ACM Learning Center
http://learning.acm.org

- 1,350+ trusted technical books and videos by leading publishers including O’Reilly, Morgan Kaufmann, others

- Online courses with assessments and certification-track mentoring, member discounts on tuition at partner institutions

- Learning Webinars on big topics (Cloud/Mobile Development, Cybersecurity, Big Data, Recommender Systems, SaaS, Agile, Machine Learning, Natural Language Processing, Parallel Programming, IPv6, WebGL, Big Data, ICSM)

- ACM Tech Packs on top current computing topics: Annotated Bibliographies compiled by subject experts

- Popular video tutorials/keynotes from ACM Digital Library, A.M. Turing Centenary talks/panels

- Podcasts with industry leaders/award winners
Talk Back

• Use the Facebook widget in the bottom panel to share this presentation with friends and colleagues

• Use Twitter widget to Tweet your favorite quotes from today’s presentation with hashtag #ACMWebinarGPU

• Submit questions and comments via Twitter to @acmeducation - we’re reading them!
Founded in 1993

Jen-Hsun Huang is co-founder and CEO

Listed with NASDAQ under the symbol NVDA in 1999

Invented the GPU in 1999; shipped more than 1 billion to date

FY13: $4.3 billion in revenue

8,500 employees worldwide

6,400 patent assets

Headquartered in Santa Clara, Calif.
My Three Points

1. What is Accelerated Computing?

2. The Why and How of GPUs

3. Resources
Moore’s Law
What makes up the brick wall?

Power Wall

Memory Wall

+ __________ ILP Wall

= Brick Wall
Heterogeneous Computing

CPU + Accelerator
The basic idea

Application Code

Compute-Intensive Functions

Rest of Sequential CPU Code

Accelerator

CPU
Three Major Accelerators
From Phones to Cars
DIAGNOSTIC IMAGING PERFORMANCE
Real-Time Image Reconstruction with GPUs.
Only 2% of surgeons will operate on a beating heart

Patient stands to lose 1 point of IQ every 10 min with heart stopped

GPU enables real-time motion compensation to virtually stop beating heart for surgeons

Courtesy Laboratoire d’Informatique de Robotique et de Microelectronique de Montpellier
How do I use GPUs?
What is CUDA?

- Programming language?
- Compiler?
- Classic car?
- Beer?
- Wine?
- Coffee?
## CUDA Parallel Computing Platform

www.nvidia.com/getcuda

### Programming Approaches
- **Libraries**: “Drop-in” Acceleration
- **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
Growth of GPU Computing

- 100M CUDA-Capable GPUs
- 150K CUDA Downloads
- 77 Supercomputing Teraflops
- 60 University Courses
- 4,000 Academic Papers

2008
Growth of GPU Computing

- **2008**
  - 100M CUDA-Capable GPUs
  - 150K CUDA Downloads
  - 60 University Courses
  - 4,000 Academic Papers

- **2014**
  - 430M CUDA-Capable GPUs
  - 2.2M CUDA Downloads
  - 77 Supercomputing Teraflops
  - 738 University Courses
  - 50,000 Academic Papers
3 Ways to Accelerate Applications

Applications

- Libraries
  - “Drop-in” Acceleration

- Directives
  - Easily Accelerate Applications

- Programming Languages
  - Maximum Flexibility
Top HPC Applications

Molecular Dynamics
- AMBER
- CHARMM
- DESMOND

Quantum Chemistry
- Abinit
- Gaussian

Material Science
- CP2K
- QMCPACK
- Quantum Espresso
- VASP

Weather & Climate
- COSMO
- GEOS-5
- HOMME
- CAM-SE
- NEMO
- NIM
- WRF

Lattice QCD
- Chroma
- MILC

Plasma Physics
- GTC
- GTS

Structural Mechanics
- ANSYS Mechanical
- LS-DYNA Implicit
- MSC Nastran
- OptiStruct
- Abaqus/Standard

Fluid Dynamics
- ANSYS Fluent
- Culises (OpenFOAM)

Accelerated, In Development

Solid Growth of GPU Accelerated Apps

<table>
<thead>
<tr>
<th>Year</th>
<th># of GPU-Accelerated Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>113</td>
</tr>
<tr>
<td>2012</td>
<td>182</td>
</tr>
<tr>
<td>2013</td>
<td>272</td>
</tr>
</tbody>
</table>
## Popular GPU-Accelerated Applications

**Research: Higher Education and Supercomputing**

### Computational Chemistry and Biology

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
<th>Version</th>
<th>Dependencies</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biochemistry</td>
<td>Sequence mapping software</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
<tr>
<td>CMAP</td>
<td>Quantum mechanics simulations of proteins, DNA, and organics</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
<tr>
<td>AMBER</td>
<td>Solute-solvent interactions</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
<tr>
<td>CHARMM</td>
<td>Molecular dynamics of biological molecules</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
<tr>
<td>BRIDGES</td>
<td>Molecular dynamics simulation of membranes</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
</tbody>
</table>

### Molecular Dynamics

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
<th>Version</th>
<th>Dependencies</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEL</td>
<td>Electrostatic potentials and molecular dynamics of biological molecules</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
<tr>
<td>LAMMPS</td>
<td>Molecular dynamics simulation of proteins, DNA, and organics</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
<tr>
<td>NAMD</td>
<td>Structural dynamics of biological molecules</td>
<td>Yes</td>
<td>Other software</td>
<td>Available</td>
</tr>
</tbody>
</table>

### Popular Applications

- **270+ GPU-Accelerated Applications**
- www.nvidia.com/appscatalog
Performance on Leading Scientific Applications

- Structural Mechanics
  - ANSYS

- Physics
  - CHROMA

- Molecular Dynamics
  - AMBER

- Material Science
  - QMCPACK

- Earth Science
  - SPECFEM3D

Graph showing performance metrics for different applications.
3 Ways to Accelerate Applications

- Libraries: “Drop-in” Acceleration
- Directives: Easily Accelerate Applications
- Programming Languages: Maximum Flexibility
Programming with CUDA
GPU Accelerated Libraries
“Drop-in” Acceleration for your Applications

NVIDIA cuBLAS
NVIDIA cuRAND
NVIDIA cuSPARSE
NVIDIA NPP

GPU VSIPL
Vector Signal Image Processing

CULA tools
GPU Accelerated Linear Algebra

MAGMA
Matrix Algebra on GPU and Multicore

Rogue Wave Software
IMSL Library

{CSP}
ArrayFire Matrix Computations

CUSP
Sparse Linear Algebra

Thrust
C++ STL Features for CUDA
Thrust C++ Template Library

Serial C++ Code
with STL and Boost

```
int N = 1<<20;
std::vector<float> x(N), y(N);
...

// Perform SAXPY on 1M elements
std::transform(x.begin(), x.end(),
y.begin(), y.end(),
2.0f * _1 + _2);
```

Parallel C++ Code

```
int N = 1<<20;
thrust::host_vector<float> x(N), y(N);
...

thrust::device_vector<float> d_x = x;
thrust::device_vector<float> d_y = y;

// Perform SAXPY on 1M elements
thrust::transform(d_x.begin(), d_x.end(),
d_y.begin(), d_y.end(),
2.0f * _1 + _2);
```

www.boost.org/libs/lambda

thrust.github.com
Explore the CUDA (Libraries) Ecosystem

CUDA Tools and Ecosystem described in detail on NVIDIA Developer Zone:
developer.nvidia.com/cuda-tools-ecosystem
3 Ways to Accelerate Applications

- Libraries: “Drop-in” Acceleration
- Directives: Easily Accelerate Applications
- Programming Languages: Maximum Flexibility
OpenACC Directives

Program myscience
... serial code ...
!$acc kernels
do k = 1,n1
do i = 1,n2
... parallel code ...
enddo
dendo
!$acc end kernels
... End Program myscience

CPU

GPU

Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs & multicore CPUs
SAXPY - OpenACC

**SAXPY in C**

```c
void saxpy(int n,
    float a,
    float *x,
    float *restrict y)
{
    #pragma acc parallel loop
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}

... // Perform SAXPY on N elements
saxpy(N, 3.0, x, y);
...
```

**SAXPY in Fortran**

```fortran
subroutine saxpy(n, a, x, y)
    real :: x(*), y(*), a
    integer :: n, i
    !$acc parallel loop
    do i=1,n
        y(i) = a*x(i)+y(i)
    enddo
    !$acc end parallel
end subroutine saxpy

... ! Perform SAXPY on N elements
saxpy(N, 3.0, x, y)
...
```
void saxpy(int n,
    float a,
    float *x,
    float *restrict y)
{
    #pragma omp parallel for
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}

... // Perform SAXPY on N elements
saxpy(N, 3.0, x, y);
...

double saxpy(double n, double a, double *x, double *restrict y)
{
    #pragma omp parallel for
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}

... // Perform SAXPY on N elements
saxpy(N, 3.0, x, y);
...
OpenACC Implementations

OpenACC 2.0 launched December 2013

OpenACC 2.0 launched December 2013

OpenACC 2.0 Rolling out from January 2014

OpenACC 2.0 Targeted for late 2014/ early 2015

Known academic efforts:

- accULL – U. of La Laguna/EPCC
- Omni – U. of Tsukuba
- OpenARC – ORNL
- OpenUH – U. of Houston
3 Ways to Accelerate Applications

- Libraries: “Drop-in” Acceleration
- Directives: Easily Accelerate Applications
- Programming Languages: Maximum Flexibility
## GPU Programming Languages

<table>
<thead>
<tr>
<th>Category</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical analytics</td>
<td>MATLAB, Mathematica, LabVIEW</td>
</tr>
<tr>
<td>Fortran</td>
<td>OpenACC, CUDA Fortran</td>
</tr>
<tr>
<td>C</td>
<td>OpenACC, CUDA C</td>
</tr>
<tr>
<td>C++</td>
<td>Thrust, CUDA C++</td>
</tr>
<tr>
<td>Python</td>
<td>CUDA Python, PyCUDA</td>
</tr>
<tr>
<td>F#</td>
<td>Alea.cubase</td>
</tr>
</tbody>
</table>
CUDA 6 - Unified Memory
Dramatically Lower Developer Effort

Developer View Today

System Memory

GPU Memory

CUDA 6 Release Candidate available now
Programming a CUDA-enabled Language

CUDA C/C++

- Based on industry-standard C/C++
- Small set of extensions to enable heterogeneous programming
- Straightforward APIs to manage devices, memory etc.
Prerequisites

- You (probably) need experience with C or C++
- You don’t need GPU experience
- You don’t need parallel programming experience
- You don’t need graphics experience
Heterogeneous Computing

- **Terminology:**
  - *Host*  The CPU and its memory (host memory)
  - *Device* The GPU and its memory (device memory)
void saxpy(int n, float a, float *x, float *y) {
    for (int i = 0; i < n; ++i)  
        y[i] = a*x[i] + y[i]; 
}

int N = 1<<20;

// Perform SAXPY on 1M elements
saxpy(N, 2.0f, x, y);
Parallelism on a GPU - CUDA Blocks

A function which runs on a GPU is called a “kernel”

Each parallel invocation of a function running on the GPU is called a “block”

A block can identify itself by reading `blockIdx.x`

- `blockIdx.x = 0`
- `blockIdx.x = 1`
- `blockIdx.x = 2`
- `blockIdx.x = W-1`

...
Parallelism on a GPU - CUDA Threads

Each block is then broken up into "threads". A thread can identify itself by reading `threadIdx.x`. The total number of threads per block can be read with `blockDim.x`.

In the above example, `blockDim.x = M`.

```
threadIdx.x = 0
threadIdx.x = 1
threadIdx.x = 2
threadIdx.x = M - 1
```

Block = THREAD

```
threadIdx.x = 2
threadIdx.x = M - 1
```
Why threads and blocks?

Threads within a block can communicate very quickly (share memory) and synchronize (wait for all threads to catch up).

Why break up into blocks?

- Limiting cooperation to a subset of threads enables building a high-performance hardware implementation.
- By requiring all blocks to be independent, programs can scale to larger or smaller GPUs without code changes or even rec kompilation.
void saxpy_cpu(int n, float a, float *x, float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}
SAXPY kernel

```c
__global__ void saxpy_gpu(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];
}
```

blockIdx.x: Our Block ID
blockDim.x: Number of threads per block
threadIdx.x: Our thread ID

i is now an index into our input and output arrays
SAXPY kernel - with data

__global__ void saxpy_gpu(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];
}

Let's work with 30 data elements

Broken into 3 blocks, with 10 threads per block

For blockIdx.x = 0
    
    i = 0 * 10 + threadIdx.x = \{0,1,2,3,4,5,6,7,8,9\}

For blockIdx.x = 1
    
    i = 1 * 10 + threadIdx.x = \{10,11,12,13,14,15,16,17,18,19\}

For blockIdx.x = 2
    
    i = 2 * 10 + threadIdx.x = \{20,21,22,23,24,25,26,27,28,29\}

10 threads (hamsters) each with a different i
#define N (2048 * 512)
int main(void) {
  float *x, *y;     // host copies
  int size = N * sizeof(float);

  // Alloc space for x & y and
  // setup input values
  x = (float *)malloc(size);
  random_floats(x, N);
  y = (float *)malloc(size);
  random_floats(y, N);

  // Launch saxpy on CPU
  saxpy_cpu(N, 2.0f, x, y);

  // Cleanup
  free(x); free(y);
  return 0;
}

#define N (2048 * 512)
int main(void) {
  float *x, *y;     // host copies
  int size = N * sizeof(float);

  // Alloc space for x & y and
  // setup input values
  x = (float *)malloc(size);
  random_floats(x, N);
  y = (float *)malloc(size);
  random_floats(y, N);

  // Launch saxpy on GPU
  saxpy_cpu(N, 2.0f, x, y);

  // Cleanup
  free(x); free(y);
  return 0;
}
**NVIDIA® Nsight™ Eclipse Edition for Linux ’and MacOS**

**CUDA-Aware Editor**
- Automated CPU to GPU code refactoring
- Semantic highlighting of CUDA code
- Integrated code samples & docs

**Nsight Debugger**
- Simultaneously debug 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

[developer.nvidia.com/nsight](http://developer.nvidia.com/nsight)
NVIDIA® Nsight™, Visual Studio Ed.

CUDA Debugger
- Debug CUDA kernels directly on GPU hardware
- Examine thousands of threads executing in parallel
- Use on-target conditional breakpoints to locate errors

CUDA Memory Checker
- Enables precise error detection

System Trace
- Review CUDA activities across CPU and GPU
- Perform deep kernel analysis to detect factors limiting maximum performance

CUDA Profiler
- Advanced experiments to measure memory utilization, instruction throughput and stalls
NVIDIA Visual Profiler
Beyond HPC
Big Data

Analyzing Twitter

Searching Audio

Visual Shopping

Real-time Video Delivery
With Fricken Laserbeams!

- Created by Intellectual Ventures to help fight malaria in third world countries
- Image detection and targeting is done with NVIDIA GPUs
Resources: developer.nvidia.com/cudazone

Parallel Forall: devblogs.nvidia.com/parallellforall
  - CUDACasts at bit.ly/cudacasts
    - Short how-to screencasts

Self-paced labs: nvidia.qwiklab.com
  - 90-minute labs, simply need a supported web browser

Documentation: docs.nvidia.com

Technical Questions:
  - NVIDIA Developer forums devtalk.nvidia.com
  - Search or ask on stackoverflow.com/tags/cuda
CUDA Registered Developer Program

- Access to exclusive developer downloads
  - Double-Double Precision Library and Source
  - SIMD within a Word
  - Optimized LINPACK
  - And more...

- Exclusive access to pre-release CUDA Installers
  - Like CUDA 6!

- Submit bugs and features requests to NVIDIA Engineering

- Exclusive activities and special offers

- Membership is free and easy!

www.nvidia.com/paralleldeveloper
Test Drive the World’s Fastest GPU
Accelerate Your Code on the New Tesla K40 GPU

Accelerate your codes on latest GPUs today

Sign up for FREE GPU Test Drive on remotely hosted clusters

www.nvidia.com/GPUTestDrive
WHERE ART MEETS SCIENCE
MEETS ENGINEERING
MEETS BUSINESS

4 Days
500+ Sessions
150+ Research Posters
1:1 With NVIDIA Experts
50+ Countries

www.nvidia.com/gtc

20% Discount using coupon code GM20ACM
Getting Started

2. Subscribe to Parallel Forall blog
3. Sign up as a Registered Developer
4. Install the CUDA Toolkit
5. Attend GTC 2014 or watch GTC On-Demand
ACM: The Learning Continues...

Questions about this webcast? learning@acm.org

ACM Learning Webinars (on-demand archive): http://learning.acm.org/webinar

ACM Learning Center: http://learning.acm.org

ACM SIGHPC: http://www.sighpc.org/

ACM Queue: http://queue.acm.org
nvidia.qwiklab.com

- 90-minute, online, self-paced GPU Programming labs
- Only requires a browser and network which supports Web Sockets
  - You can verify by going to websocketstest.com and look for:
    - Internet Explorer 9 and earlier do NOT support Web Sockets
Udacity Parallel Programming Course

Learn. Think. Do.
Invent your future through free interactive college classes.

30,000+ Students Already Registered
IT’S FREE!

Udacity.com