High Performance Computing with CUDA

Supercomputing 2009 Tutorial

David Luebke NVIDIA Research

Welcome!

- GPUs have become a major force in HPC
  - National & commercial supercomputer installations
  - Changing the landscape with “personal supercomputing”
  - Emerging ecosystem of tools, vendors, languages, codes
- GPU codenamed “Fermi” will accelerate this trend
  - ECC, 8x double precision performance
  - Powerful development, debugging, profiling tools

Tutorial topics

- CUDA programming model
- Tools, languages, and libraries for GPU computing
- Advanced CUDA: optimization, irregular parallelism
- Case studies:
  - CFD
  - Seismic processing
  - QCD
  - Molecular dynamics

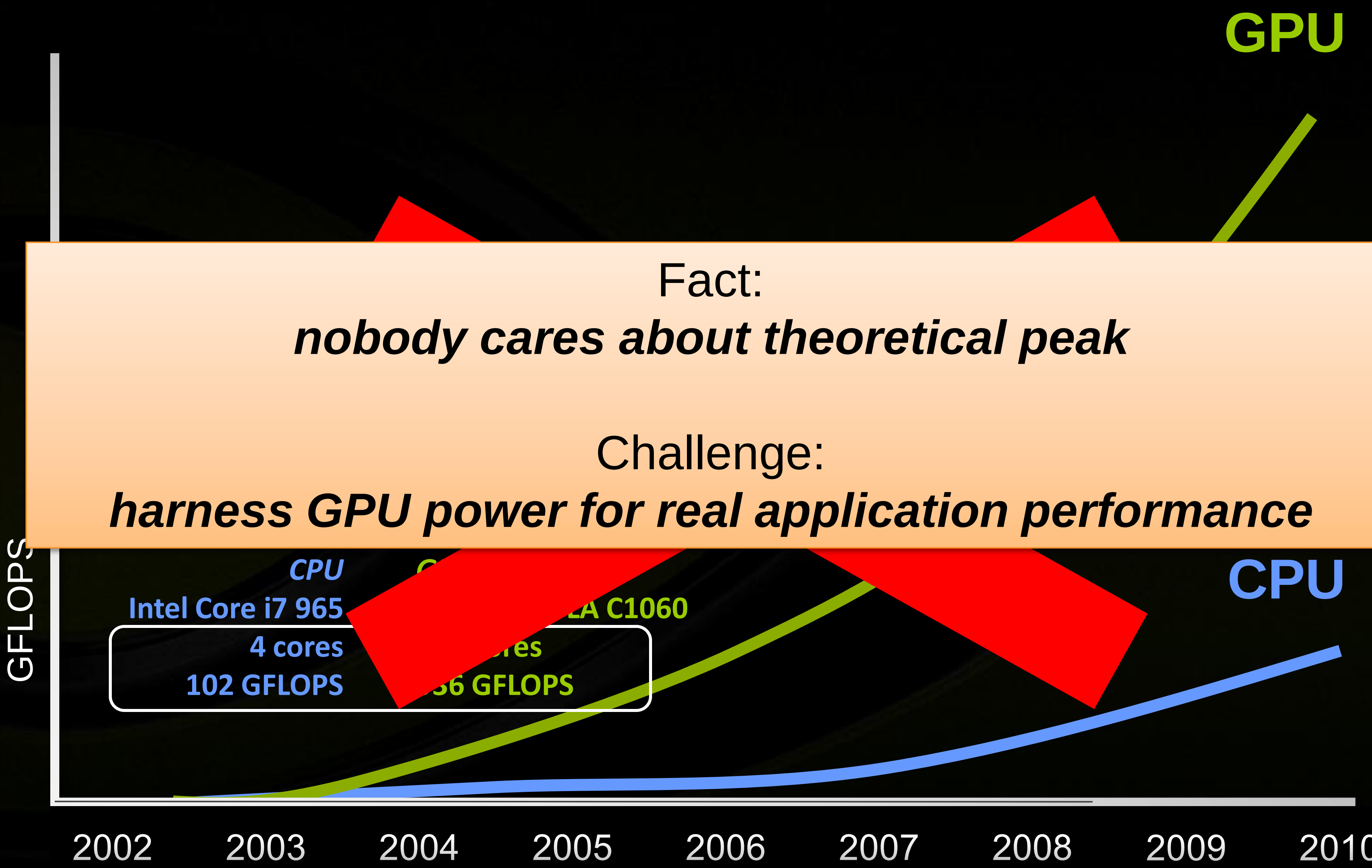
Motivation

Fact:

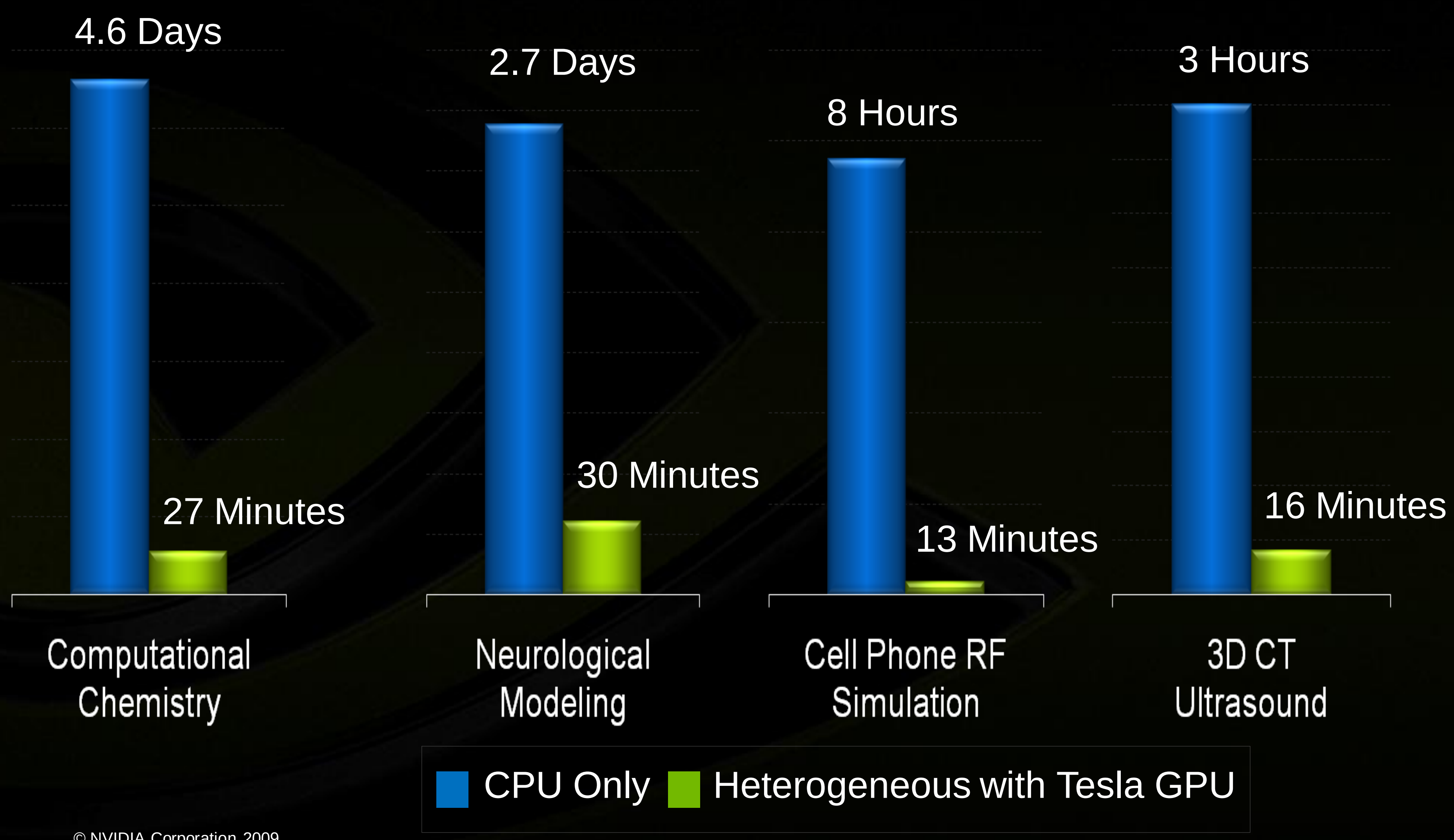
nobody cares about theoretical peak

Challenge:

harness GPU power for real application performance



Motivation: Accelerating Insight



CUDA is everywhere

Over 270 universities teach CUDA
Over 2500 research papers

CUDA powered supercomputers

180 Million CUDA GPUs

100,000 active developers

639 CUDA applications and counting

NVIDIA GPUs at Supercomputing 09

12% of papers use NVIDIA GPUs

GPU-based Paper by Hamada up for Gordon Bell

Jack Dongarra, ORNL, Sandia, Los Alamos, Matsuoka speaking at NVIDIA Booth

20+ System Providers are demoing Tesla GPUs

HP, Dell, Cray, Bull, Appro, NEC, SGI, Sun, SuperMicro, Penguin, Colfax, Silicon Mechanics, Scalable, Verari, Tycrid, Mellanox, Creative Consultants, Microway, ACE, TeamHPC

11+ Software Providers building on CUDA GPUs

Microsoft, The Mathworks, Allinea, TotalView, Accelereyes, EM Photonics, Tech-X, CAPS, Platform Computing, NAG, PGI, Wolfram

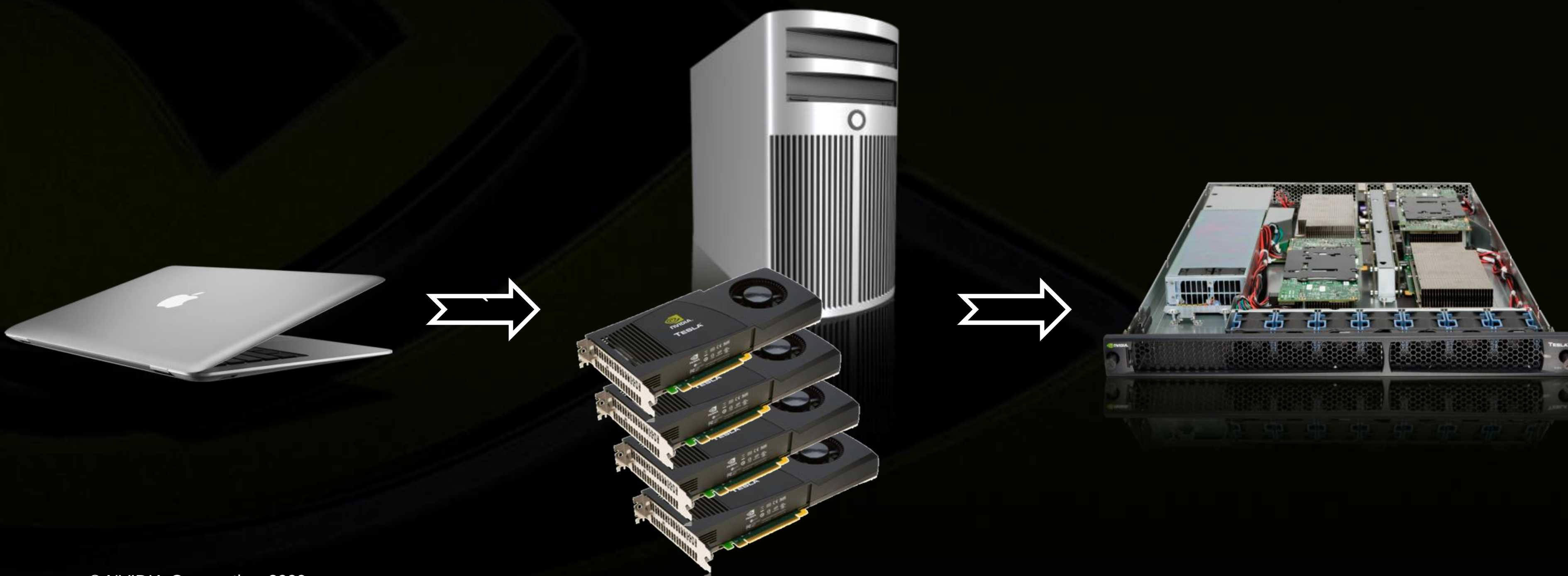
LANL, ORNL, SLAC, TACC, GaTech, HPC Advisory Council, Khronos Group showing GPU Computing Demos

GPUs in high-performance computing

	GPUs
Commercial cluster	>3000
Government	>2000
Chinese Academy of Sciences - Industrial Process Inst.	828
Tokyo Institute of Technology Supercomputing Center	680
NCSA – National Center for Supercomputing Applications	384
Seismic processing	256
Pacific Northwest National Labs – Biomedical research	256
CSIRO – Australian National Supercomputing Center	252
Riken – Japanese Astrophysical Research	220
Seismic processing	200
Chinese Academy of Sciences – Inst. of Modern Physics	200

Products

CUDA is in products from laptops to supercomputers



Emerging HPC Products

New class of hybrid GPU-CPU servers

SuperMicro 1U
GPU Server

Bull Bullx
Blade Enclosure

Tutorial goals

- A detailed introduction to high performance computing with CUDA
- We emphasize:
  - Understanding the architecture & programming model
  - Core computational building blocks
  - Libraries and tools
  - Optimization strategy & tactics
- Case studies to bring it all together

Tutorial prerequisites

- Tutorial intended to be accessible to any savvy computer or computational scientist
- Helpful but not required: familiarity with data-parallel algorithms and programming
- Target audience: HPC practitioners using or considering CUDA

Speakers:

- In order of appearance:

David Luebke	NVIDIA
Ian Buck	NVIDIA
Jonathan Cohen	NVIDIA
John Owens	University of California Davis
Paulius Micikevicius	NVIDIA
Scott Morton	Hess
John Stone	University of Illinois Urbana-Champaign
Mike Clark	Harvard

Schedule

8:30 Introduction **Luebke**

Welcome, overview, CUDA basics

9:00 CUDA programming environments **Buck**

Toolchain, languages, wrappers

10:00 Break

10:30 CUDA libraries & tools **Cohen**

MAGMA & CULA, Thrust, CuFFT, CuBLAS…

CUDA-gdb, Visual Profiler, codename “Nexus”…

Schedule

11:15	Optimizing GPU performance	Micikevicius
12:00	Lunch	
1:30	Optimizing CPU-GPU performance	Micikevicius
1:45	Irregular algorithms & data structures *Sparse linear algebra, tree traversal, hash tables*	Owens

Schedule: case studies

2:30	Molecular modeling	Stone
3:00	Break	
3:30	Seismic imaging	Morton
4:00	Computational fluid dynamics	Cohen
5:00	Quantum Chromodynamics	Clark
5:00	Wrap!	

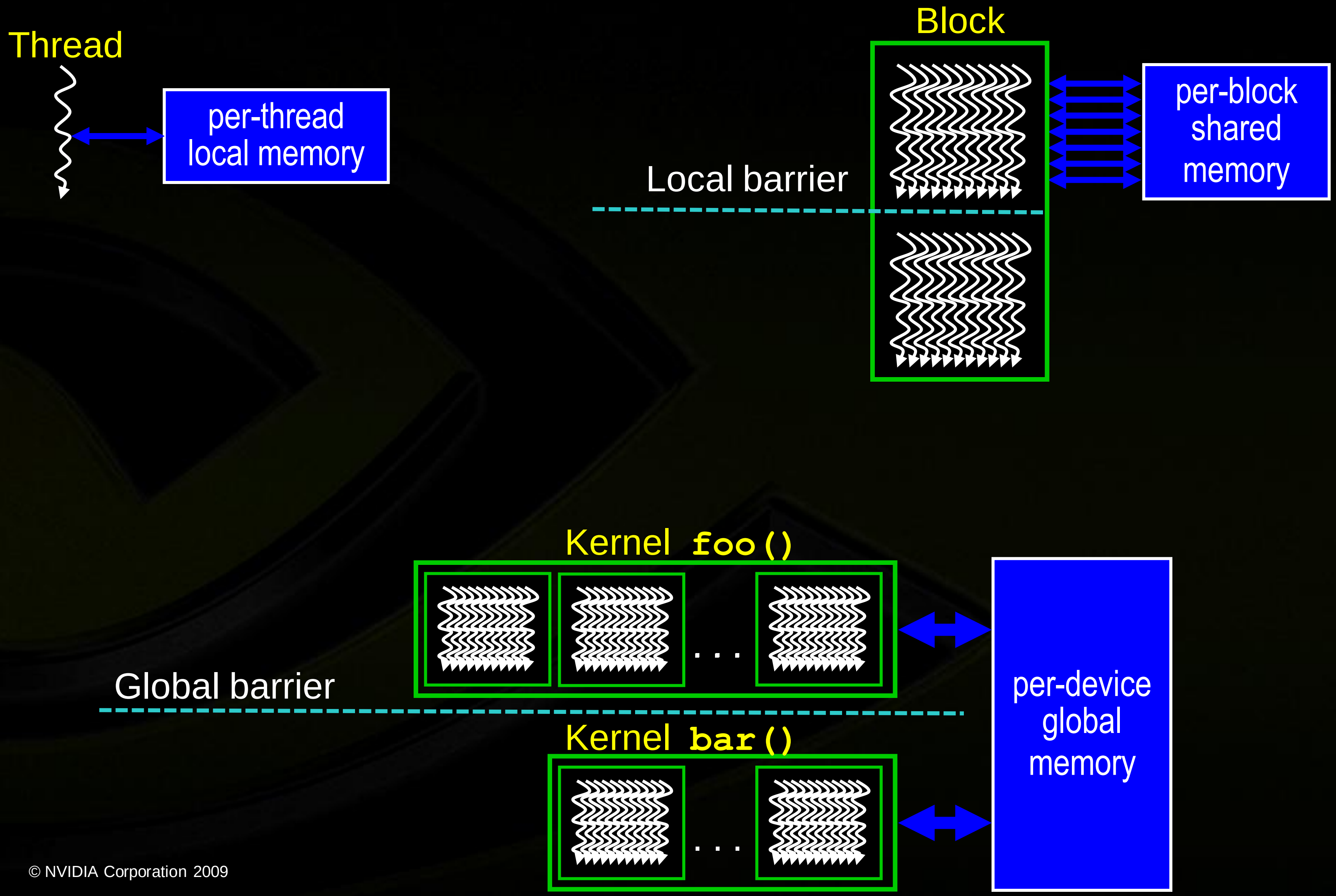
CUDA Basics

Programming model & simple examples

David Luebke NVIDIA Research

NVIDIA

CUDA In One Slide



CUDA C Example

```
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];
}

// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

Serial 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];
}

// Invoke parallel SAXPY kernel with 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

Parallel C Code

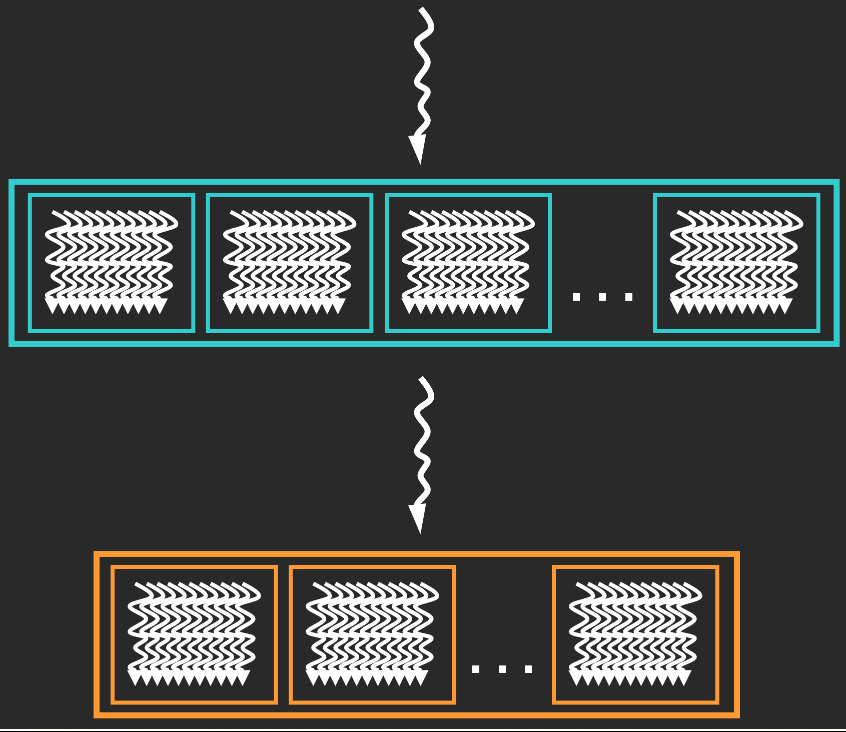
Heterogeneous Programming

- Use the right processor for the right job

Serial Code

Parallel Kernel

```
foo<<< nBlk, nTid >>>(args);
```



Serial Code

Parallel Kernel

```
bar<<< nBlk, nTid >>>(args);
```

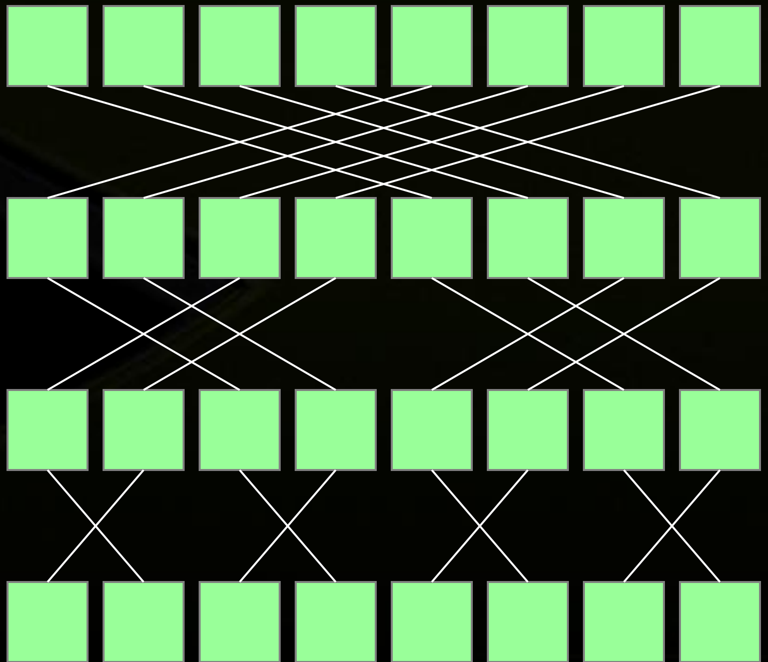
Example: Parallel Reduction

- Summing up a sequence with 1 thread:

```
int sum = 0;
for(int i=0; i<N; ++i)  sum += x[i];
```

- Parallel reduction builds a summation tree
  - each thread holds 1 element
  - stepwise partial sums
  - N threads need log N steps

  - one possible approach:
    Butterfly pattern

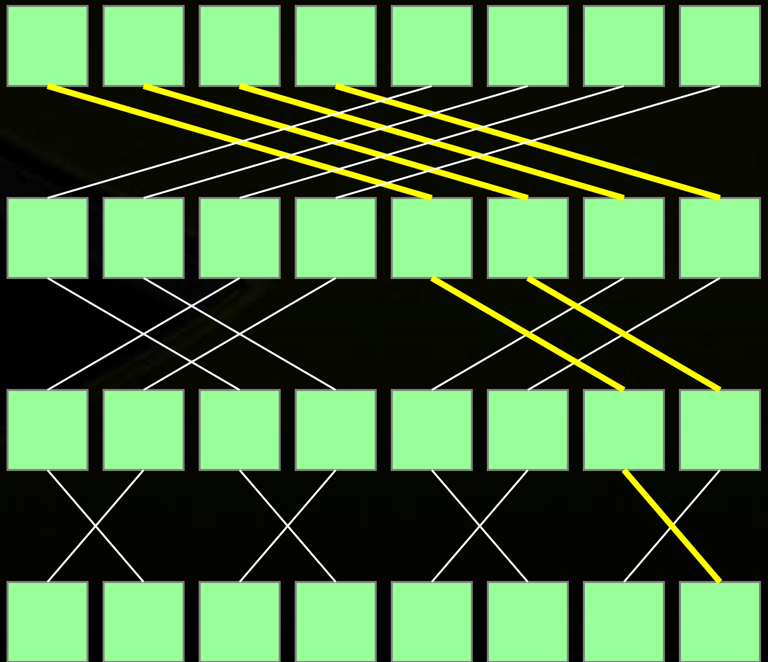


Example: Parallel Reduction

- Summing up a sequence with 1 thread:

```
int sum = 0;
for(int i=0; i<N; ++i)  sum += x[i];
```

- Parallel reduction builds a summation tree
  - each thread holds 1 element
  - stepwise partial sums
  - N threads need log N steps
  - one possible approach: Butterfly pattern

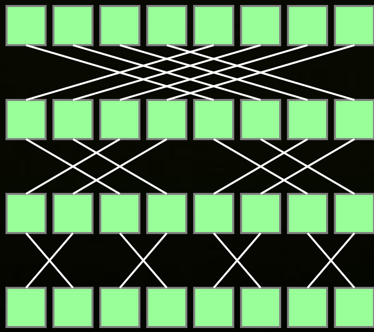


Parallel Reduction for 1 Block

```
// INPUT: Thread i holds value x_i
int i = threadIdx.x;
__shared__ int sum[blocksize];
```

```
// One thread per element
sum[i] = x_i; __syncthreads();
```

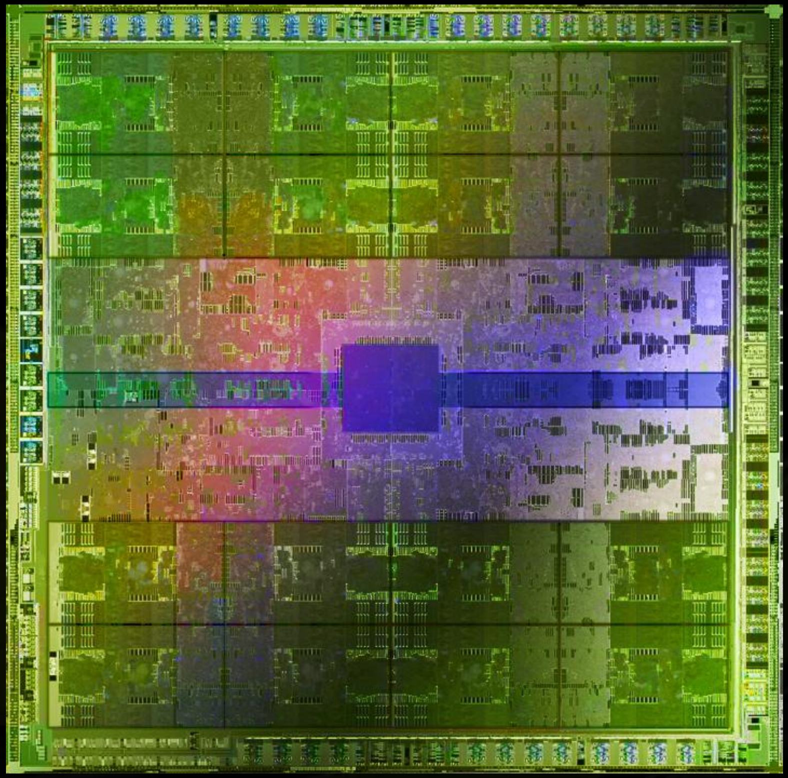
```
for(int bit=blocksize/2; bit>0; bit/=2)
{
    int t=sum[i]+sum[i^bit]; __syncthreads();
    sum[i]=t;                __syncthreads();
}
```



```
// OUTPUT: Every thread now holds sum in sum[i]
```

Codename "Fermi"

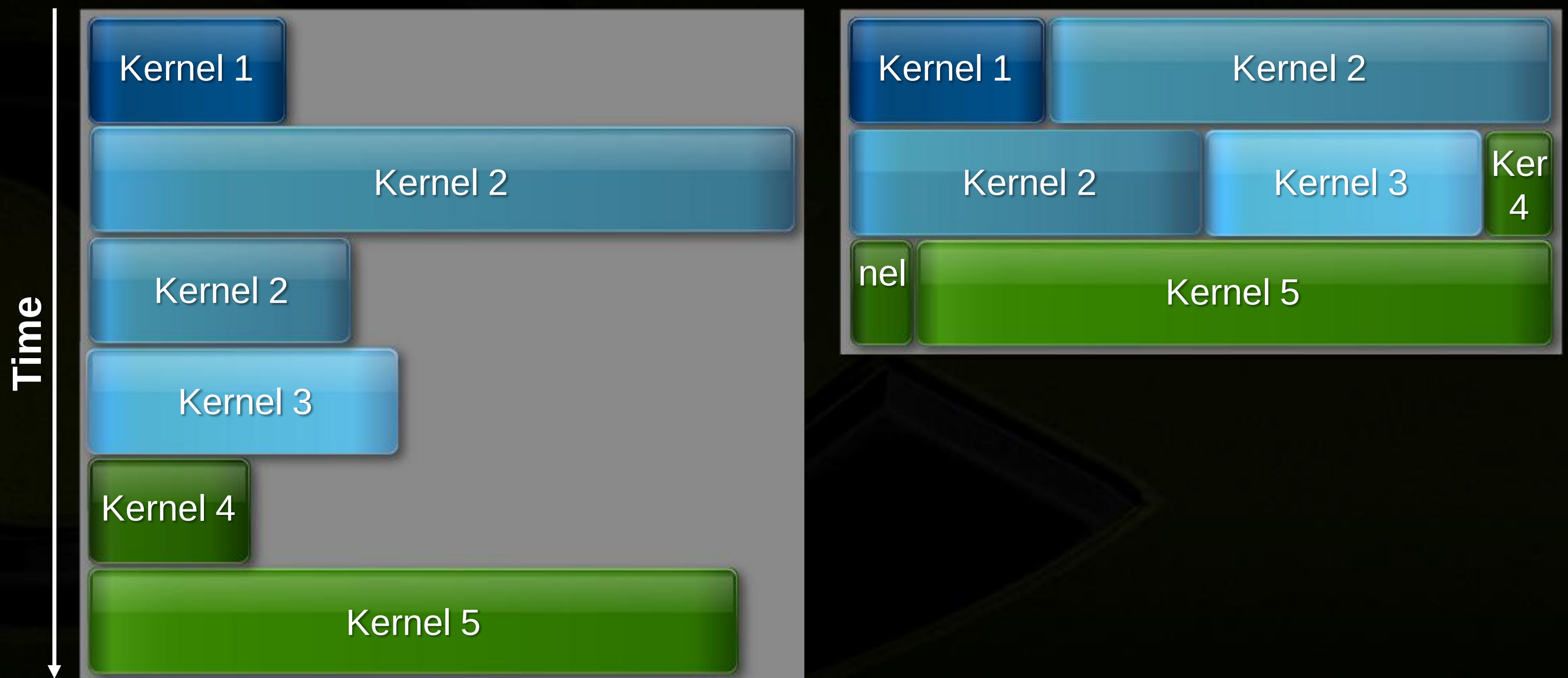
Next-Gen GPU: codename *Fermi*



- 3 billion transistors
- 512 CUDA cores
- ~2x the memory bandwidth
- L1 and L2 caches
- 8x the peak fp64 performance
- ECC
- C++

Hardware Thread Scheduling

Concurrent kernel execution + faster context switch



Serial Kernel Execution **Parallel Kernel Execution**

More Fermi Goodness

- Unified 40-bit address space for local, shared, global
- Configurable 64K L1$ / shared memory
- 10x faster atomics
- Dual DMA engines for CPU←→ GPU transfers
- IEEE 754-2008: Fused Multiply-Add (FMA) for SP, DP

Conclusion

- GPUs are massively parallel manycore computers
  - Ubiquitous - most successful parallel processor in history
  - Useful - users achieve huge speedups on real problems
- CUDA is a powerful parallel programming model
  - Heterogeneous - mixed serial-parallel programming
  - Scalable - hierarchical thread execution model
  - Accessible – many languages, OSs, vendors
- They provide tremendous scope for innovation

Questions?

dluebke@nvidia.com

NVIDIA

At the NVIDIA booth (#2365)

- **GPU Computing Poster Showcase (Monday 7pm - 9pm)**
- **Demo of Next Generation “Fermi” Architecture**
- **3D Internet Demo – Cloud Computing with NVIDIA RealityServer**
- **NVIDIA Theater, including talks by:**

Jack Dongarra (Univ of Tenn)

Bill Dally (NVIDIA)

Jeff Vetter (Oak Ridge Nat’l Lab)

Satoshi Matsuoka (Tokyo Institute of Tech)

Pat McCormick (Los Alamos Nat’l Lab)

Paul Crozier (Sandia Nat’l Lab)

Mike Clark (Harvard Univ)

Ross Walker (San Diego Supercomputing Center / UCSD)