

**nVIDIA®**

# **GPU Computing: The Democratization of Parallel Computing**

**David Luebke  
NVIDIA Research**

# Parallel Computing's Golden Age



- **1980s, early `90s: a golden age for parallel computing**
  - Particularly data-parallel computing
- **Architectures**
  - Connection Machine, MasPar, Cray
  - True supercomputers: incredibly exotic, powerful, expensive
- **Algorithms, languages, & programming models**
  - Solved a wide variety of problems
  - Various parallel algorithmic models developed
  - P-RAM, V-RAM, circuit, hypercube, etc.

# Parallel Computing's Dark Age



- **But...impact of data-parallel computing limited**
  - Thinking Machines sold 7 CM-1s (100s of systems total)
  - MasPar sold ~200 systems
- **Commercial and research activity subsided**
  - Massively-parallel machines replaced by clusters of ever-more powerful commodity microprocessors
  - Beowulf, Legion, grid computing, ...

**Massively parallel computing lost momentum to the inexorable advance of commodity technology**

# Enter the GPU



- **GPU = *Graphics Processing Unit***
  - Chip in computer video cards, PlayStation 3, Xbox, etc.
  - Two major vendors: NVIDIA and ATI (now AMD)



# Enter the GPU



- **GPUs are massively multithreaded manycore chips**
  - **NVIDIA Tesla products have up to 128 scalar processors**
  - **Over 12,000 concurrent threads in flight**
  - **Over 470 GFLOPS sustained performance**
- **Users across science & engineering disciplines are achieving 100x or better speedups on GPUs**
- **CS researchers can use GPUs as a research platform for manycore computing: arch, PL, numeric, ...**



# Enter CUDA

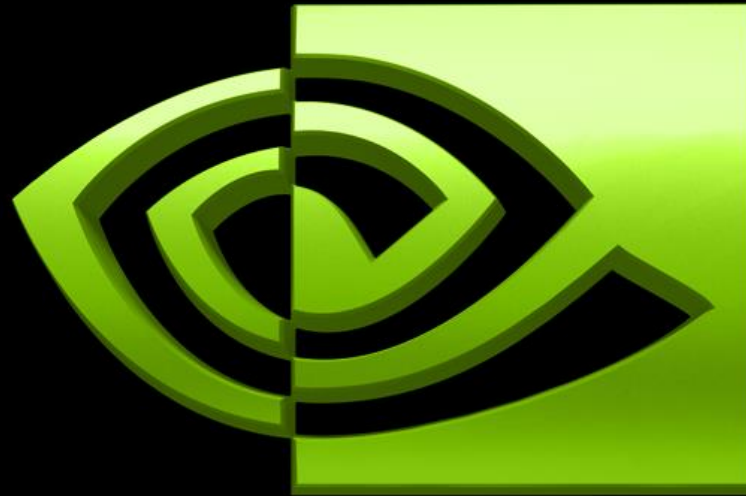


- **CUDA** is a scalable parallel programming model and a software environment for parallel computing
  - Minimal extensions to familiar C/C++ environment
  - Heterogeneous serial-parallel programming model
- NVIDIA's **TESLA** GPU architecture accelerates CUDA
  - Expose the computational horsepower of NVIDIA GPUs
  - Enable general-purpose *GPU computing*
- CUDA also maps well to multicore CPUs!

# The Democratization of Parallel Computing

- GPU Computing with CUDA brings data-parallel computing to the masses
  - Over 46,000,000 CUDA-capable GPUs sold
  - A “developer kit” costs ~\$200 (for 500 GFLOPS)
- Data-parallel supercomputers are everywhere!
  - CUDA makes this power accessible
  - We’re already seeing innovations in data-parallel computing

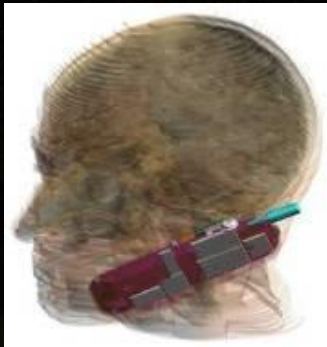
**Massively parallel computing has become a commodity technology!**



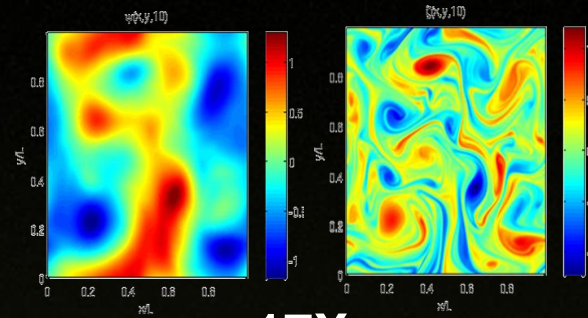
**nVIDIA®**

# **GPU Computing: Motivation**

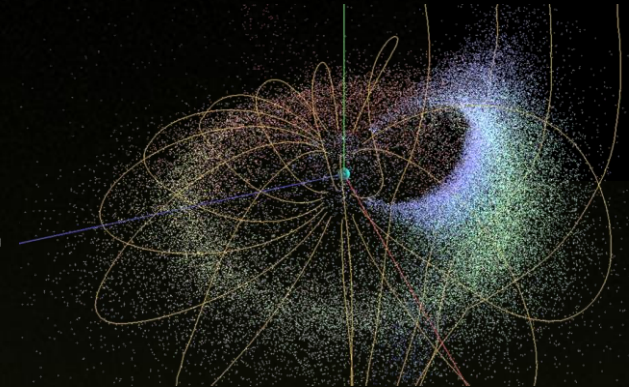




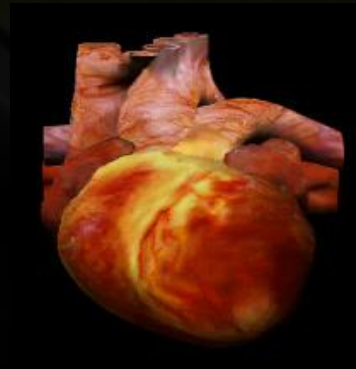
45X



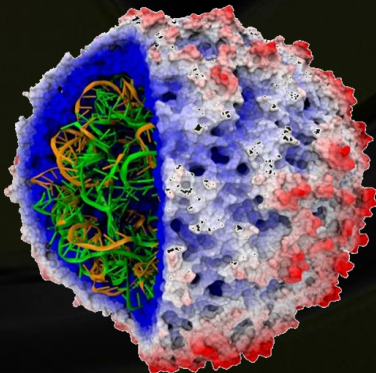
17X



100X

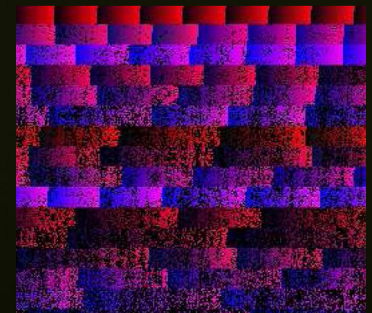


13–457x



110–240X

# GPU Computing: Motivation



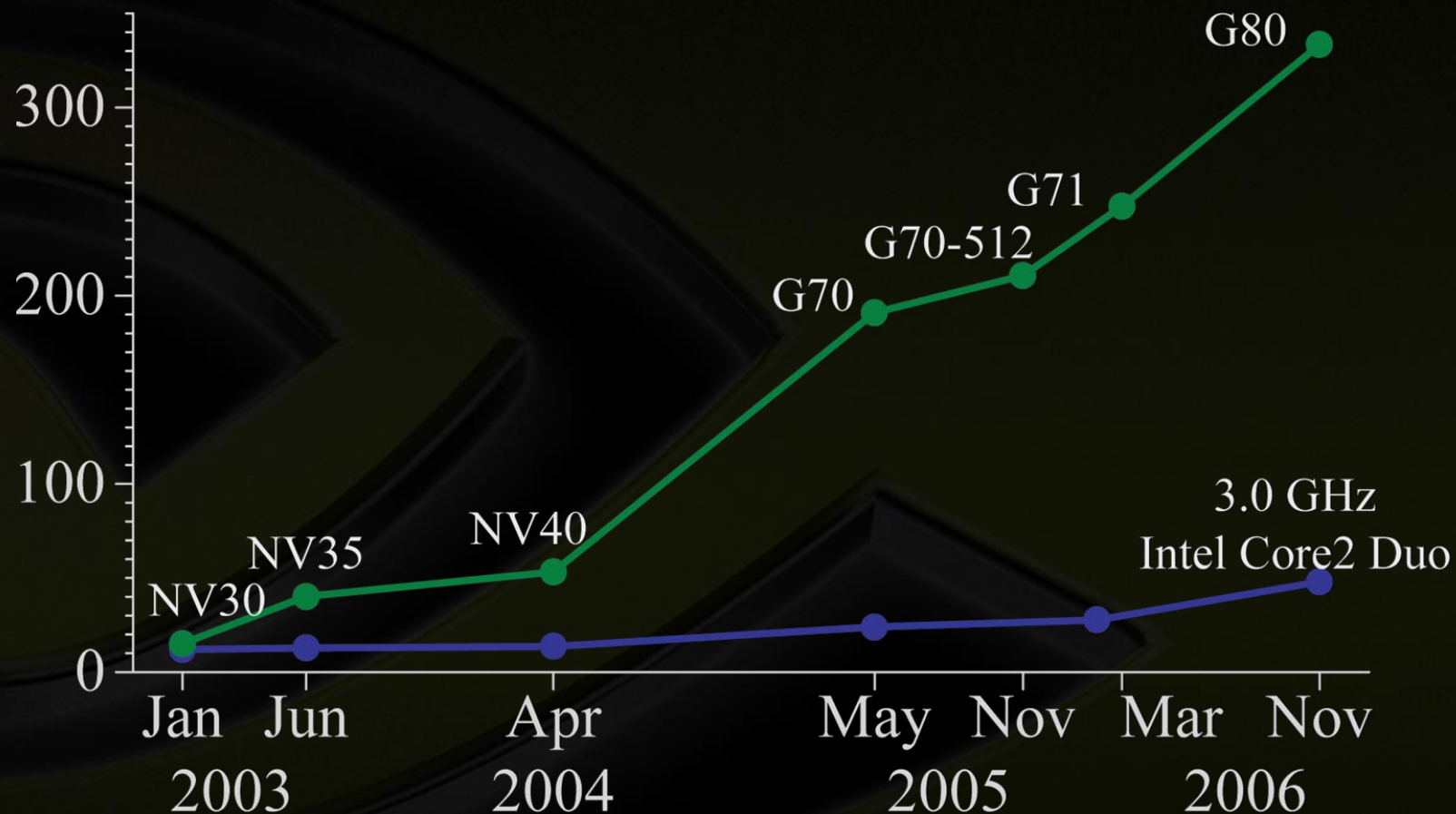
35X

# GPUs Are Fast



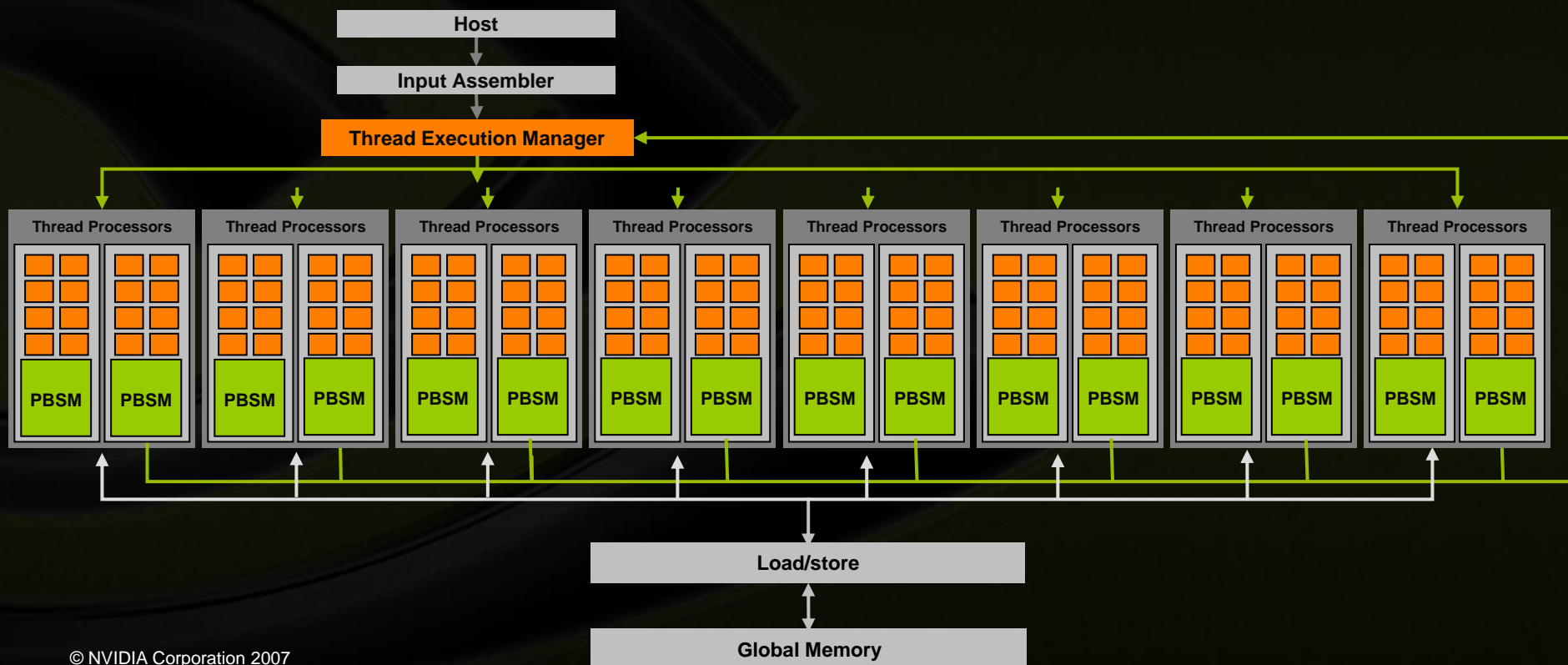
- **Theoretical peak performance: 518 GFLOPS**
- **Sustained  $\mu$ benchmark performance:**
  - Raw math: 472 GFLOPS (8800 Ultra)
  - Raw bandwidth: 80 GB per second (Tesla C870)
- **Actual application performance:**
  - Molecular dynamics: 290 GFLOPS (VMD ion placement)

# GPUs Are Getting Faster, Faster

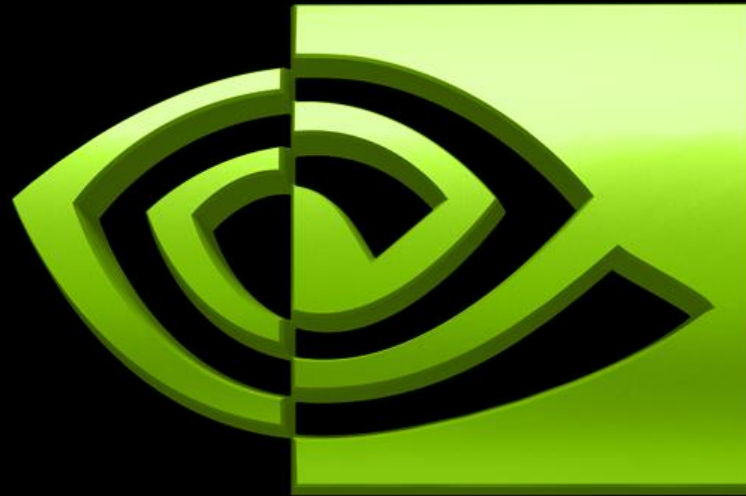


# Manycore GPU – Block Diagram

- G80 (launched Nov 2006 – GeForce 8800 GTX)
- 128 Thread Processors execute kernel threads
- Up to 12,288 parallel threads active
- Per-block shared memory (PBSM) accelerates processing







**nVIDIA®**

**CUDA Programming Model**



# Heterogeneous Programming

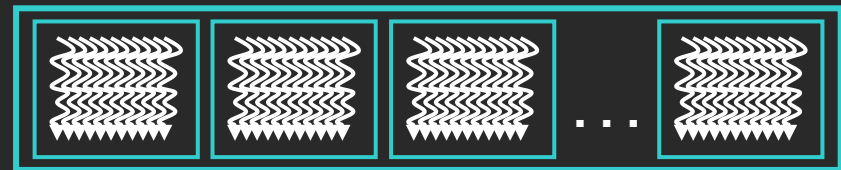


- CUDA = serial program with parallel kernels, all in C
  - Serial C code executes in a CPU thread
  - Parallel kernel C code executes in *thread blocks* across multiple processing elements

Serial Code

Parallel Kernel

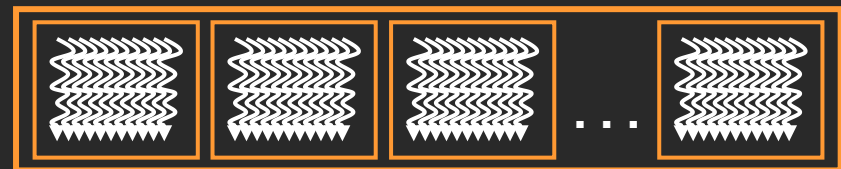
```
KernelA<<< nBlk, nTid >>>(args);
```



Serial Code

Parallel Kernel

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



# GPU Computing with CUDA: A Highly Multithreaded Coprocessor

- The GPU is a highly parallel **compute device**
  - serves as a coprocessor for the **host** CPU
  - has its own **device memory** on the card
  - executes many **threads** in parallel
- Parallel **kernels** run a single program in many threads
- GPU threads are extremely lightweight
  - Thread creation and context switching are essentially free
- GPU expects 1000's of threads for full utilization

# CUDA: Programming GPU in C



- Philosophy: provide minimal set of extensions necessary to expose power

- Declaration specifiers to indicate where things live

```
__global__ void KernelFunc(...);    // kernel function, runs on device
__device__ int GlobalVar;           // variable in device memory
__shared__ int SharedVar;           // variable in per-block shared memory
```

- Extend function invocation syntax for parallel kernel launch

```
KernelFunc<<<500, 128>>>(...);    // launch 500 blocks w/ 128 threads each
```

- Special variables for thread identification in kernels

```
dim3 threadIdx; dim3 blockIdx; dim3 blockDim; dim3 gridDim;
```

- Intrinsics that expose specific operations in kernel code

```
__syncthreads();                    // barrier synchronization within kernel
```

# Decoder Ring



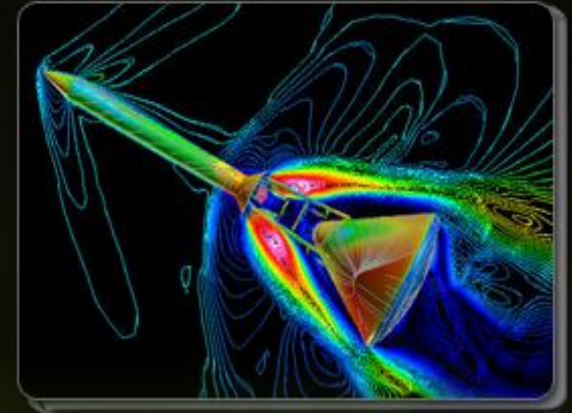
**GeForce®**  
Entertainment



**Quadro®**  
Design & Creation



**Tesla™**  
High Performance Computing



**GPU**

**Architecture: TESLA**  
**Chips: G80, G84, G92, ...**



# A New Platform: Tesla



- HPC-oriented product line
  - C870: board (1 GPU)
  - D870: desktide unit (2 GPUs)
  - S870: 1u server unit (4 GPUs)





# 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 - minimal but expressive changes to C
- **They provide tremendous scope for innovative, impactful research**