration

PARALLEL C++

```
__global__
void xyzw_frequency(int
                    char *text, int n)
{
  const char letters[] { 'x','y','z','w' };
  count_if(count, text, n, [&](char c) {
    for (const auto x : letters)
      if (c == x) return true;
      return false;
  });
}
```

Parallel Algorithms and Custom Parallel C++

DEVELOPER TOOLS



Pinpoint Bugs and Performance Opportunities

Download at developer.nvidia.com/cuda-toolkit

GPU Accelerated Libraries

Drop-in Acceleration for Your Applications

Linear Algebra & Numerical

- cuBLAS
- cuSparse
- cuSOLVER
- AmgX
- NVIDIA Math Lib
- cuRAND
- SuiteSparse CHOLMOD

Parallel Algorithms

- Thrust
- ArrayFire

Machine Learning

- cuDNN

Signal Processing

FFT, Image & Video

- cuFFT
- NVENC Video Encode
- NVIDIA NPP
- FFmpeg

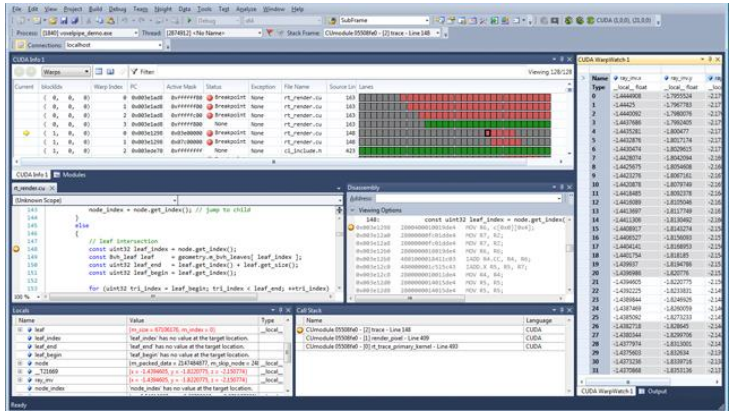
developer.nvidia.com/gpu-accelerated-libraries

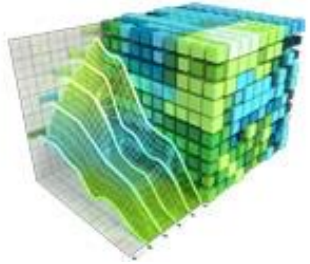
Developer Tools

Pinpoint Performance Bottlenecks and Hard-to-Find Bugs

NVIDIA®
Nsight™

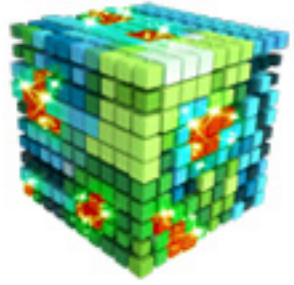
Develop, Debug, & Profile in Eclipse or Visual Studio





NVIDIA Visual Profiler
Analyze GPU Performance

CUDA-GDB
Standard Linux Debugger



CUDA-MEMCHECK
Find memory errors, leaks and race conditions

Your Next Steps

Many Learning Opportunities

Free Hands-On Self-Paced Labs in the NVIDIA Booth

Sign up for credit for on-line labs

Free In-Depth On-Line Courses

Deep Learning: developer.nvidia.com/deep-learning-courses

OpenACC: developer.nvidia.com/openacc-course

Learn from hundreds of experts at GTC 2016

All about Accelerated Computing: developer.nvidia.com

Backup

THRUST: Productive Parallel C++

High-level parallel algorithms

Resembles C++ STL

Included in CUDA Toolkit

Productive High-level API

CPU/GPU Performance portability

- CUDA, OpenMP, and TBB backends

Flexible, extensible

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

C++11 in CUDA

"C++11 Feels like a new language" – Bjarne Stroustrup

Count the number of occurrences of letters x, y, z and w in text

```
__global__
void xyzw_frequency(int *count, char *text, int n)
{
    const char letters[] { 'x','y','z','w' };

    count_if(count, text, n, [&](char c) {
        for (const auto x : letters)
            if (c == x) return true;
        return false;
    });
}
```

Initializer List

Lambda Function

Range-based For Loop

Automatic type deduction

Output:

```
Read 3288846 bytes from "warandpeace.txt"
counted 107310 instances of 'x', 'y', 'z', or 'w' in "warandpeace.txt"
```

Parallel STL: Algorithms + Policies

Higher-level abstractions for ease of use and performance portability

```
int main()
{
  size_t n = 1 << 16;
  std::vector<float> x(n, 1), y(n, 2), z(n);
  float a = 13;
  auto I = interval(0, n);

  std::for_each(std::par, std::begin(I), std::end(I),
  [&](int i)
  {
    z[i] = a * x[i] + y[i];
  });

  return 0;
}
```

On track for C++17 Standard

Includes algorithms such as for_each, reduce, transform, inclusive_scan …

Execution policies:
- std::seq
- std::par
- std::par_vec

Instruction-Level Profiling

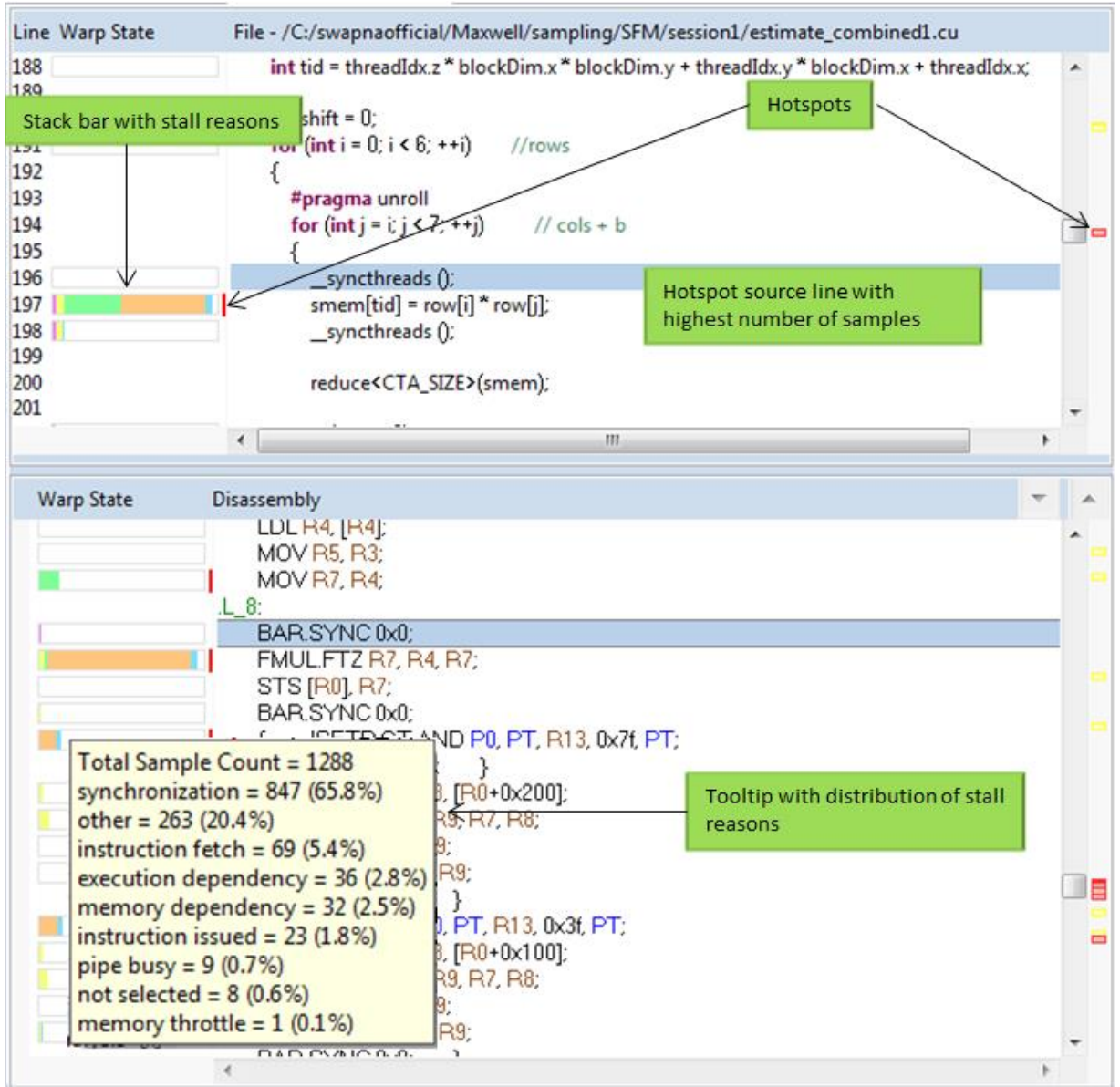
Pinpoint Performance Problems

New in CUDA 7.5 Visual Profiler

Highlights hot spots in code, helping you target bottlenecks

Quickly locate expensive code lines

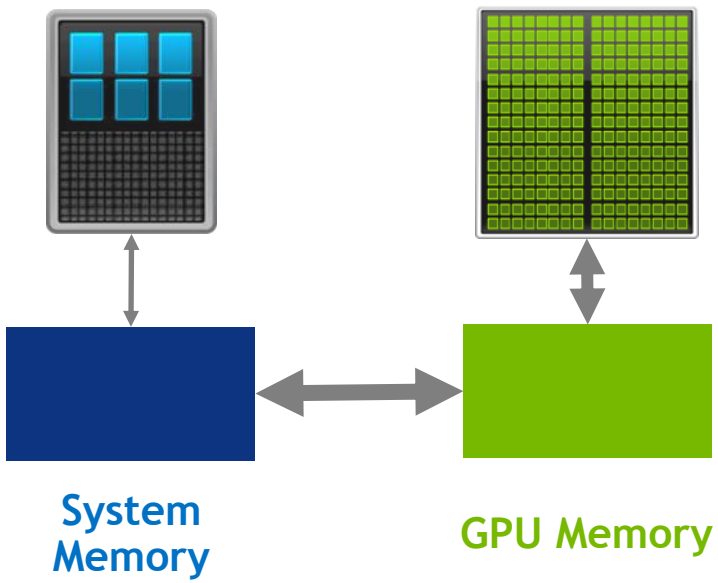
Detailed per-line stall analysis helps you understand underlying limiters



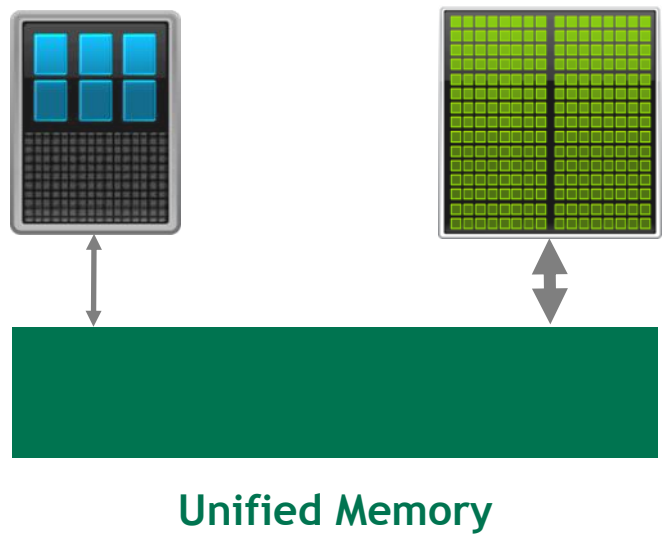
Unified Memory

Dramatically Lower Developer Effort

Past Developer View



Developer View With Unified Memory



Simplified Memory Management Code

CPU Code

```
void sortfile(FILE *fp, int N) {
  char *data;
  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

CUDA 6 Code with Unified Memory

```
void sortfile(FILE *fp, int N) {
  char *data;
  cudaMallocManaged(&data, N);

  fread(data, 1, N, fp);

  qsort<<<...>>>(data,N,1,compare);
  cudaDeviceSynchronize();

  use_data(data);

  cudaFree(data);
}
```

Great Performance with Unified Memory

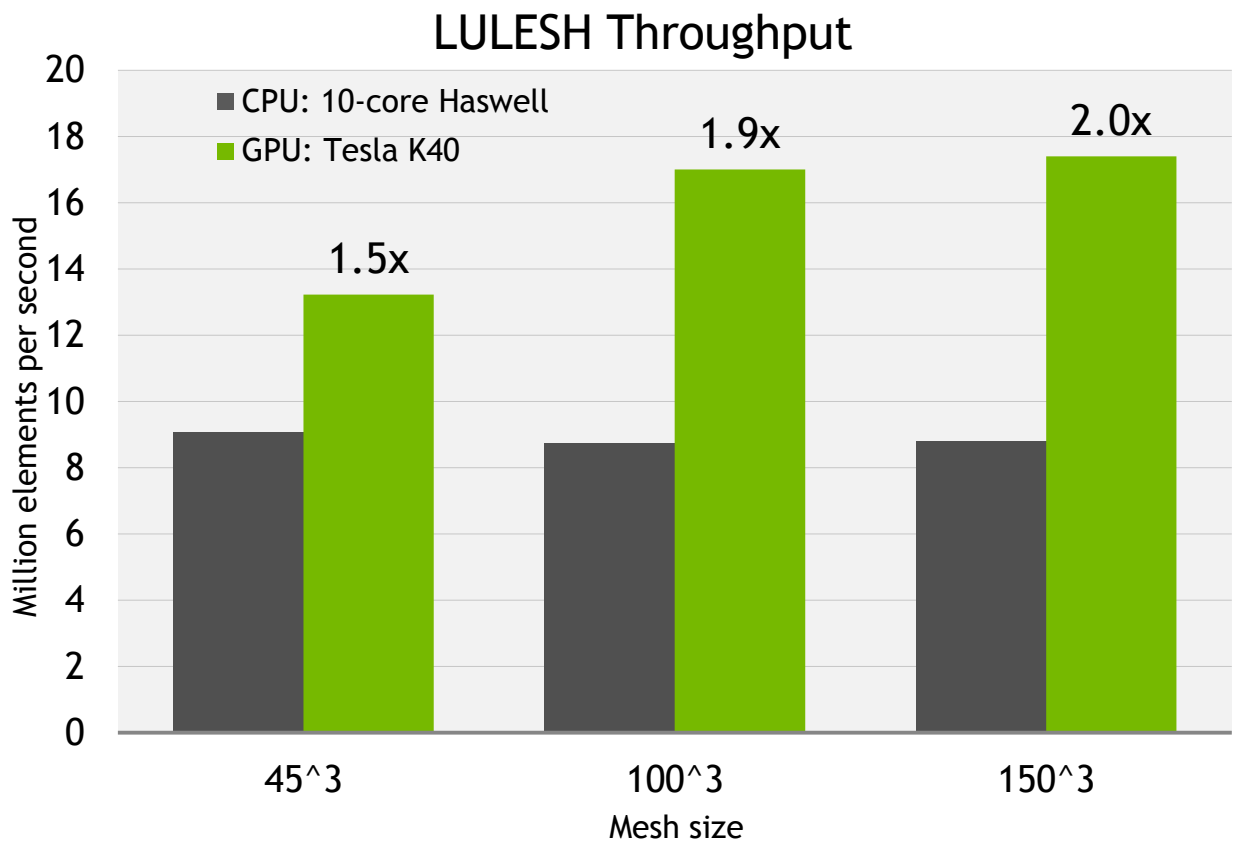
RAJA: Portable Parallel C++ Framework

RAJA uses Unified Memory for heterogeneous array allocations

Parallel `forall` loops run on CPU or GPU

"Excellent performance considering this is a "generic" version of LULESH with no architecture-specific tuning."

- Jeff Keasler, LLNL



***GPU:** NVIDIA Tesla K40, **CPU:** Intel Haswell E5-2650 v3 @ 2.30GHz, single socket 10-core*