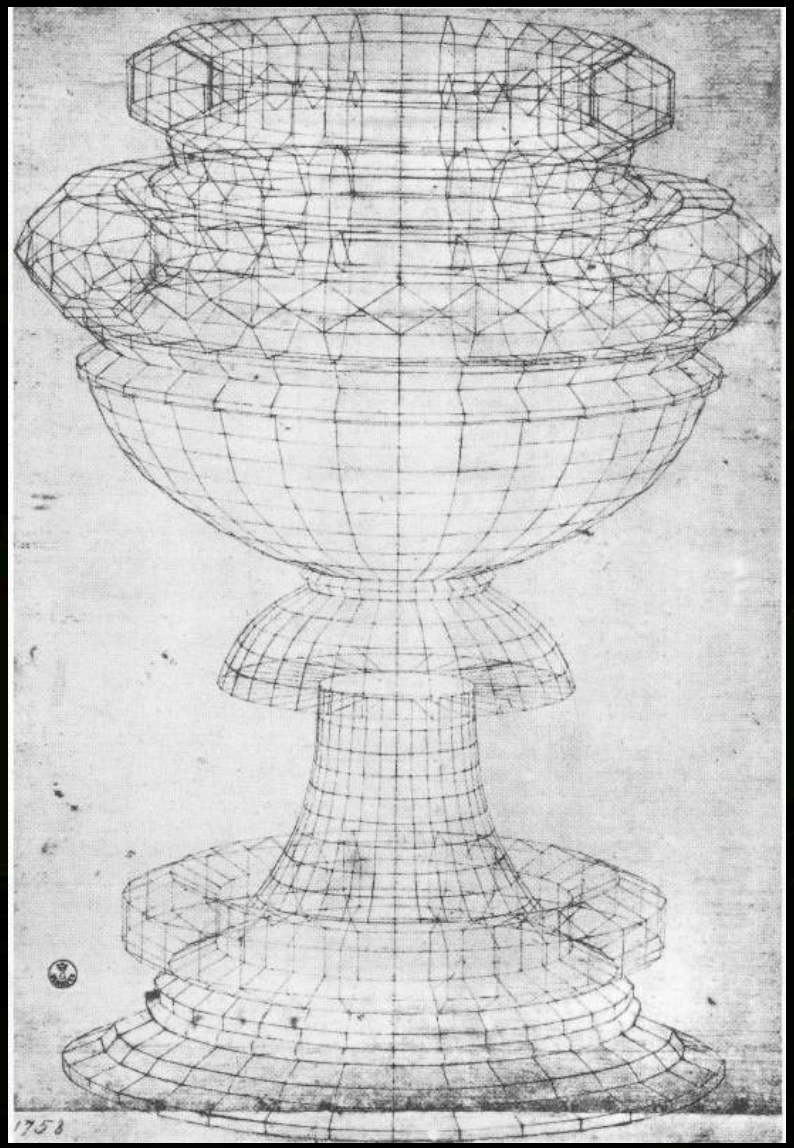

GPU Computing 3.0
The Past, Present, and
Future of GPU Computing
David Luebke
NVIDIA

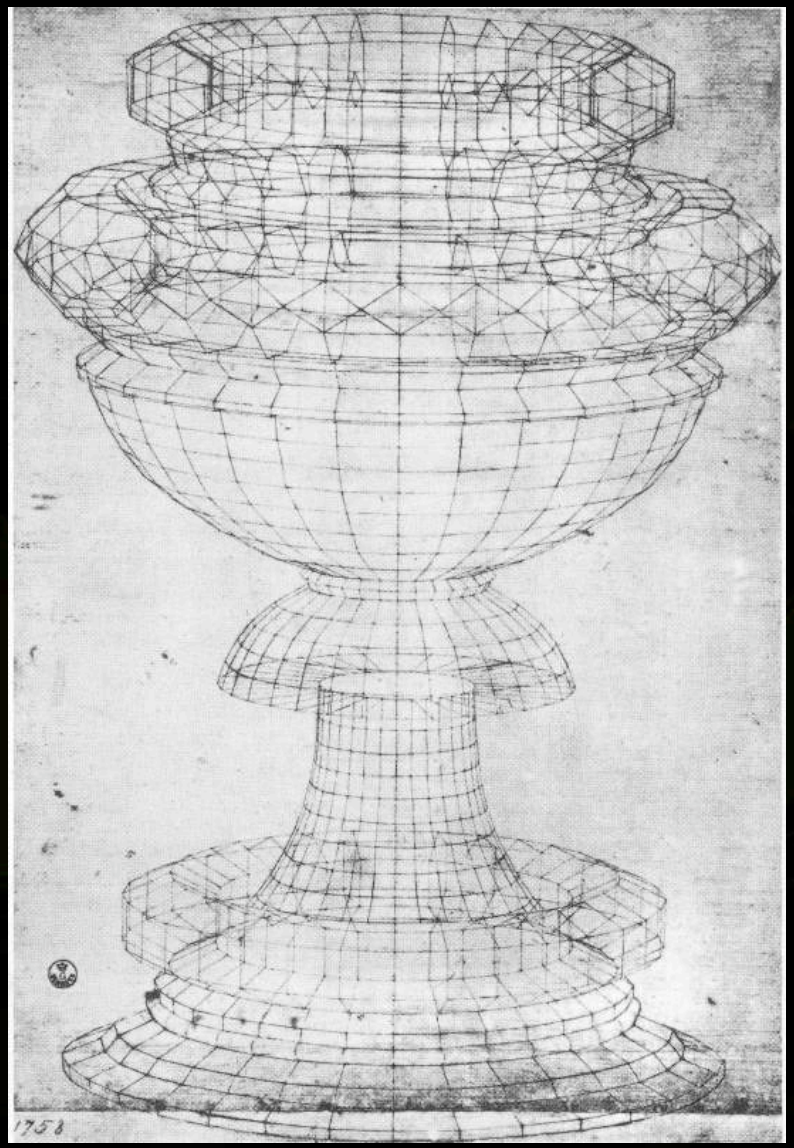

Evolution of
Graphics Hardware
NVIDIA

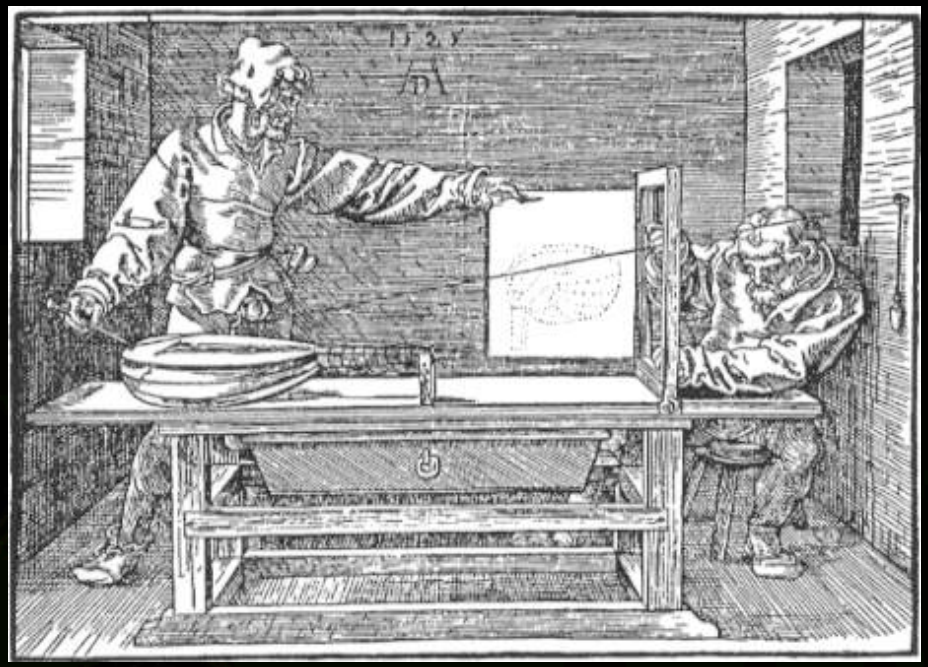
Early 3D Graphics



Perspective study of a chalice
Paolo Uccello, circa 1450

Early Graphics Hardware





Artist using a perspective machine
Albrecht Dürer, 1525

Perspective study of a chalice
Paolo Uccello, circa 1450

Early *Electronic* Graphics Hardware

SKETCHPAD: A Man-Machine Graphical Communication System
Ivan Sutherland, 1963

The Graphics Pipeline

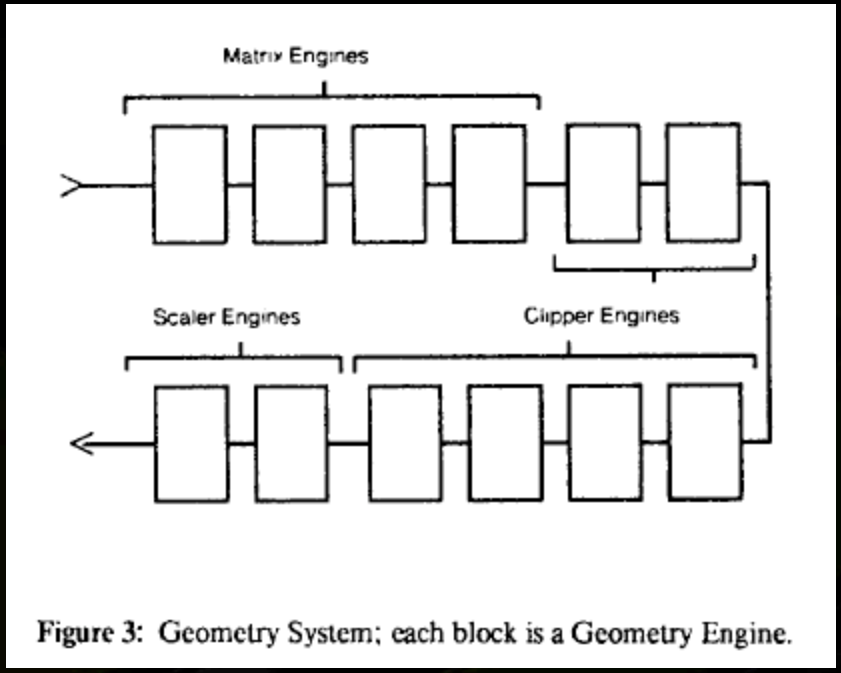
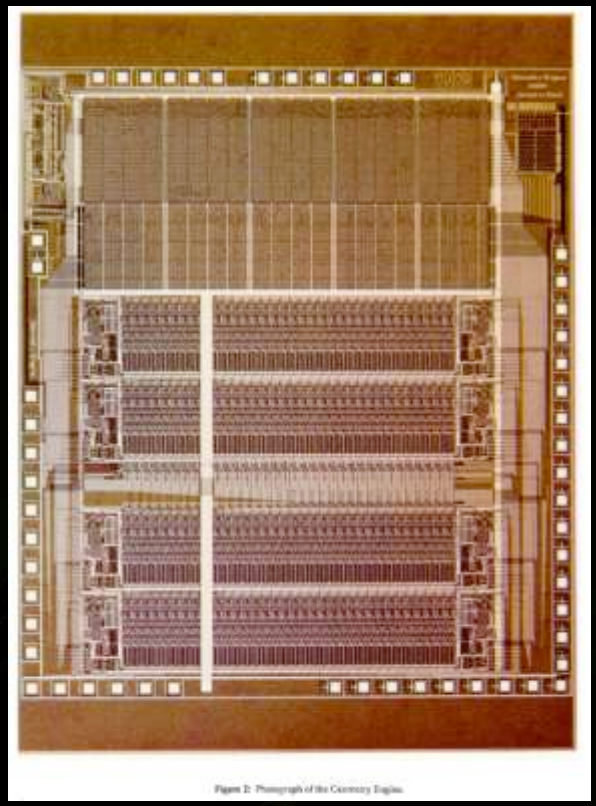
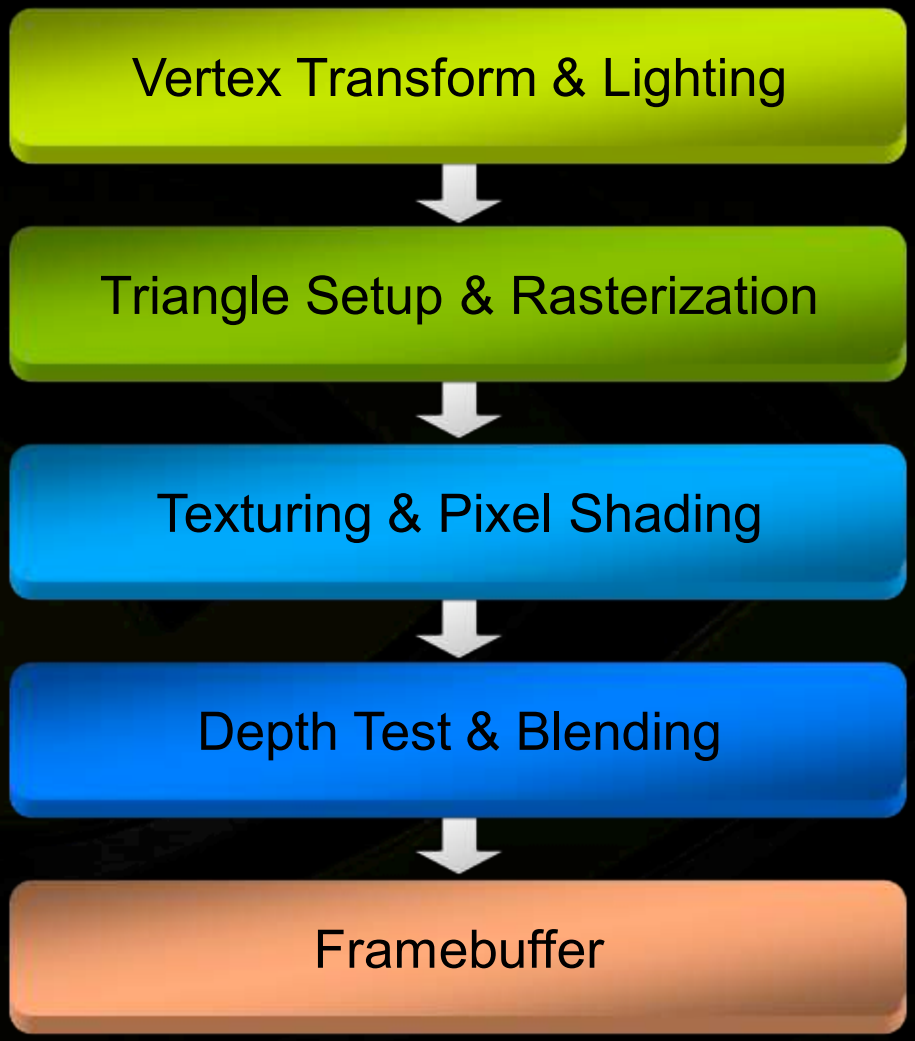


Figure 3: Geometry System; each block is a Geometry Engine.

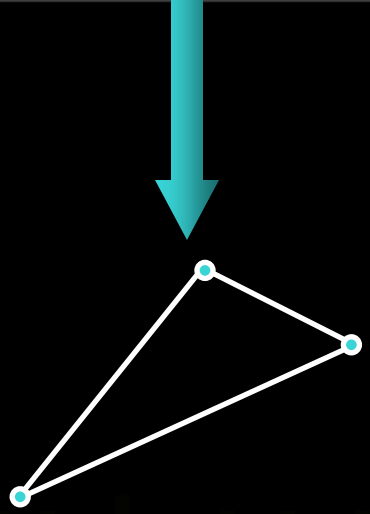


The Geometry Engine: A VLSI Geometry System for Graphics
Jim Clark, 1982

The Graphics Pipeline

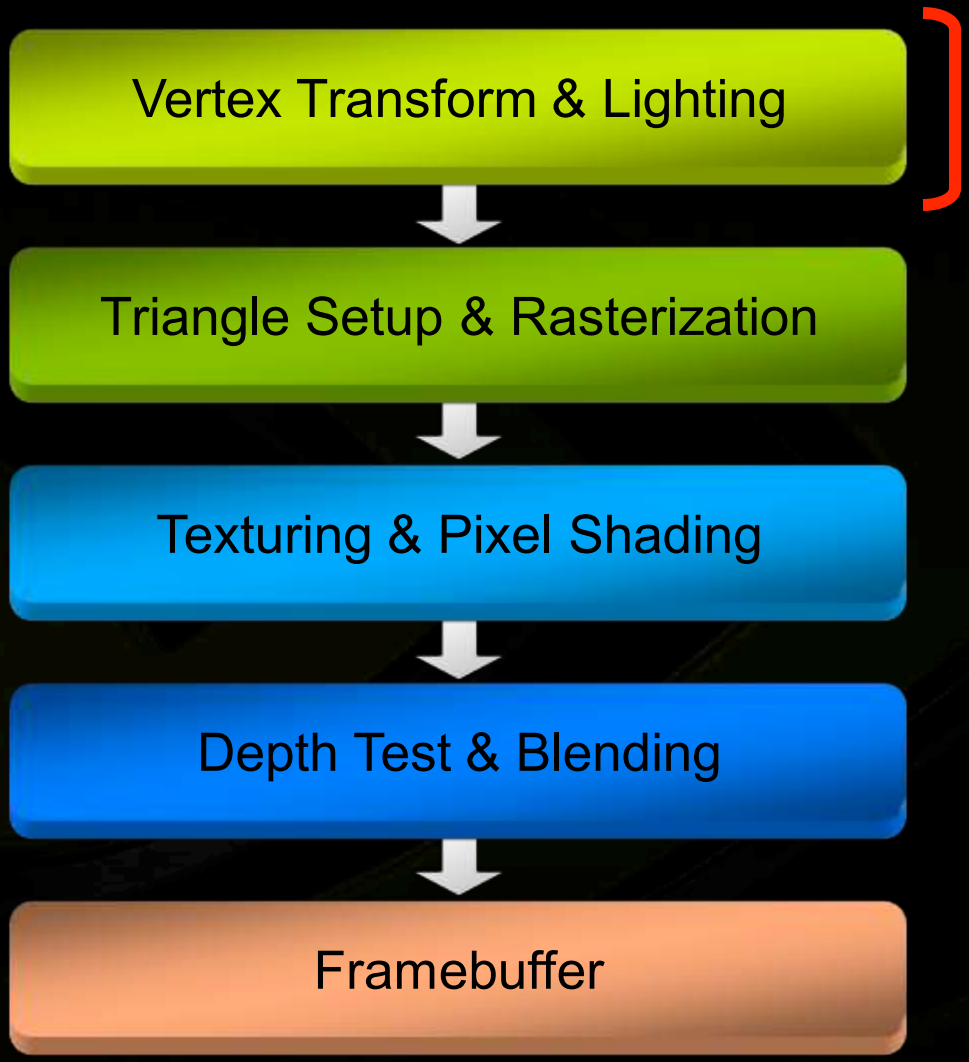


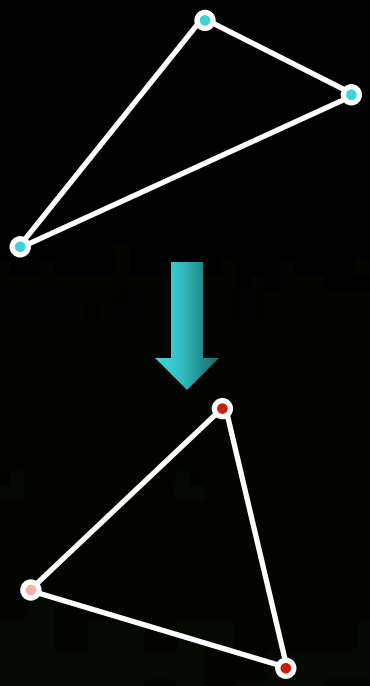
The Graphics Pipeline



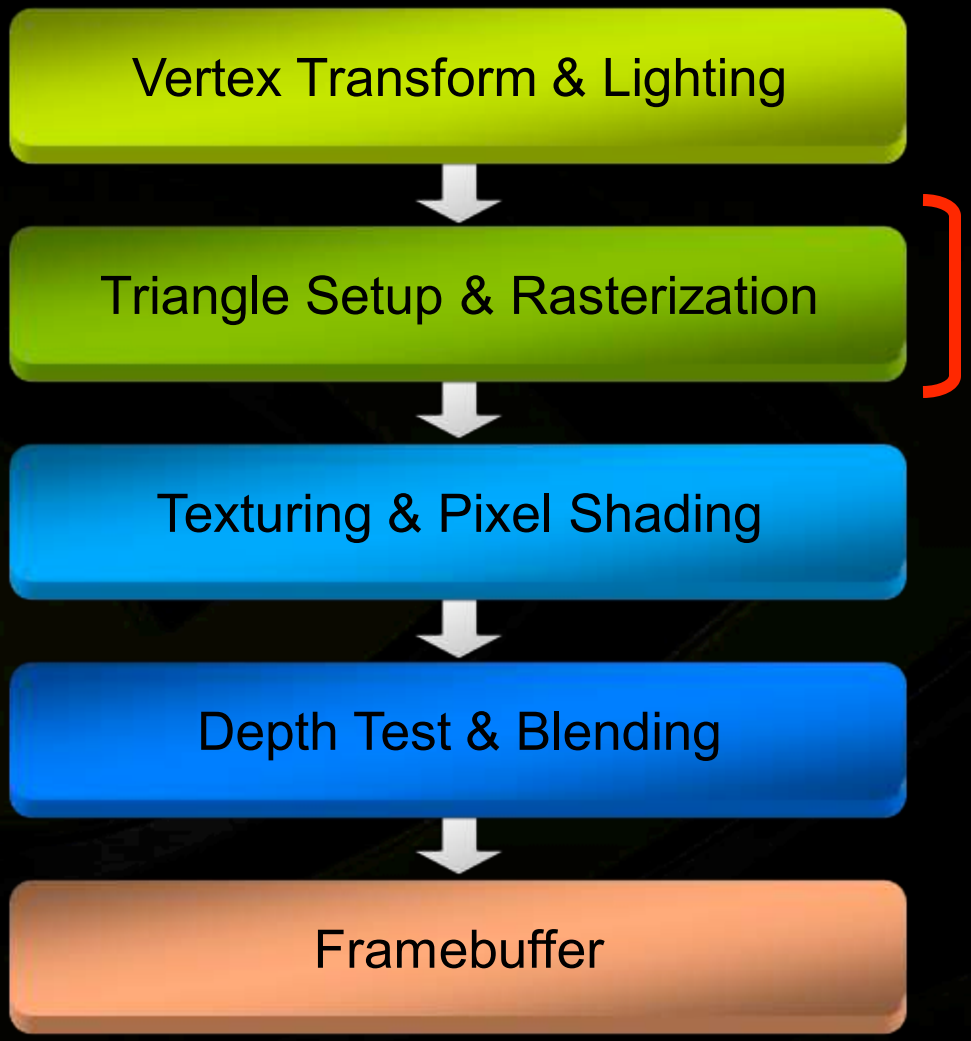
- Vertex Transform & Lighting
- Triangle Setup & Rasterization
- Texturing & Pixel Shading
- Depth Test & Blending
- Framebuffer

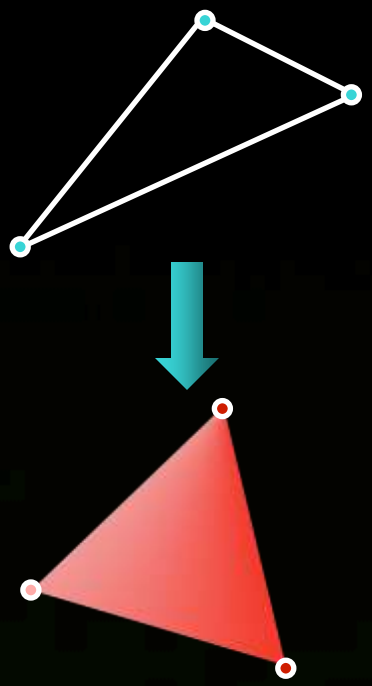
The Graphics Pipeline



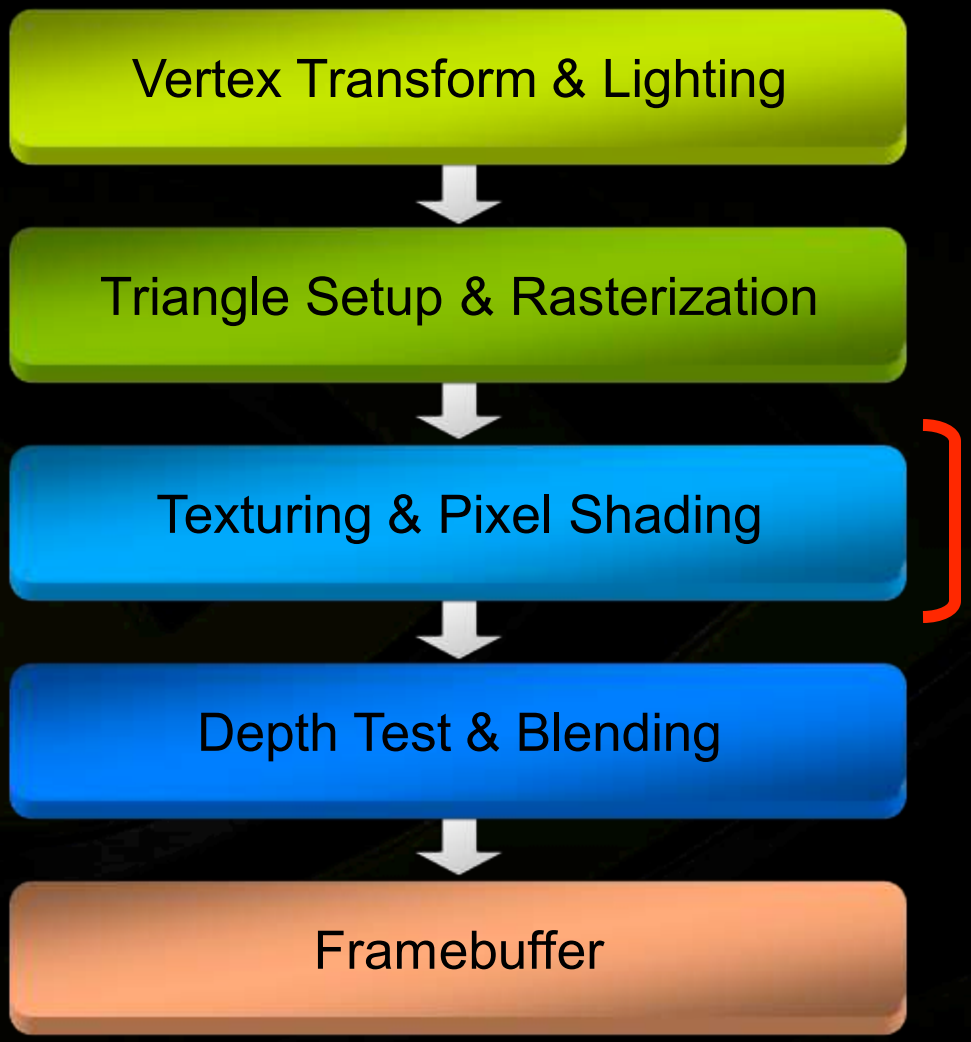


The Graphics Pipeline

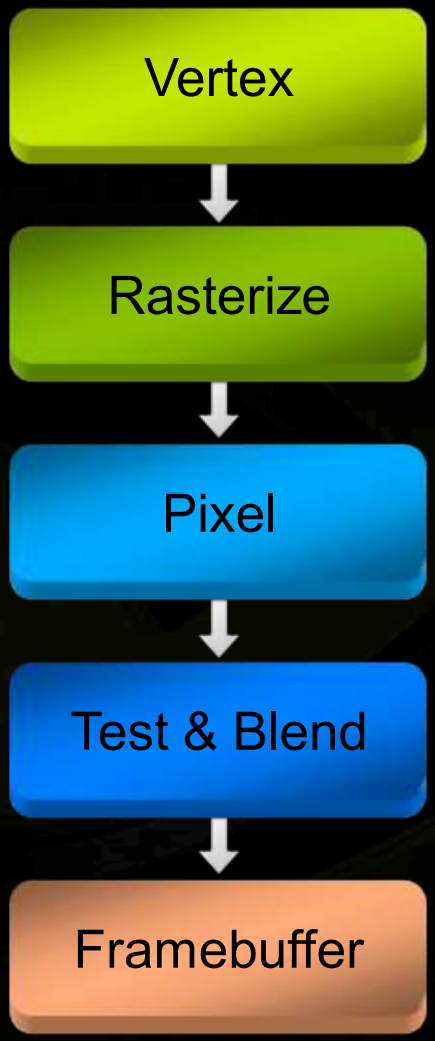




The Graphics Pipeline

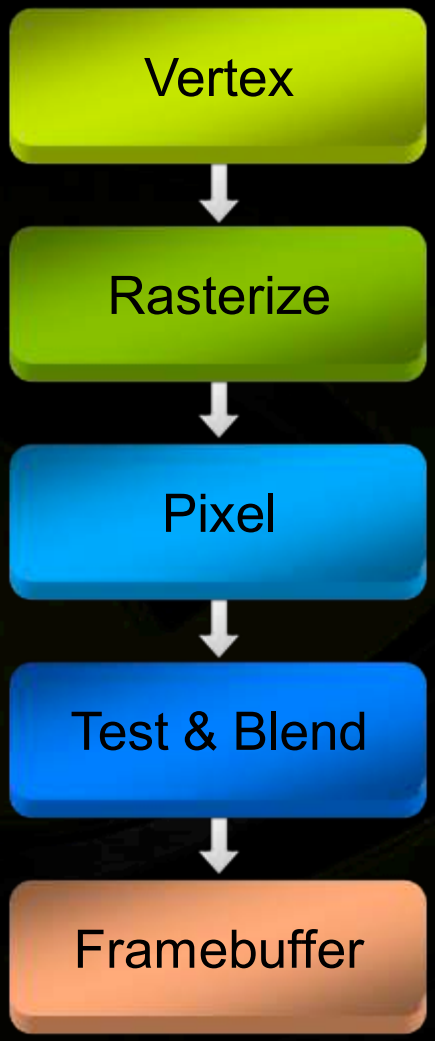


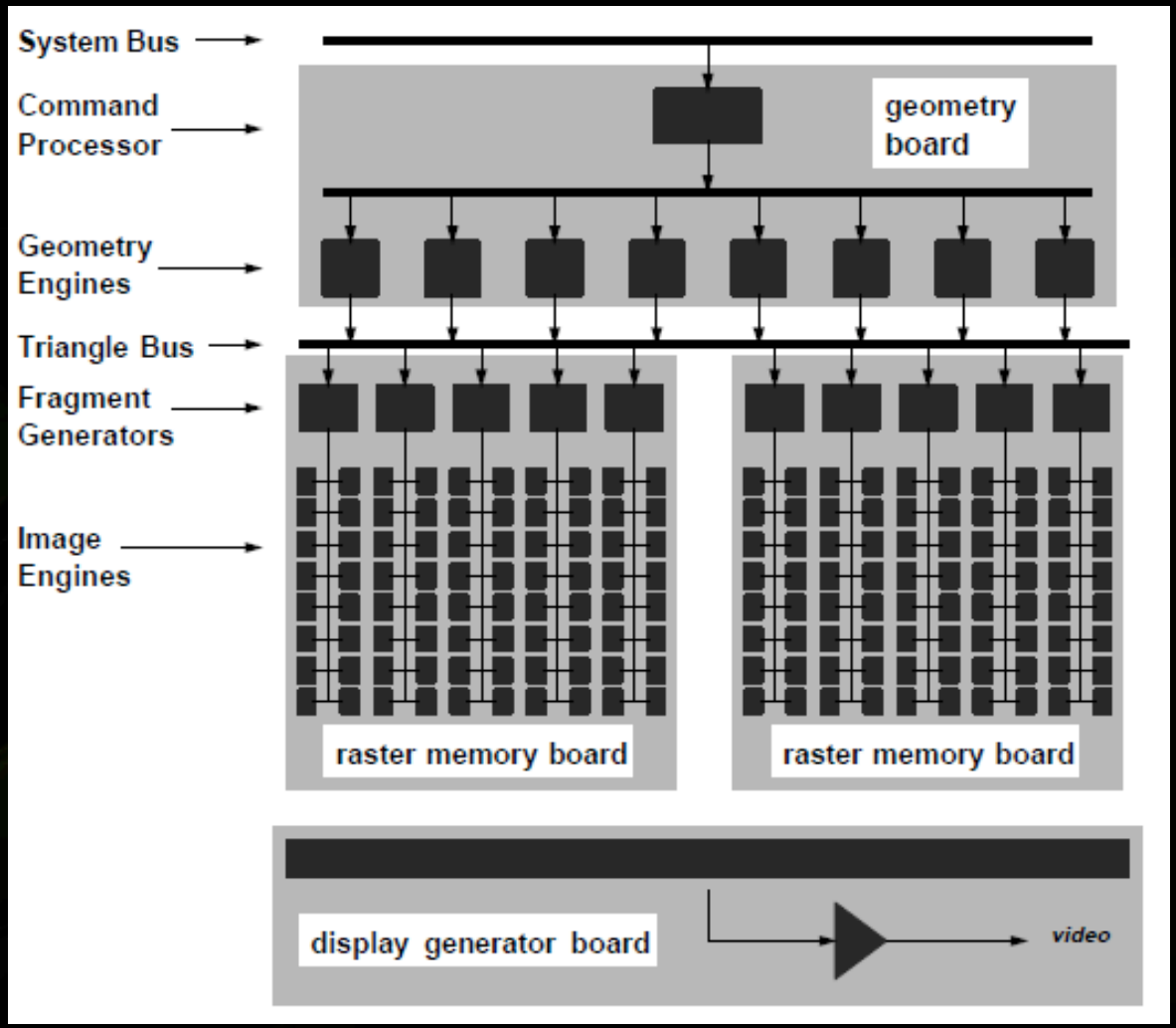
The Graphics Pipeline



- Key abstraction of real-time graphics
- Hardware used to look like this
- One chip/board per stage
- Fixed data flow through pipeline

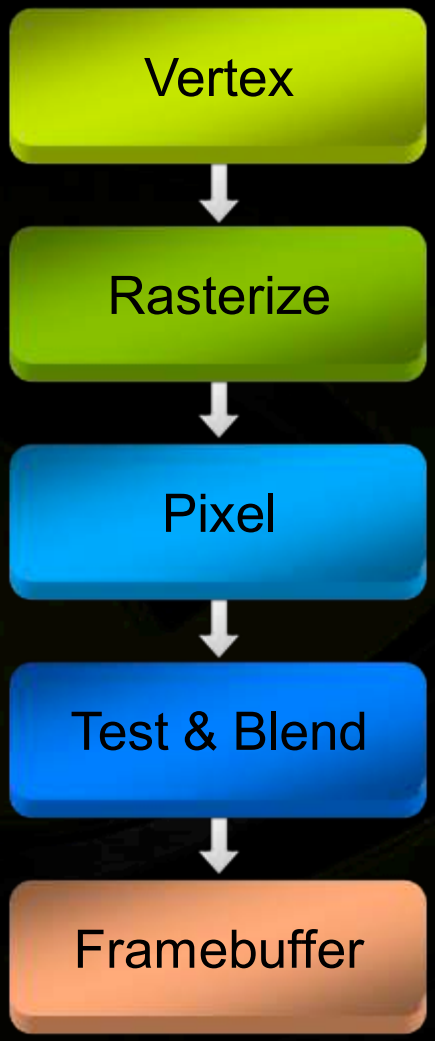
SGI RealityEngine (1993)

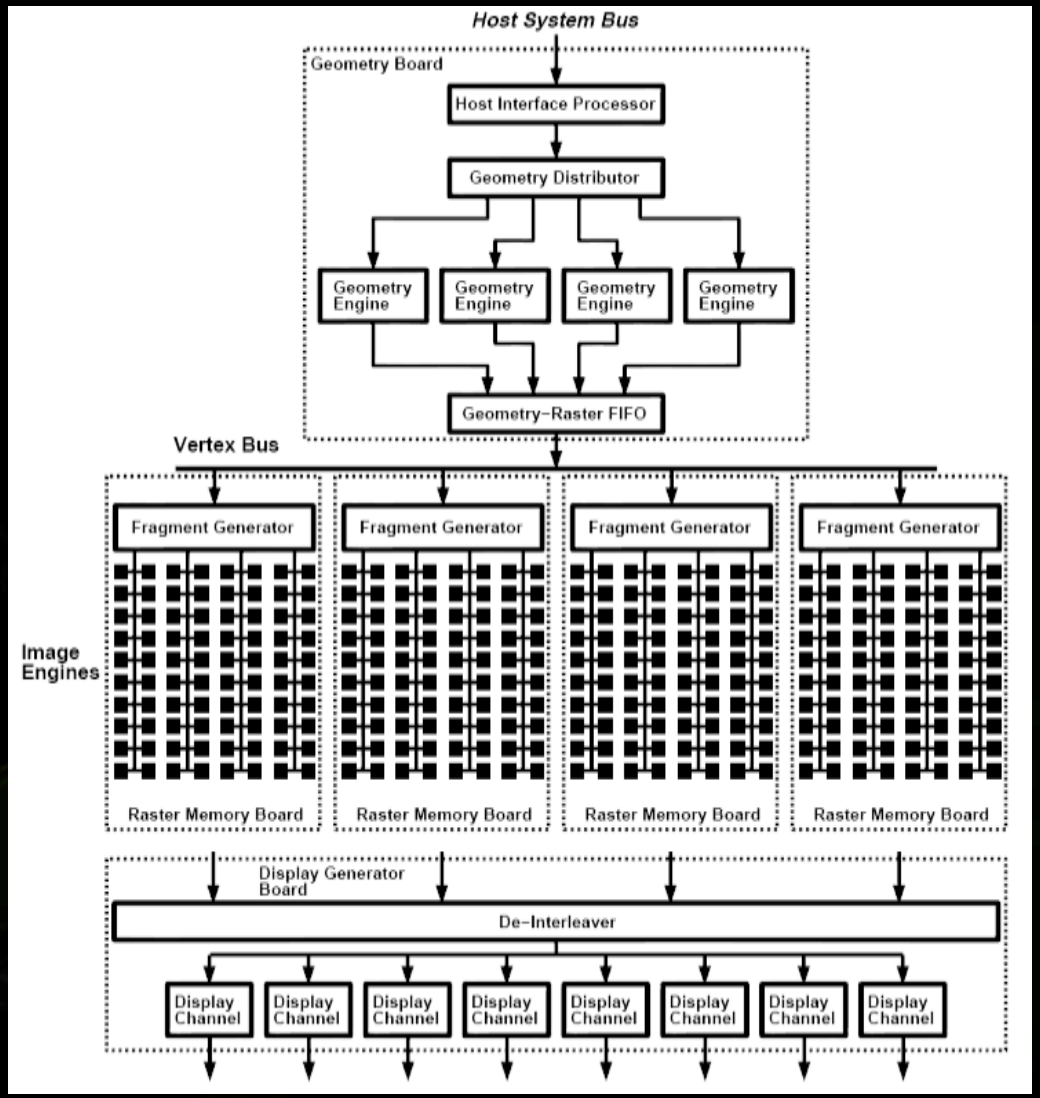




Kurt Akeley. *RealityEngine Graphics*, SIGGRAPH 93.

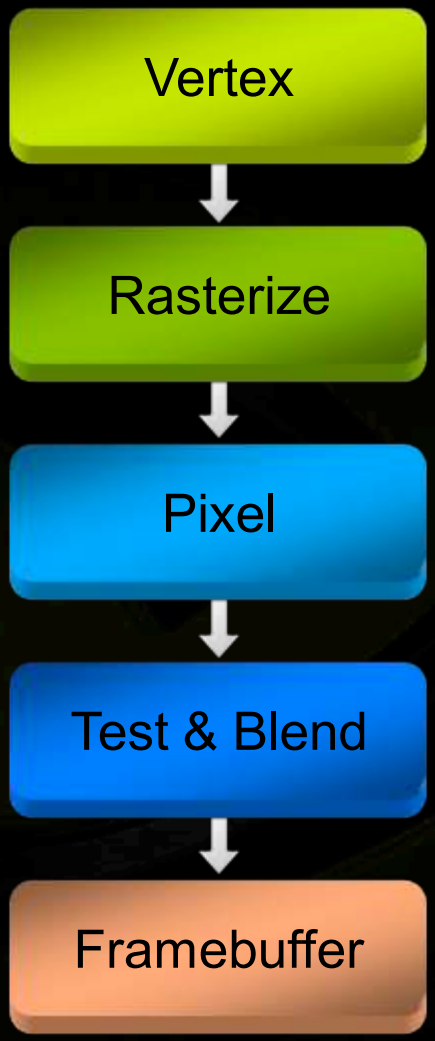
SGI InfiniteReality (1997)





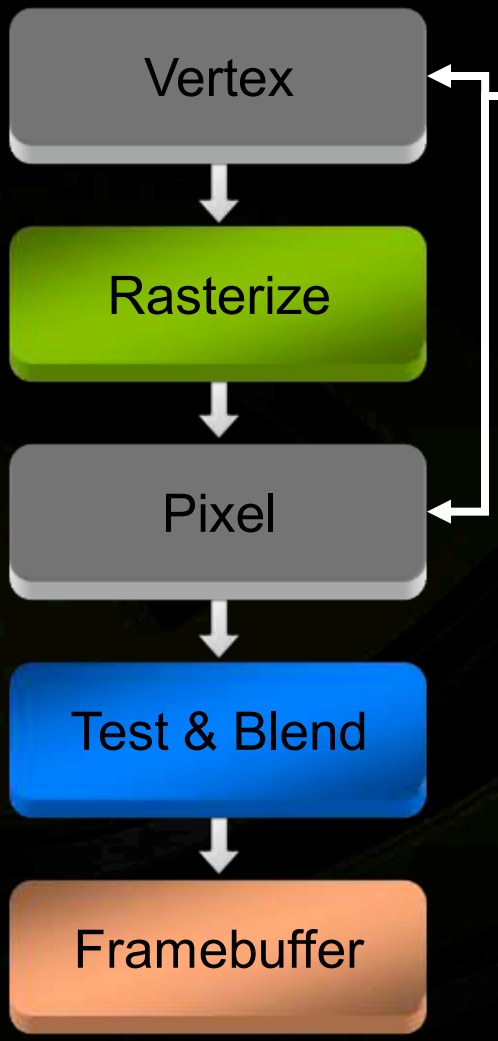
Montrym et al. *InfiniteReality: A real-time graphics system*, SIGGRAPH 97

The Graphics Pipeline



- Remains a useful abstraction
- Hardware used to look like this

The Graphics Pipeline

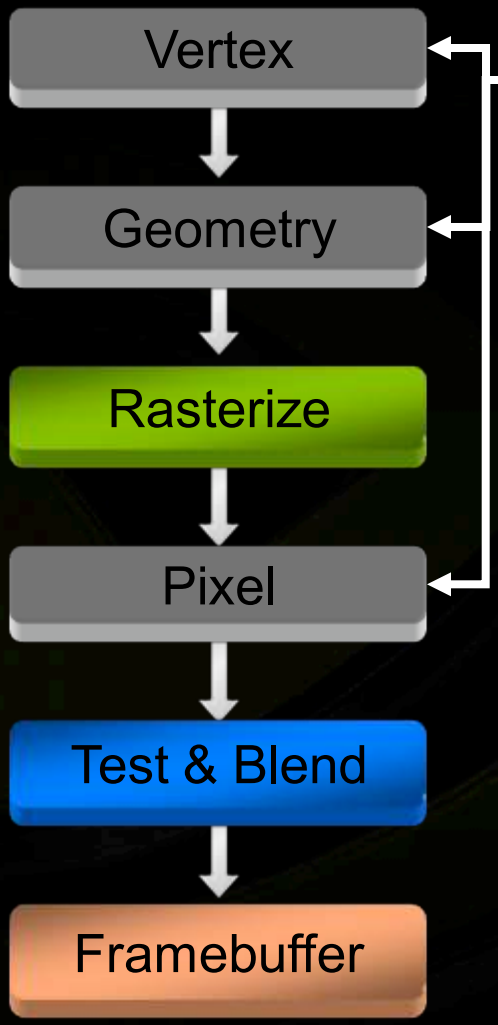



```
// Each thread performs one pair-wise addition
__global__      vecAdd(float

          threadIdx.x   blockDim.x   blockIdx.x
```

- **Hardware used to look like this:**
  - **Vertex, pixel processing became programmable**

The Graphics Pipeline

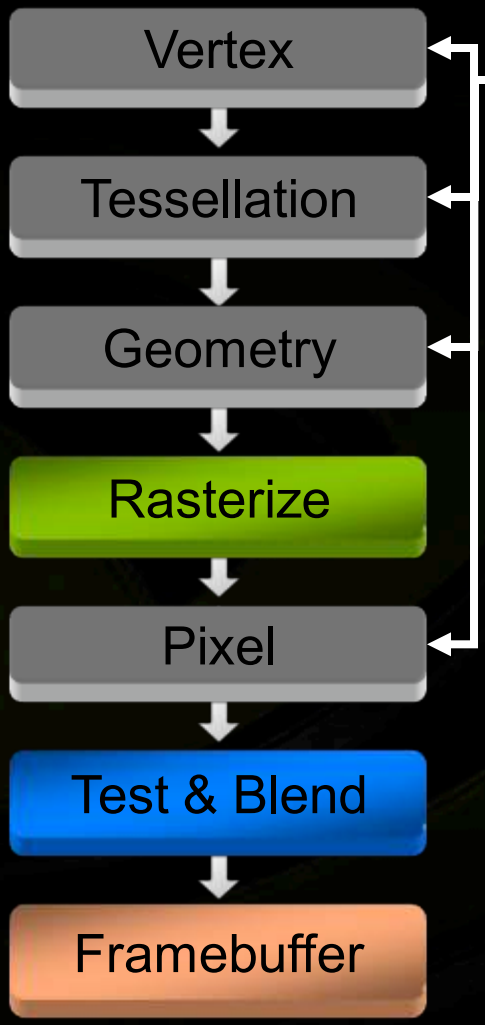



```
// Each thread performs one pair-wise addition
__global__      vecAdd(float

          threadIdx.x   blockDim.x   blockIdx.x
```

- Hardware used to look like this
  - Vertex, pixel processing became programmable
  - New stages added

The Graphics Pipeline



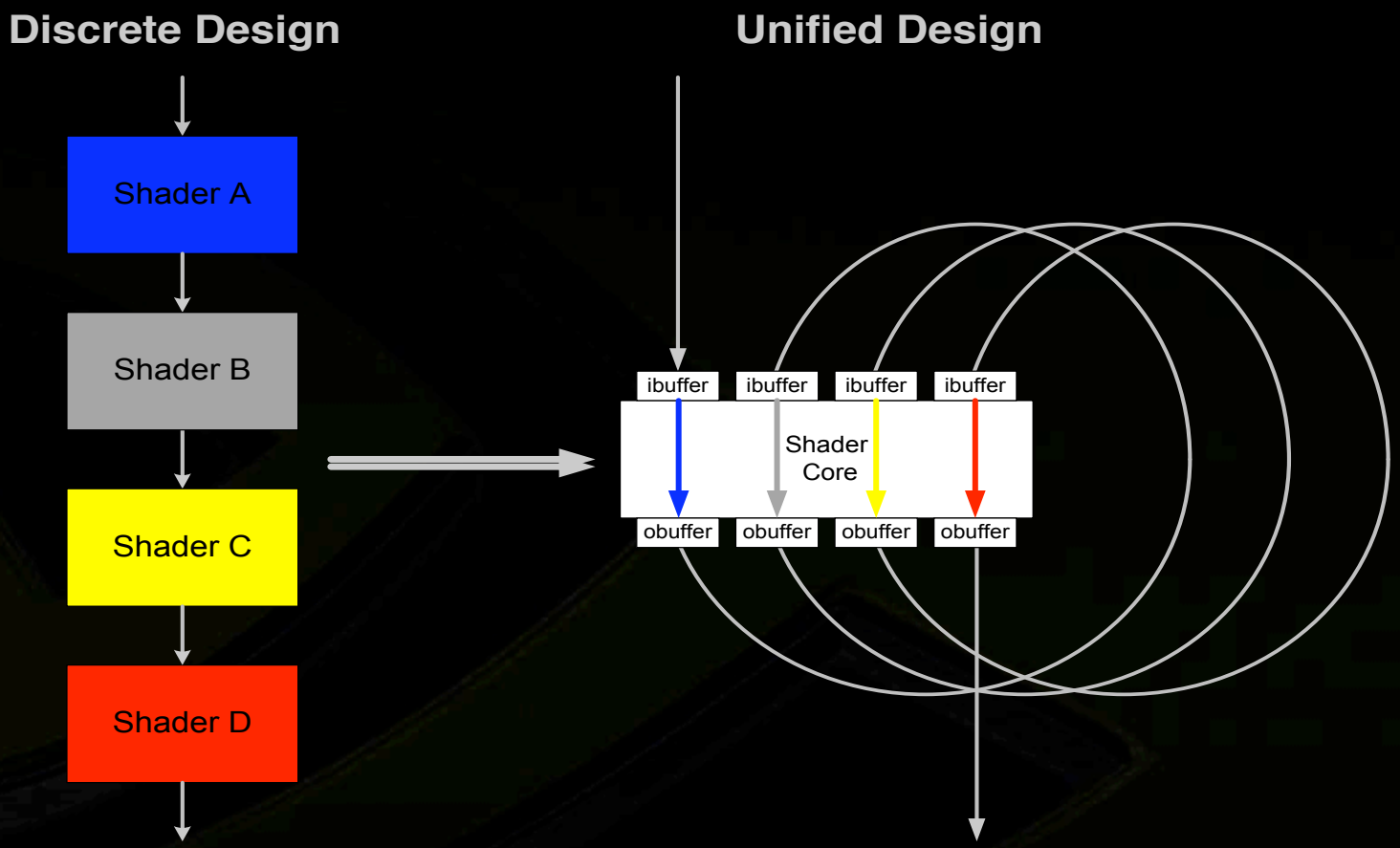

```
// Each thread performs one pair-wise addition
__global__      vecAdd(float

          threadIdx.x   blockDim.x   blockIdx.x
```

- **Hardware used to look like this**
  - **Vertex, pixel processing became programmable**
  - **New stages added**

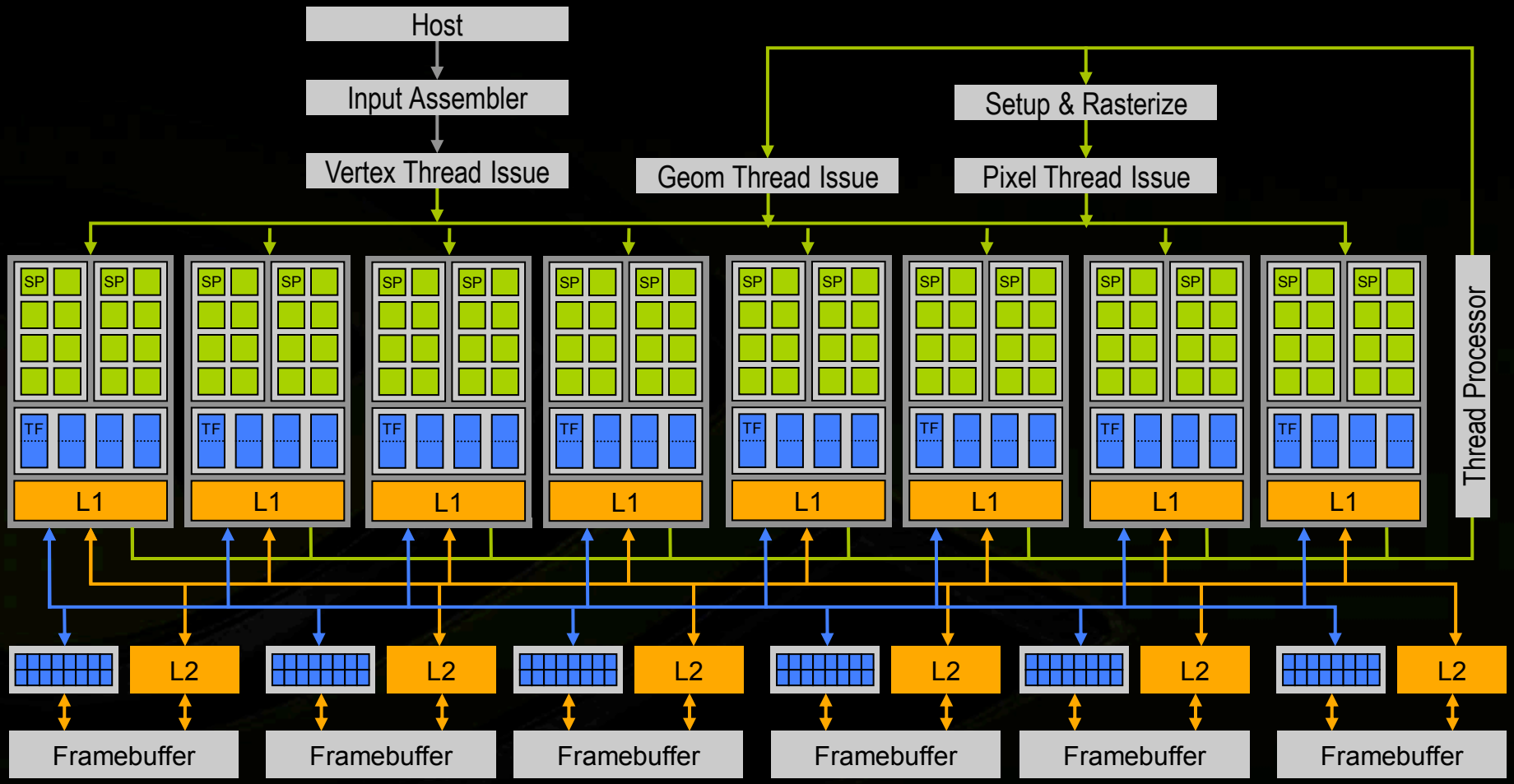
GPU architecture increasingly centers around shader execution

Modern GPUs: Unified Design

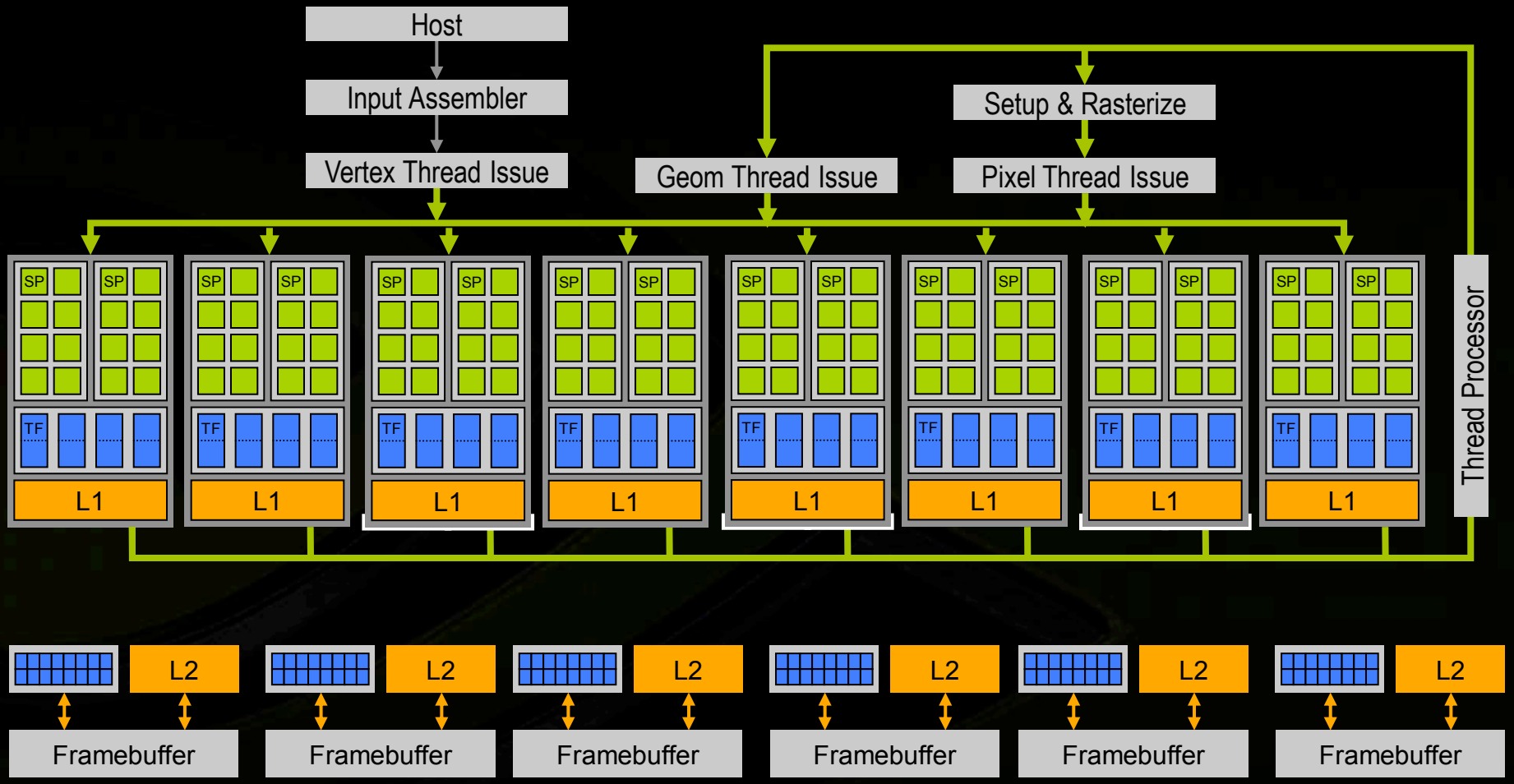


Vertex shaders, pixel shaders, etc. become *threads* running different programs on a flexible core

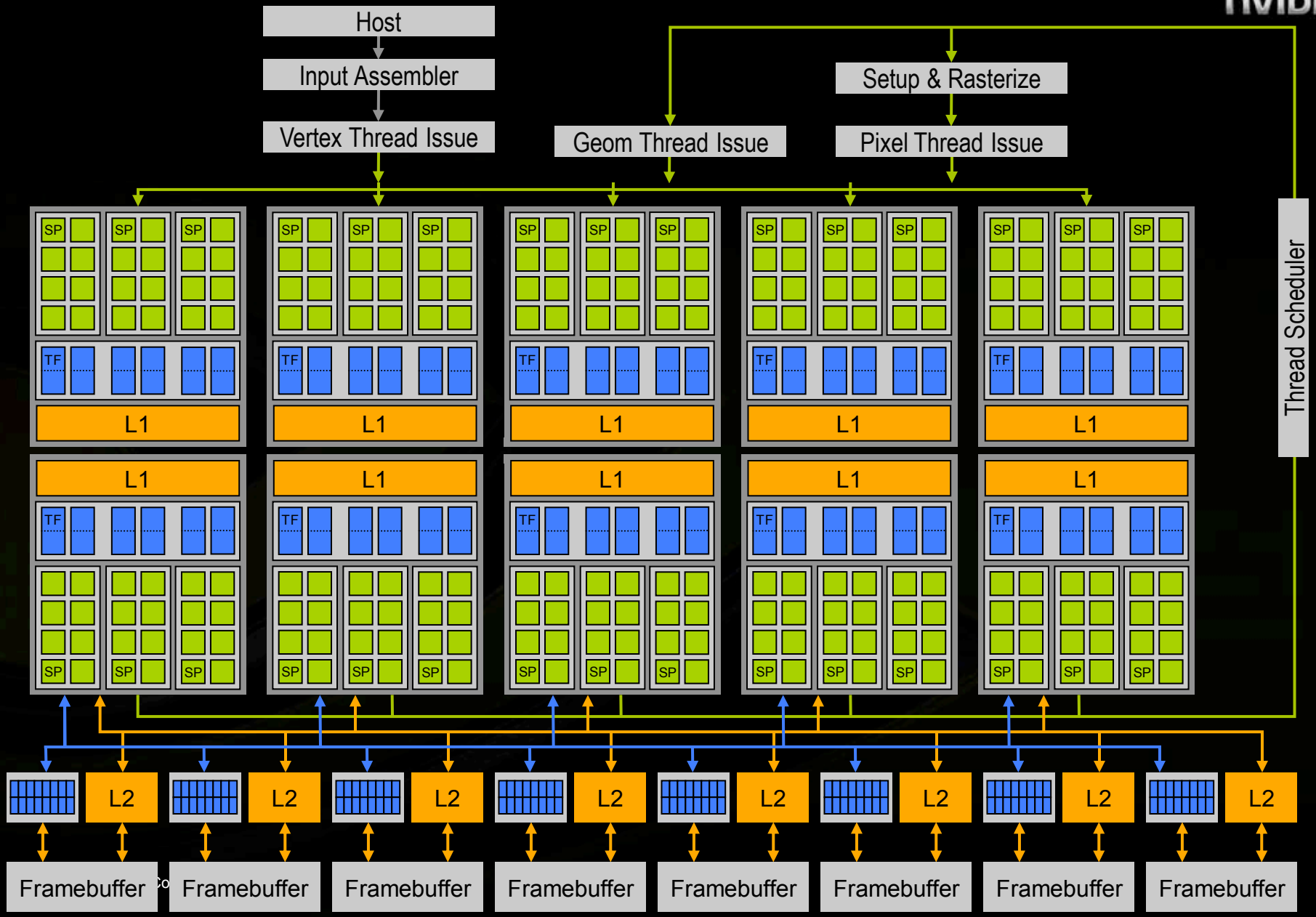
GeForce 8: Modern GPU Architecture



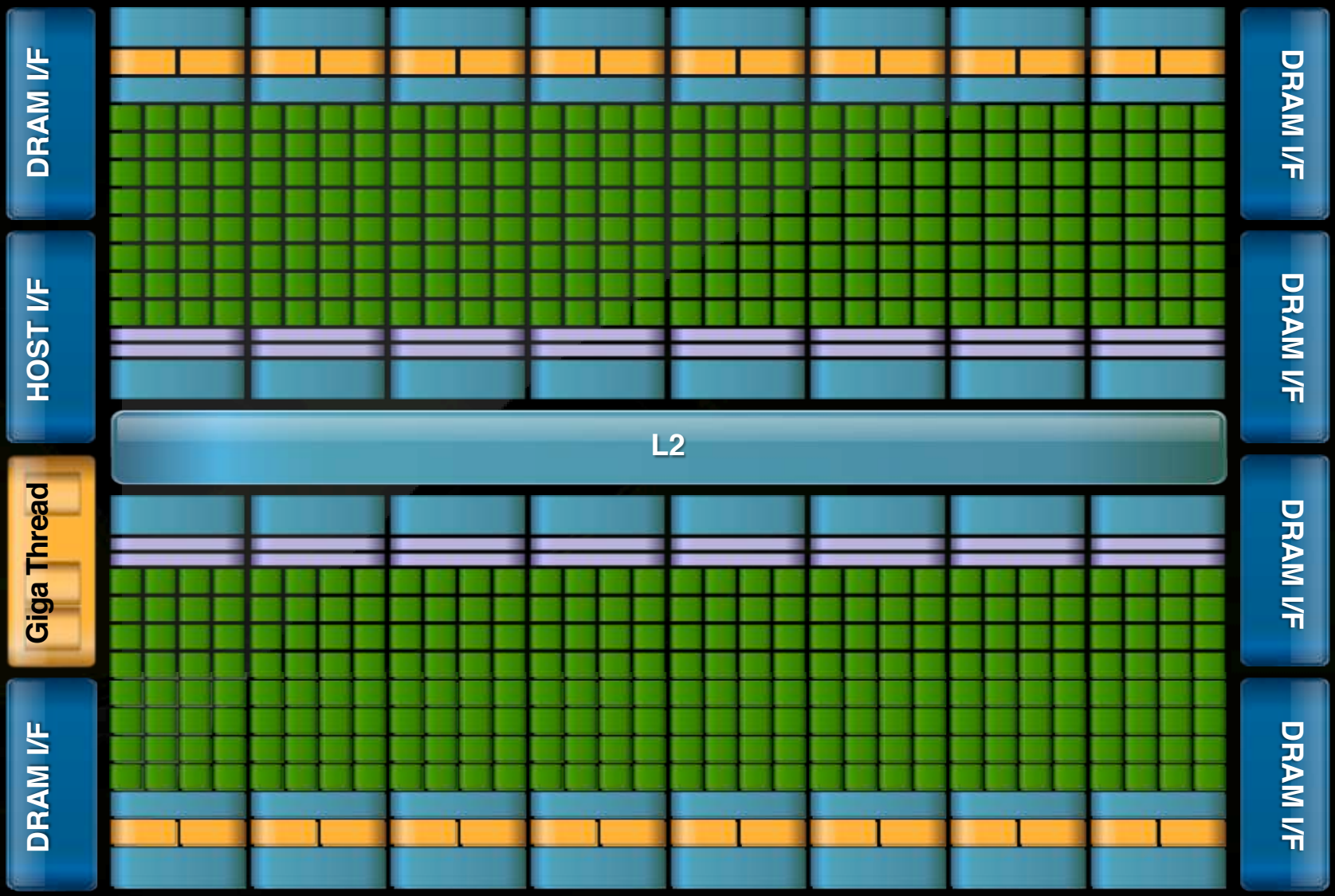
GeForce 8: Modern GPU Architecture



Modern GPU Architecture: GT200



Next-Gen GPU Architecture: *Fermi*

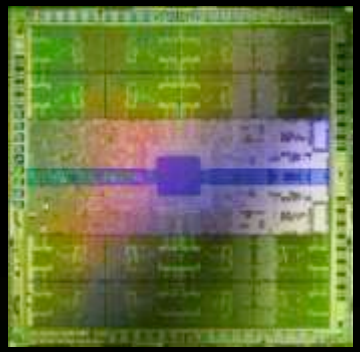


NVIDIA next-gen "Fermi" architecture

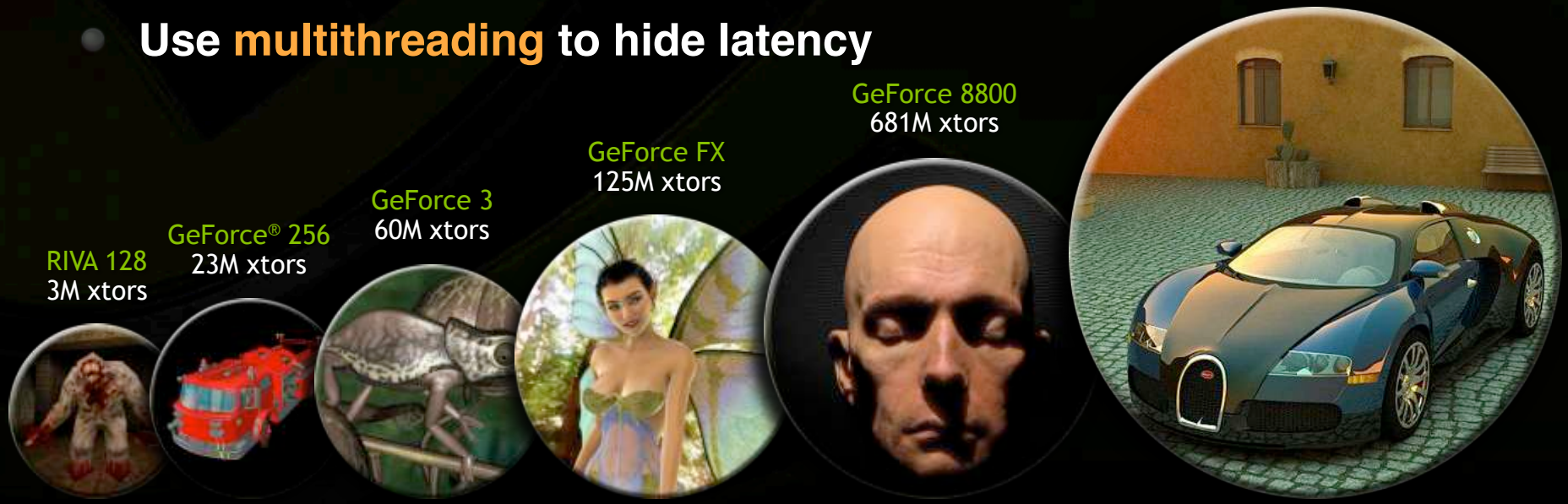
GPUs Today

Lessons from Graphics Pipeline

- **Throughput** is paramount
- Create, run, & retire **lots of threads** very rapidly
- Use **multithreading** to hide latency



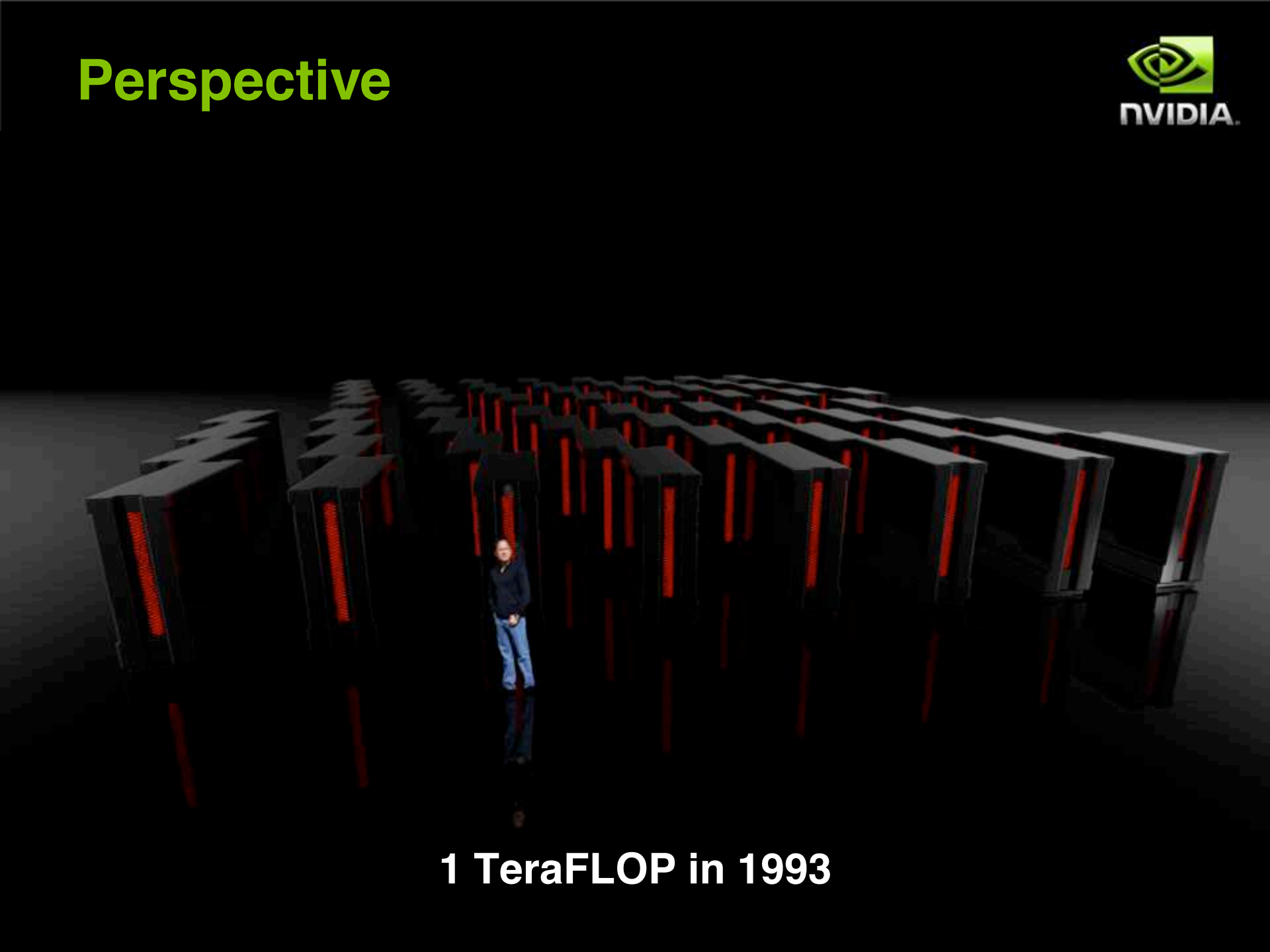
"Fermi"
3B xtors



1995 2000 2005 2010


Why
GPU Computing?
NVIDIA

Perspective



1 TeraFLOP in 1993

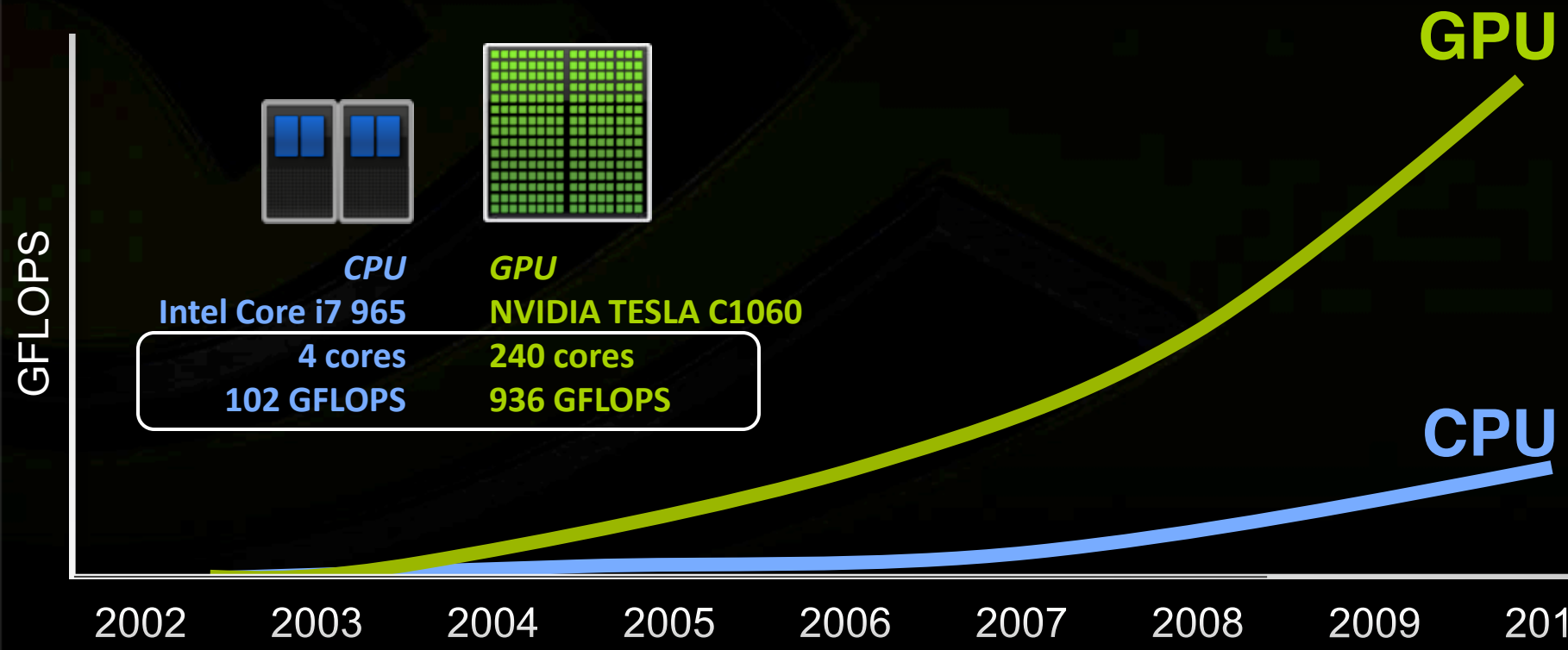
The "New" Moore's Law

- Computers no longer get faster, just wider
- You *must* re-think your algorithms to be parallel !
- Data-parallel computing is most scalable solution

Why GPU Computing?

Fact:
nobody cares about theoretical peak

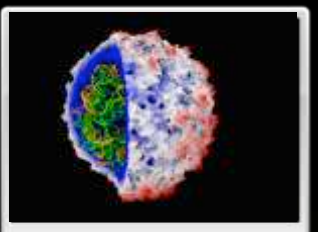
Challenge:
harness GPU power for real application performance



CUDA Successes

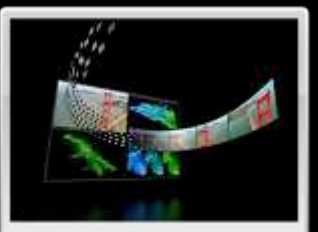
146X

Medical Imaging
U of Utah



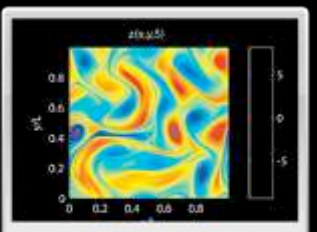
36X

Molecular Dynamics
U of Illinois



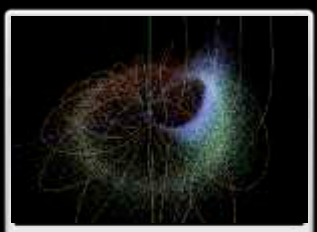
18X

Video Transcoding
Elemental Tech



50X

Matlab Computing
AccelerEyes

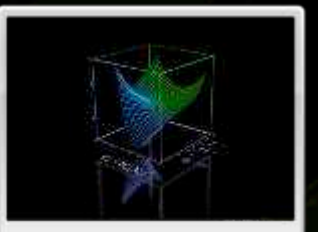


100X

Astrophysics
RIKEN

149X

Financial simulation
Oxford



47X

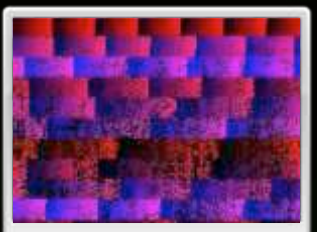
Linear Algebra
Universidad Jaime

20X

3D Ultrasound
Techniscan

130X

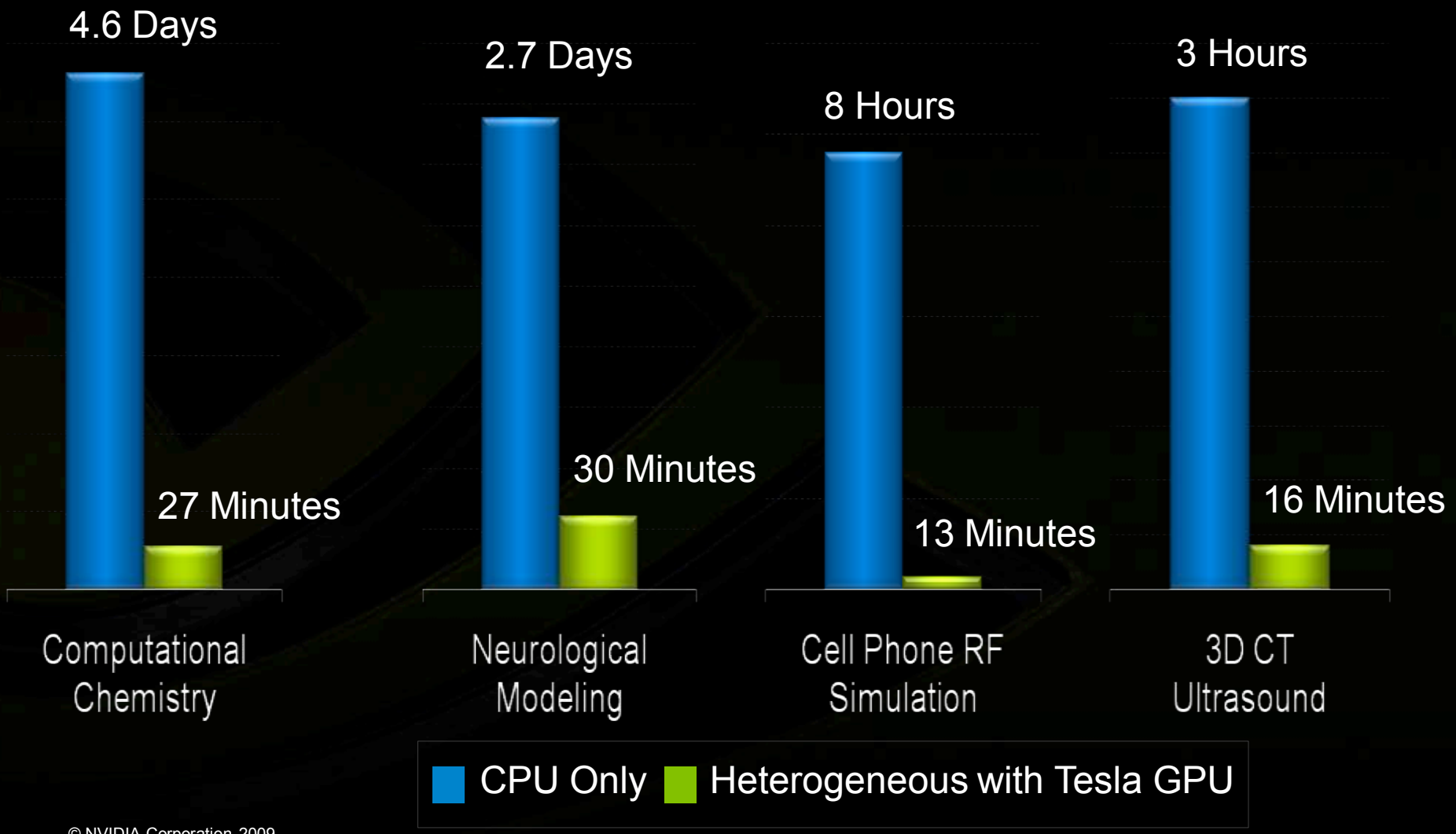
Quantum Chemistry
U of Illinois



30X

Gene Sequencing
U of Maryland

Accelerating Insight




GPU Computing
Epochs
NVIDIA

GPU Computing 1.0

(Ignoring prehistory: Ikonas, Pixel Machine, Pixel-Planes…)

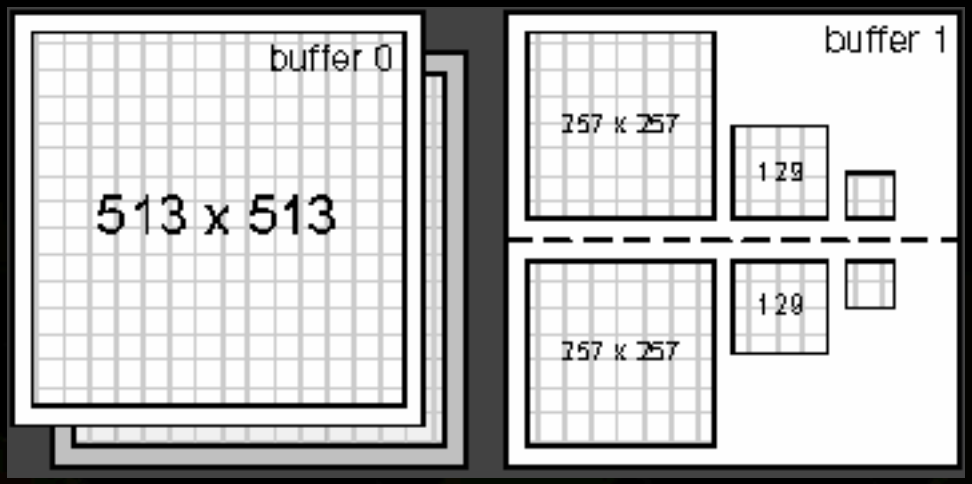
GPU Computing 1.0: *compute pretending to be graphics*

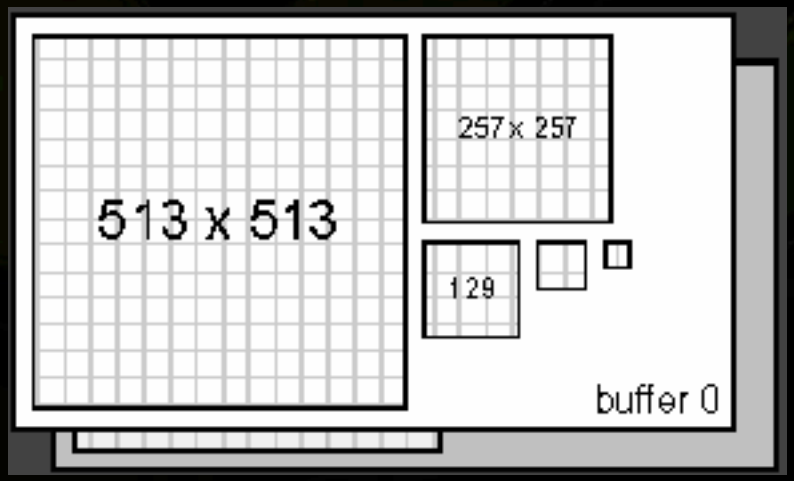
- Disguise data as textures or geometry
- Disguise algorithm as render passes

→Trick graphics pipeline into doing your computation!

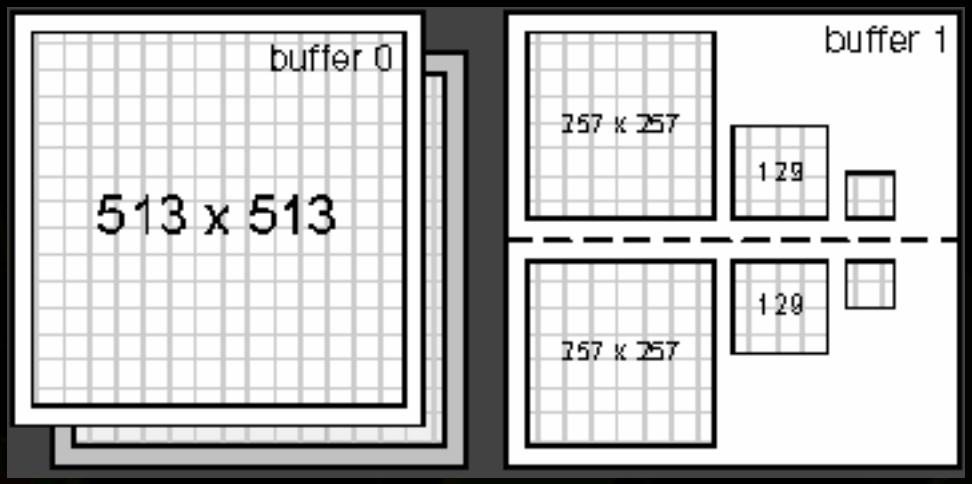
- Term *GPGPU* coined by Mark Harris

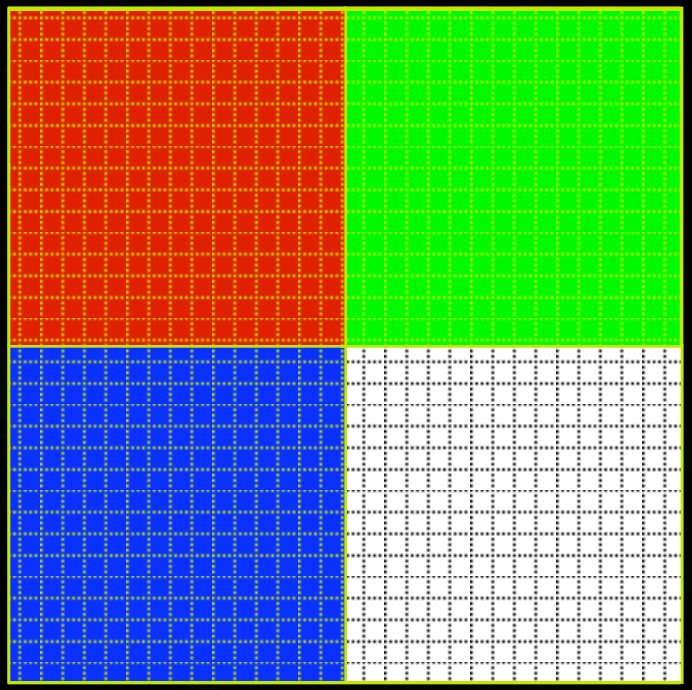
Typical GPGPU Constructs

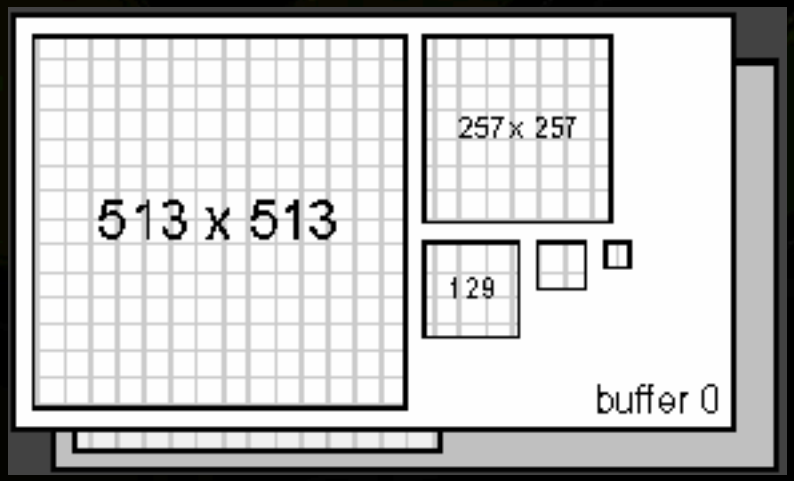




Typical GPGPU Constructs







GPGPU Hardware & Algorithms

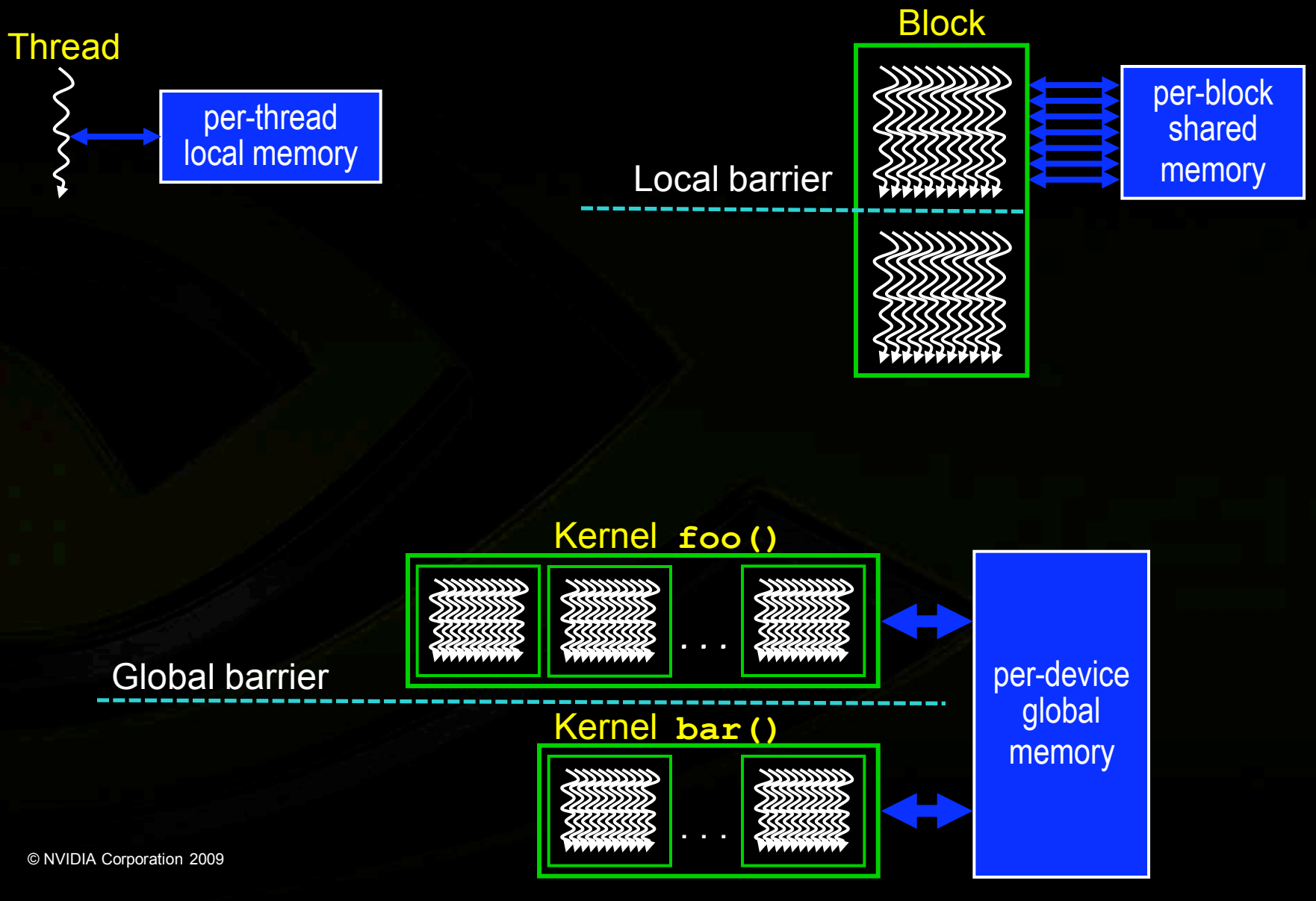
- GPUs get progressively more capable
  - Fixed-function → register combiners → shaders
  - fp32 pixel hardware greatly extends reach
- Algorithms get more sophisticated
  - Cellular automata → PDE solvers → ray tracing
  - Clever graphics tricks
- High-level shading languages emerge

GPU Computing 2.0 – Enter CUDA

GPU Computing 2.0: *direct compute*

- Program GPU directly, no graphics-based restrictions
- *GPU Computing* supplants graphics-based *GPGPU*
- November 2006: NVIDIA introduces *CUDA*

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

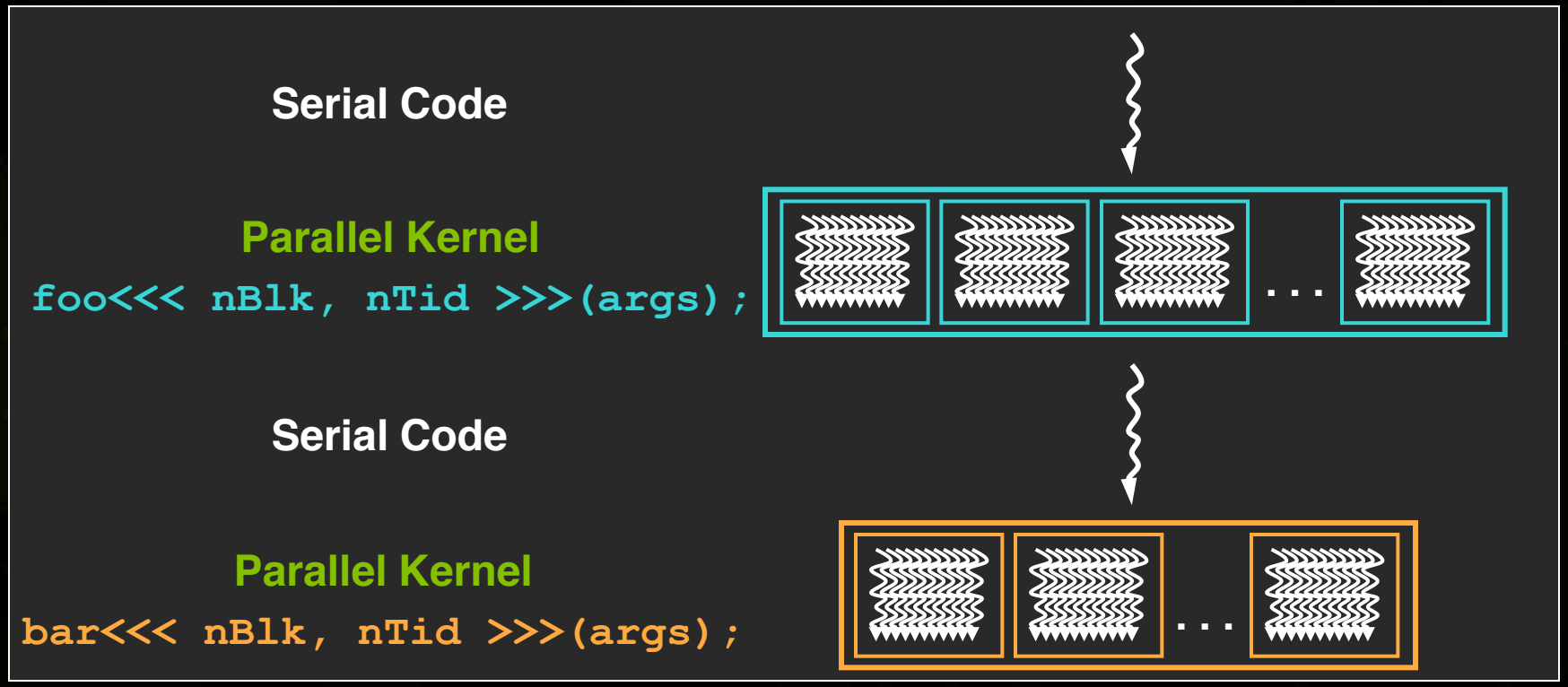
```
__global__ void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n)  y[i] = a*x[i] + y[i];
}

// 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



GPU Computing 3.0 – An Ecosystem

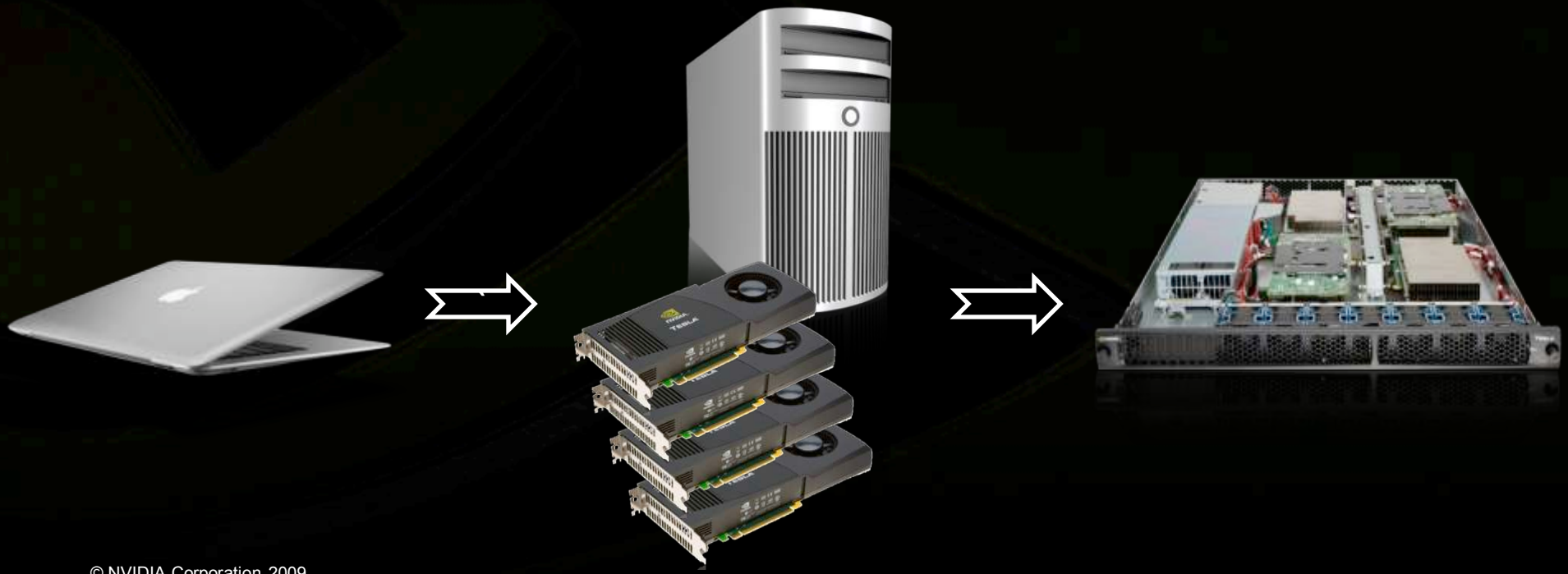
GPU Computing 3.0: ***an emerging ecosystem***

- Hardware & product lines
- Algorithmic sophistication
- Cross-platform standards
- Education & research
- Consumer applications
- High-level languages

GPU begins to "vanish" behind codes, platforms, apps

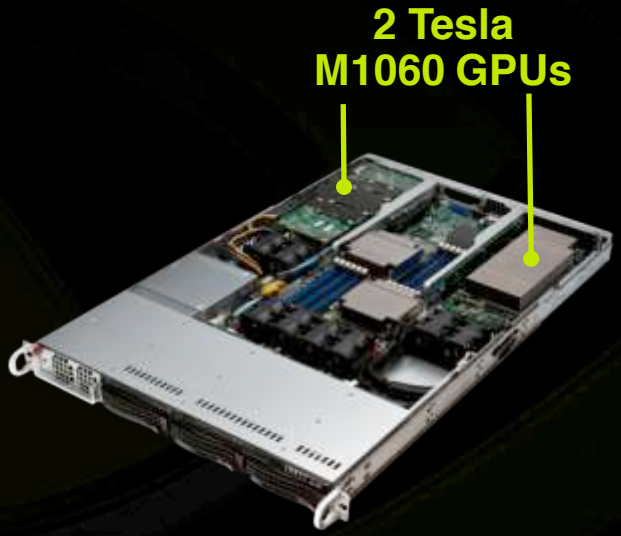
Products

- CUDA is in products from laptops to supercomputers

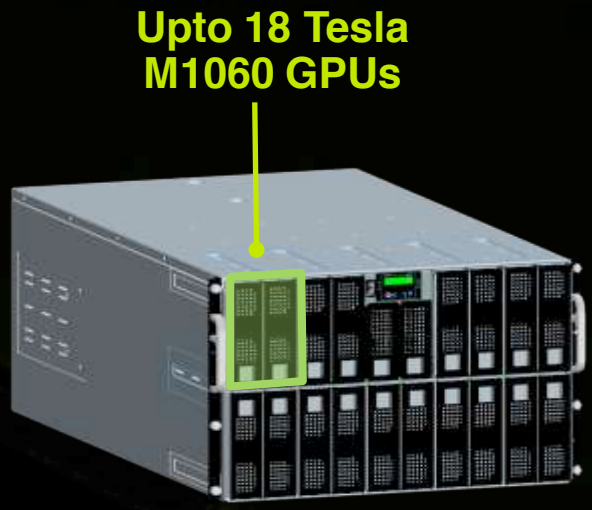


Emerging HPC Products

- New class of hybrid GPU-CPU servers



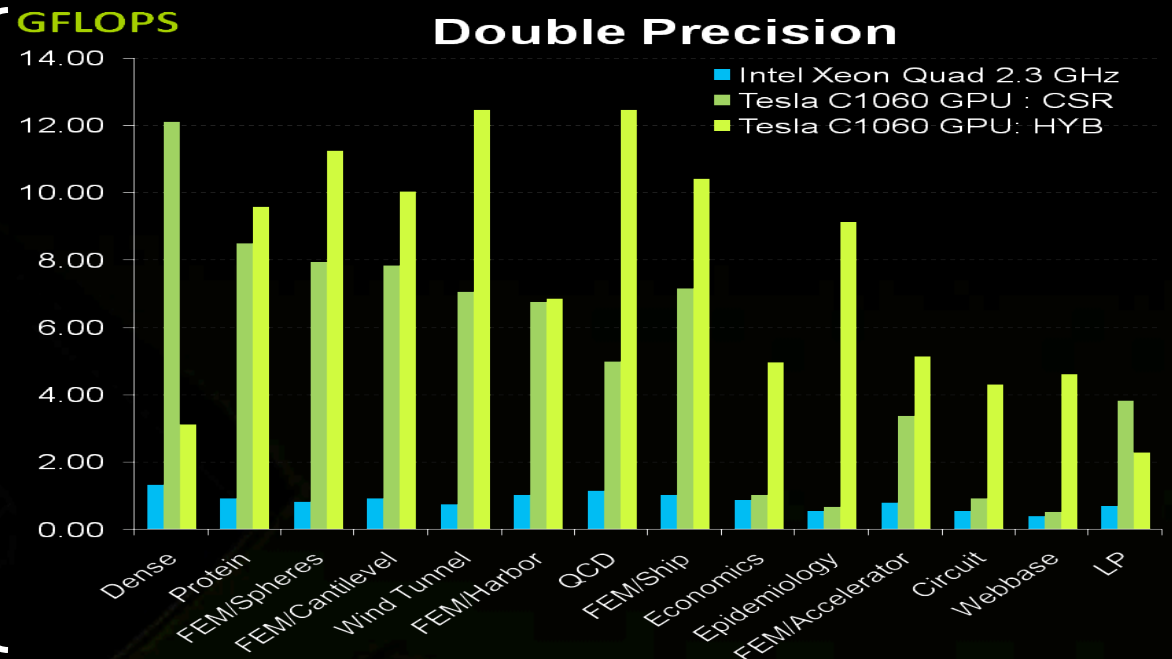
SuperMicro 1U
GPU Server



Bull Bullx
Blade Enclosure

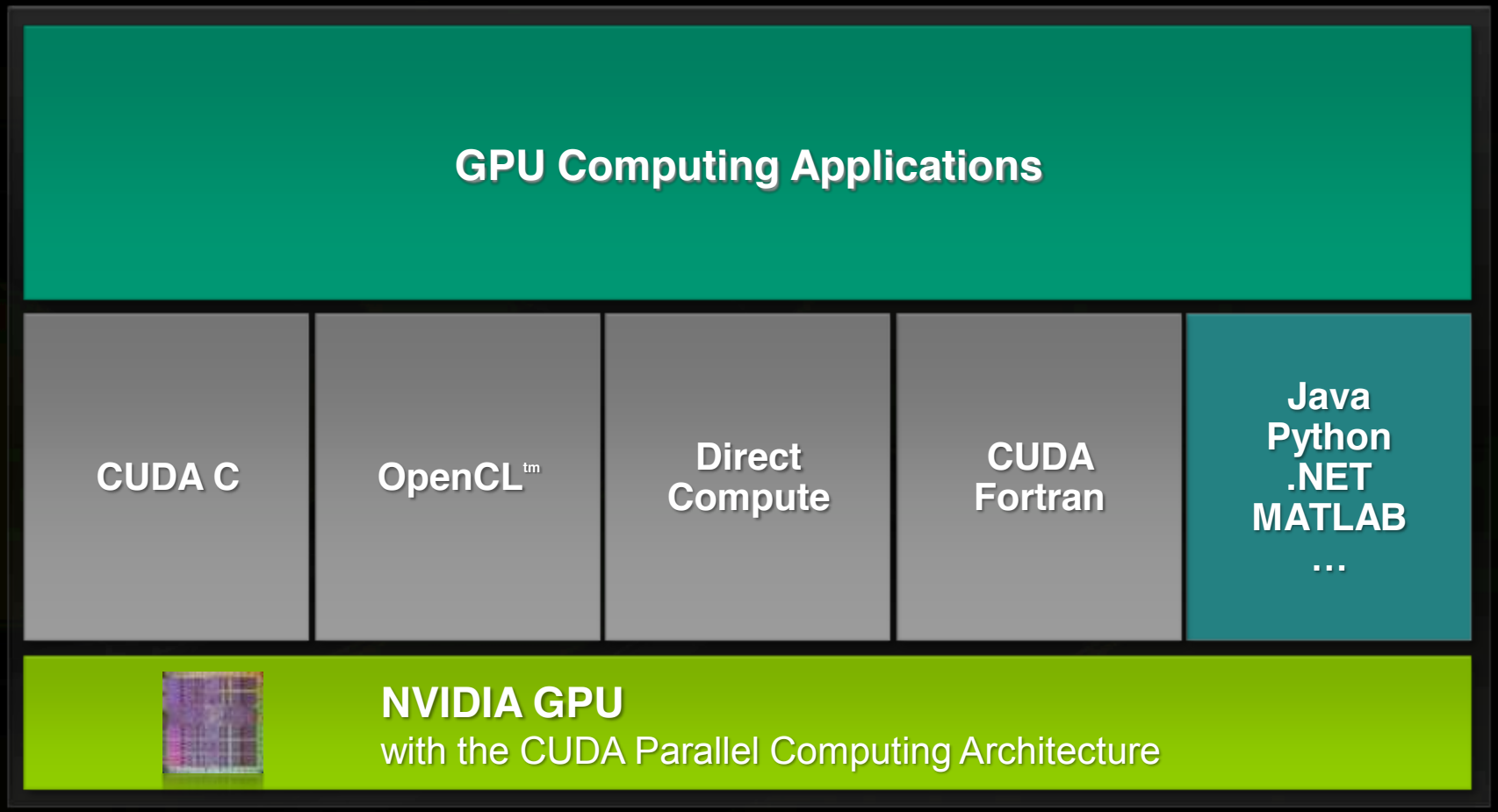
Algorithmic Sophistication

- Sort
- Sparse matrix
- Hash tables
- Fast multipole method
- Ray tracing (parallel tree traversal)



CPU Results from "Optimization of Sparse Matrix-Vector Multiplication on Emerging Multicore Platforms", Williams et al, Supercomputing 2007

Cross-Platform Standards



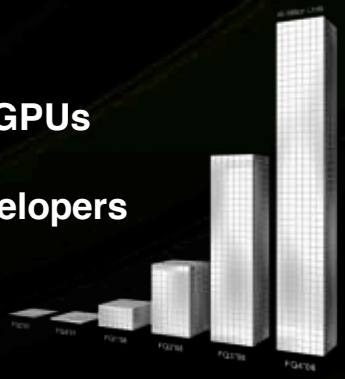
CUDA Momentum

Over 250 universities teach CUDA
Over 1000 research papers

CUDA powered TSUBAME
29th fastest supercomputer
in the world

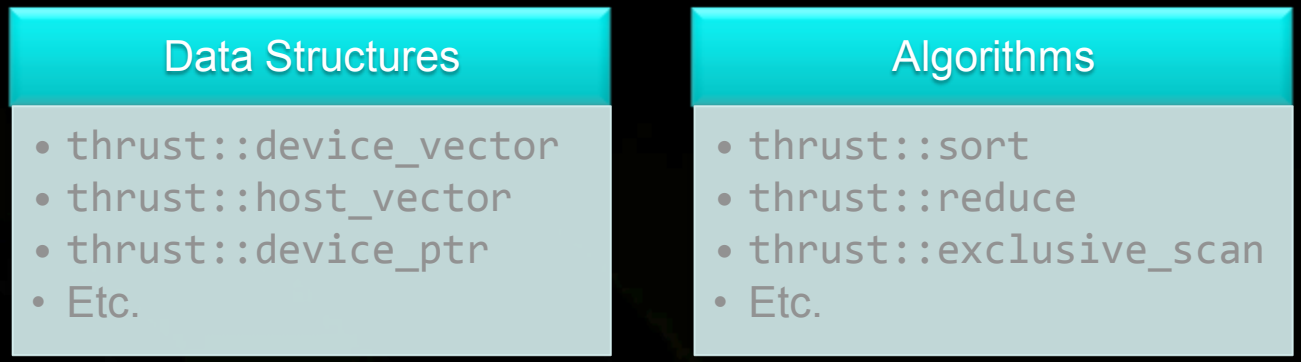
180 Million CUDA GPUs

100,000 active developers



Over 500 CUDA Apps
www.nvidia.com/CUDA

thrust



- **thrust: a library of data parallel algorithms & data structures with an interface similar to the C++ Standard Template Library for CUDA**
- **C++ template metaprogramming automatically chooses the fastest code path at compile time**

thrust::sort

sort.cu

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <cstdlib>

int main(void)
{
  // generate random data on the host
  thrust::host_vector<int> h_vec(1000000);
  thrust::generate(h_vec.begin(), h_vec.end(), rand);

  // transfer to device and sort
  thrust::device_vector<int> d_vec = h_vec;
  // sort 140M 32b keys/sec on GT200
  thrust::sort(d_vec.begin(), d_vec.end());

  return 0;
}
```

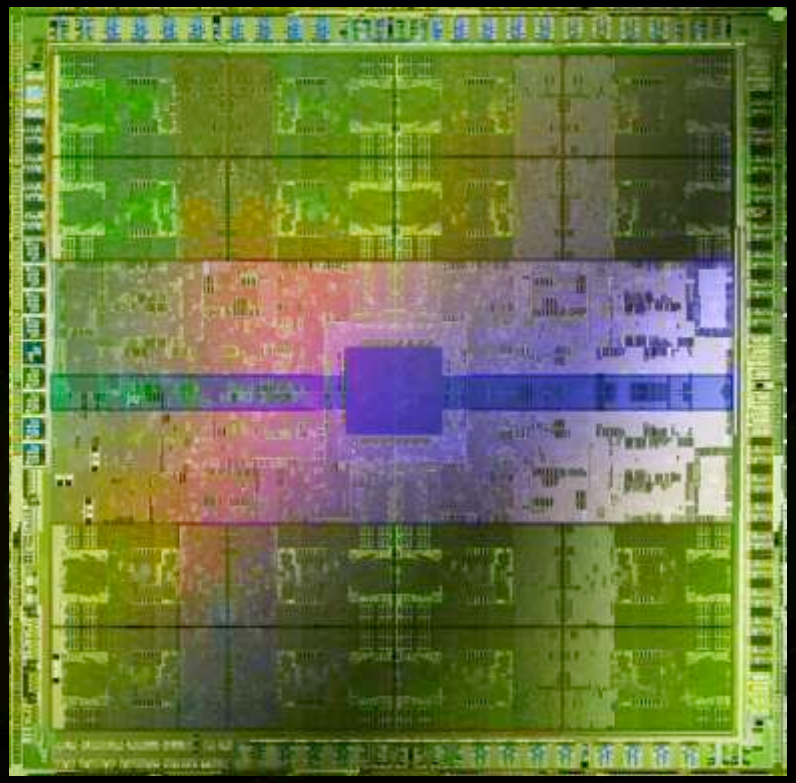
http://thrust.googlecode.com

GPU Begins to Vanish

- Ever-increasing number of codes & commercial packages "just go faster" when GPU is present
  - Some bio/chem codes available or porting:
    - NAMD / VMD, GROMACS (alpha), HOOMD, GPU HMMER, MUMmerGPU, AutoDock…
    - LAMMPS, CHARMM, Q-Chem, Gaussian, AMBER…
  - Consumer applications, e.g. Photoshop
    - Badaboom, vReveal, Nero MoveIt
  - OS: Microsoft *Windows 7*, MacOS *Snowleapord*


Fermi
NVIDIA

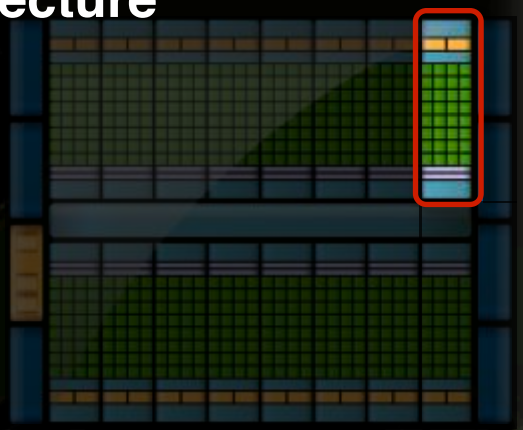
Next-Gen GPU Architecture: *Fermi*

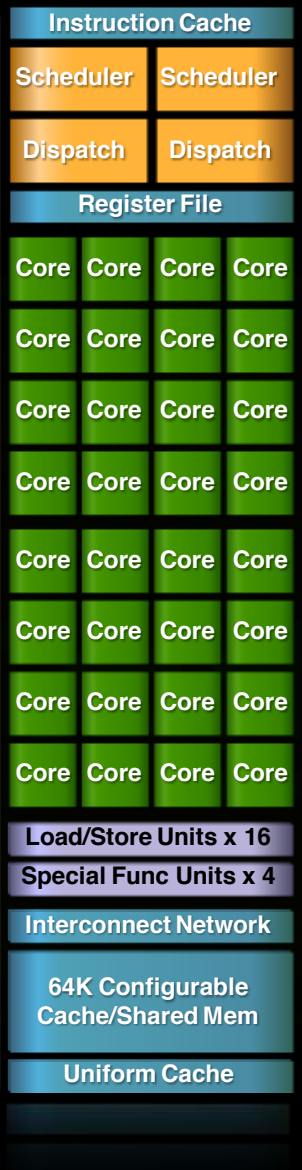


- 3 billion transistors
- Over 2x the cores (512 total)
- ~2x the memory bandwidth
- L1 and L2 caches
- 8x the peak DP performance
- ECC
- C++

SM Microarchitecture

- Objective – optimize for GPU computing
  - New ISA
  - Revamp issue / control flow
  - New CUDA core architecture
- 32 cores per SM (512 total)
- 64KB of configurable L1$ / shared memory

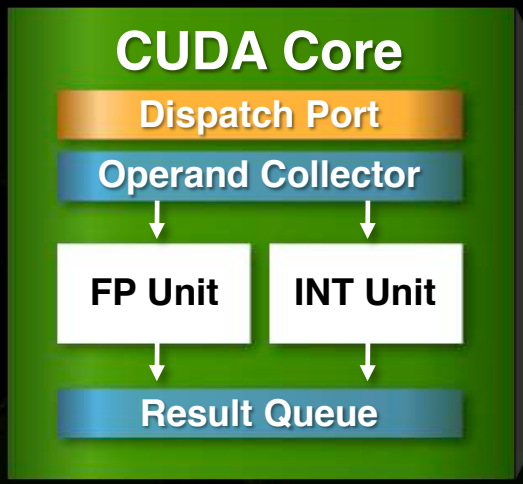


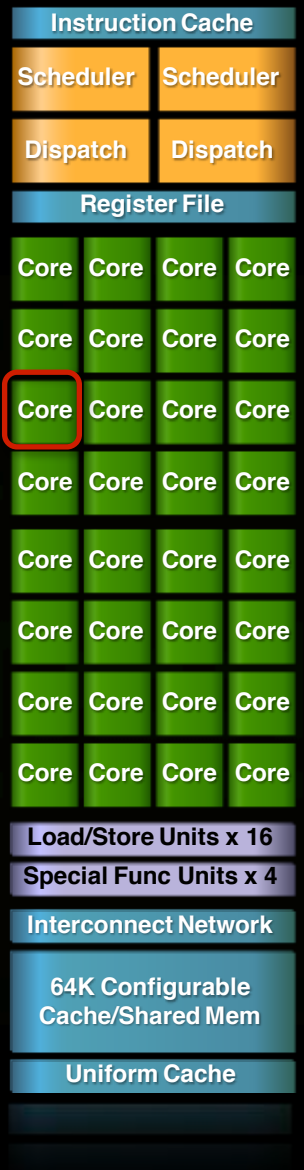


	FP32	FP64	INT	SFU	LD/ST
Ops / clk	32	16	32	4	16

SM Microarchitecture

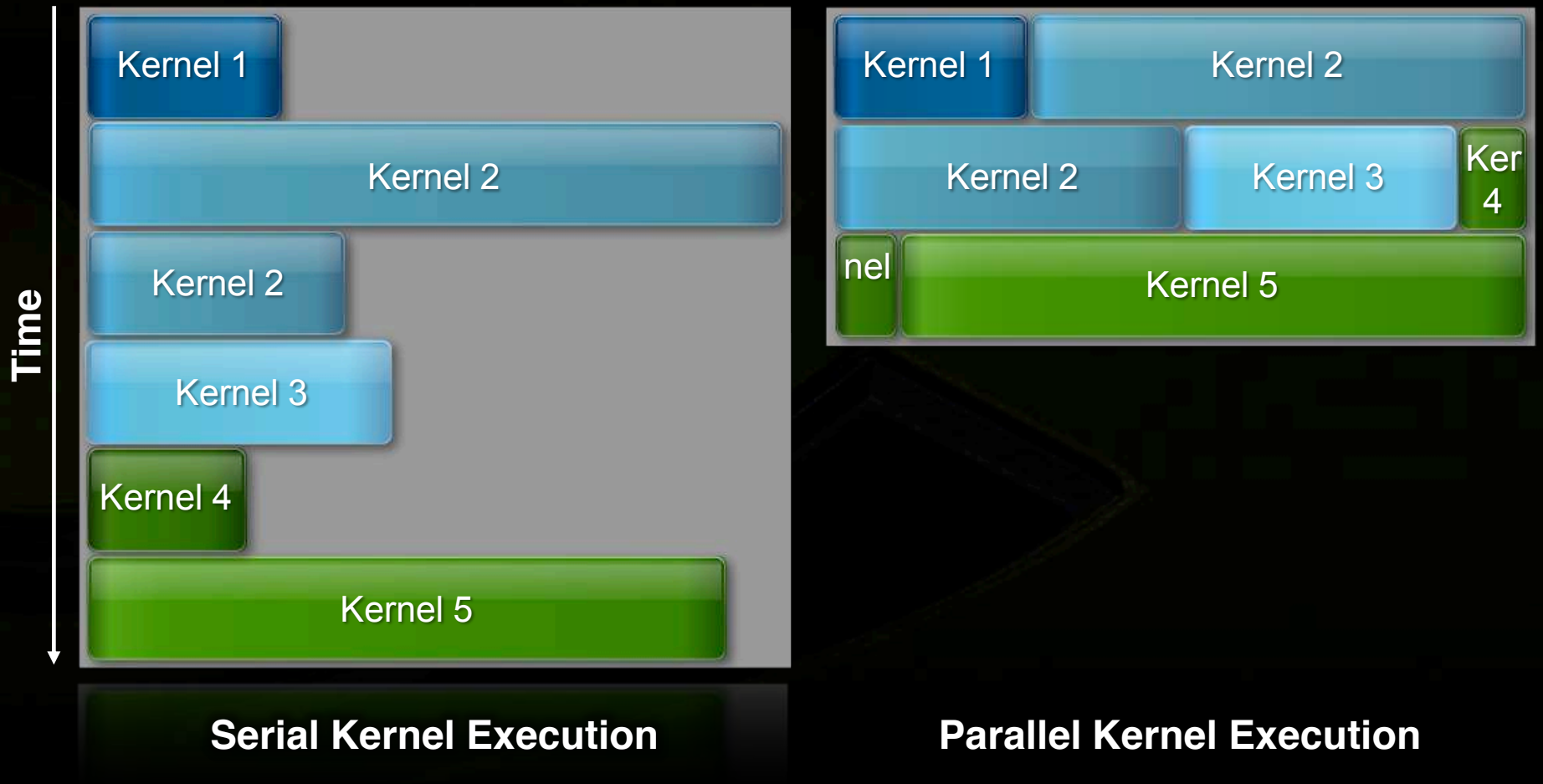
- New IEEE 754-2008 arithmetic standard
- Fused Multiply-Add (FMA) for SP & DP
- New integer ALU optimized for 64-bit and extended precision ops





Hardware Thread Scheduling

Concurrent kernel execution + faster context switch



More Fermi Goodness

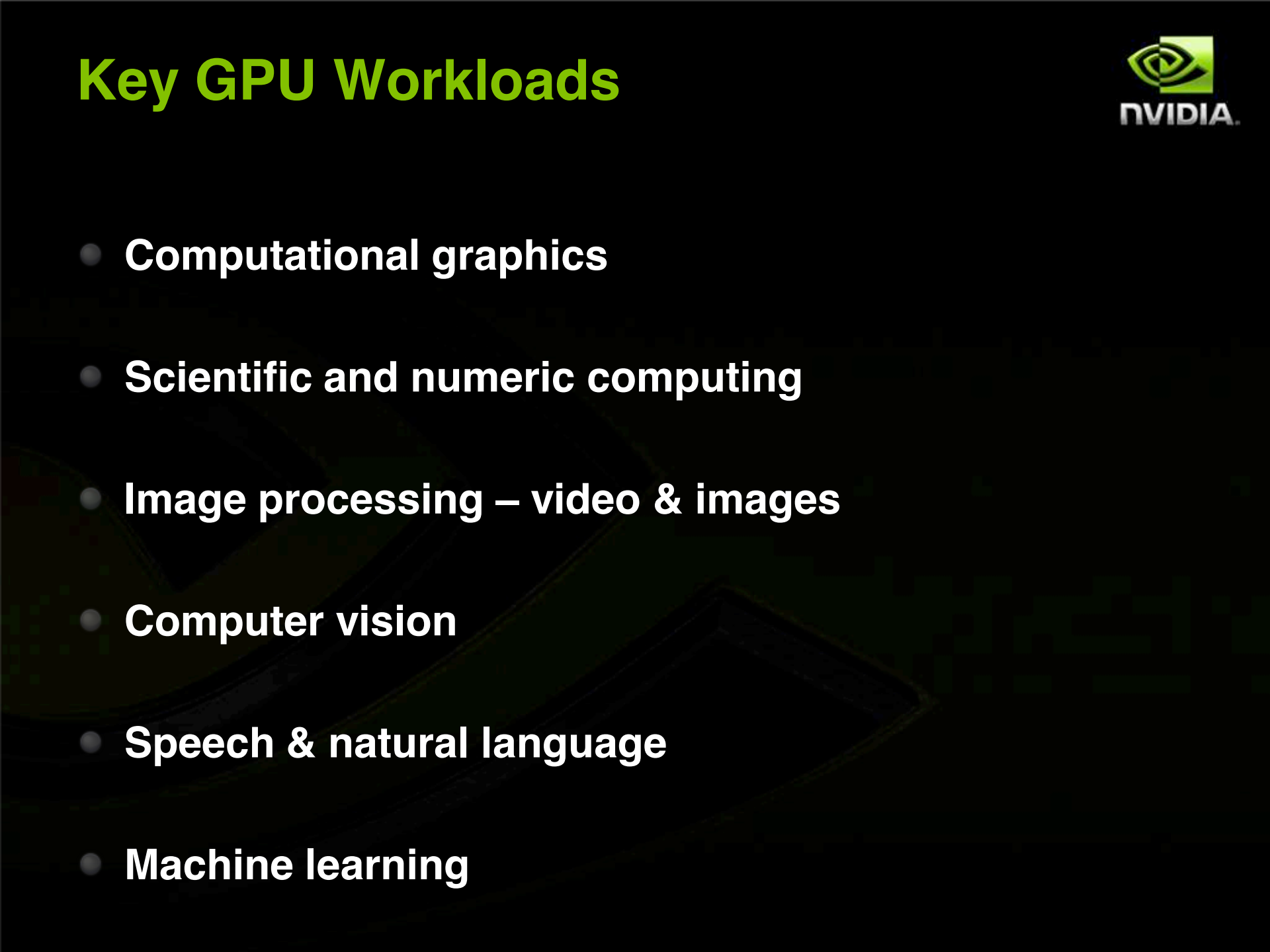
- ECC protection for DRAM, L2, L1, RF
- Unified 40-bit address space for local, shared, global
- 5-20x faster atomics
- Dual DMA engines for CPU←→ GPU transfers
- ISA extensions for C++ (e.g. virtual functions)


GPU Computing 4.0?
NVIDIA

Key CUDA Challenges

- Express other programming models elegantly
  - Producer-consumer: persistent thread blocks reading and writing work queues
  - Task-parallel: kernel, thread block or warp as parallel task
  - Both common “power-user” CUDA patterns
- Foster more high-level languages & platforms
- Improve & mature development environment

Key GPU Workloads



- Computational graphics
- Scientific and numeric computing
- Image processing – video & images
- Computer vision
- Speech & natural language
- Machine learning

The Future of Computing?

- **Forward-looking statements:**

All future interesting problems are *throughput* problems.

GPUs will evolve to be *the* general-purpose throughput processors.

CPUs as we know them will become (already are?) "good enough", and shrink to a corner of the die.

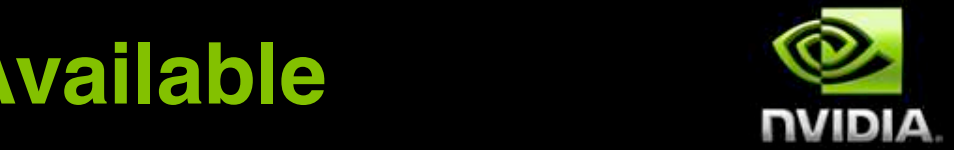
Final Thoughts – Education

- We should teach parallel computing in CS 1 or CS 2
  - Computers don't get faster, just wider
  - Manycore is the ~~future~~ now of computing
- Heapsort and mergesort
  - Both O(n lg n)
  - One parallel-friendly, one not
  - Students need to understand this early


Questions?
dluebke@nvidia.com
NVIDIA

Miscellaneous Thoughts

NVIDIA Resources Available



NVIDIA enables heterogeneous computing research and teaching:

- **Grants for leading researchers in GPU Computing**
  - **Ph.D. Fellowship program**
  - **Professor Partnership program**
- **Discounted hardware**
- **GPU Computing Ventures – Venture fund focused on GPU computing**
- **Technical Training and Assistance**