NVIDIA GPU Architecture for General Purpose Computing

Anthony Lippert
4/27/09

Outline

- Introduction
- GPU Hardware
- Programming Model
- Performance Results
- Supercomputing Products
- Conclusion

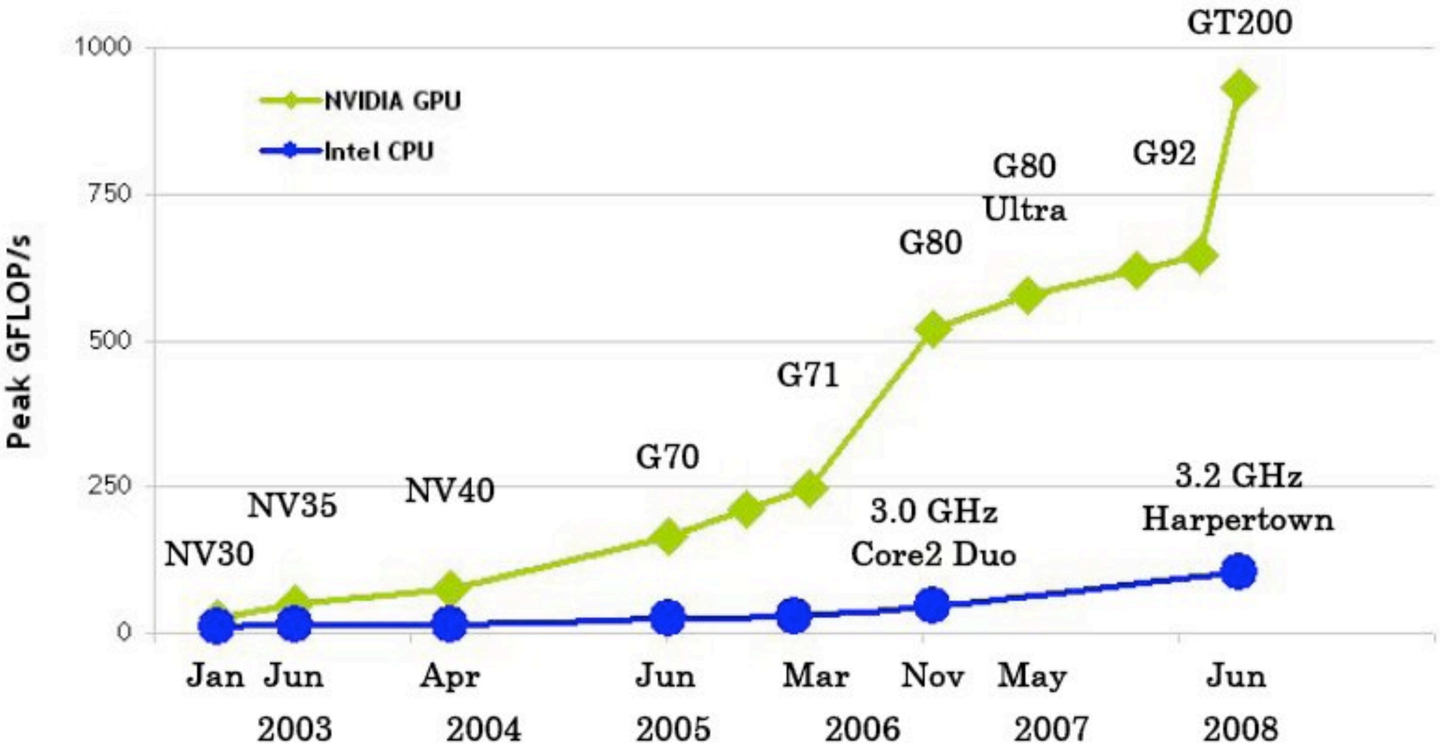
Intodution

GPU: Graphics Processing Unit

- Hundreds of Cores
- Programmable
- Can be easily installed in most desktops
- Similar price to CPU
- GPU follows Moore's Law better than CPU

Introduction

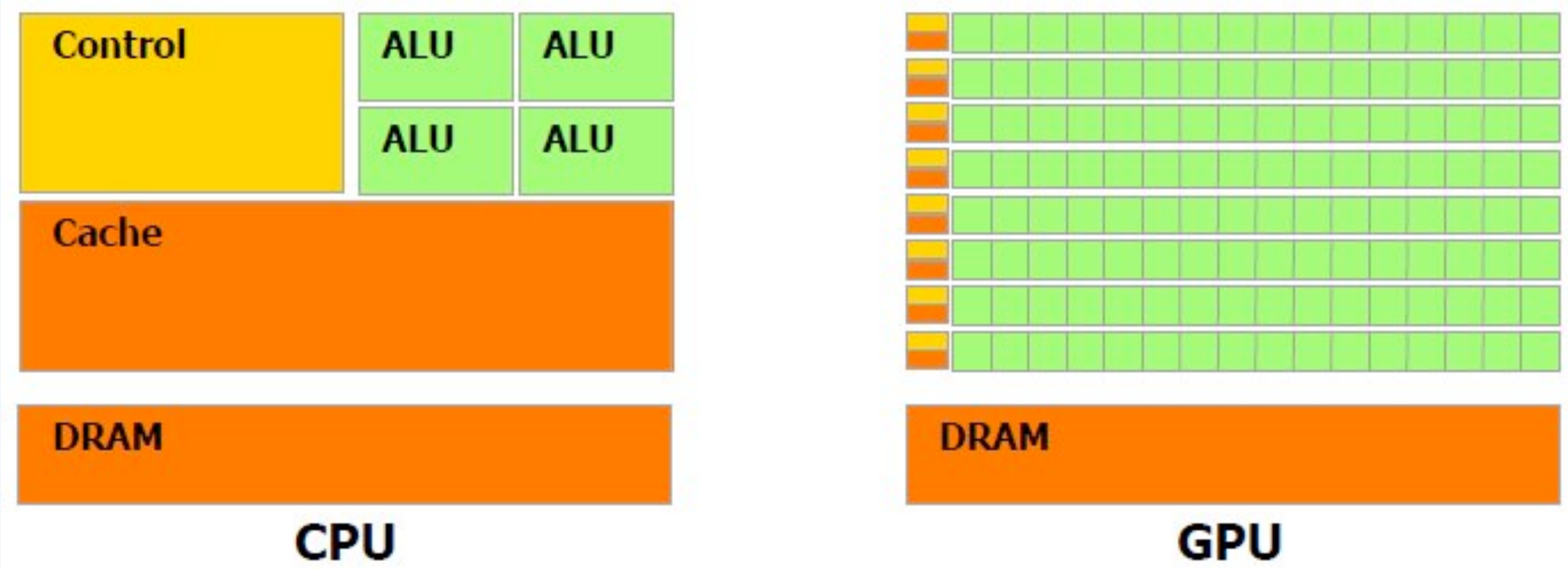
Motivation:



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

GPU Hardware

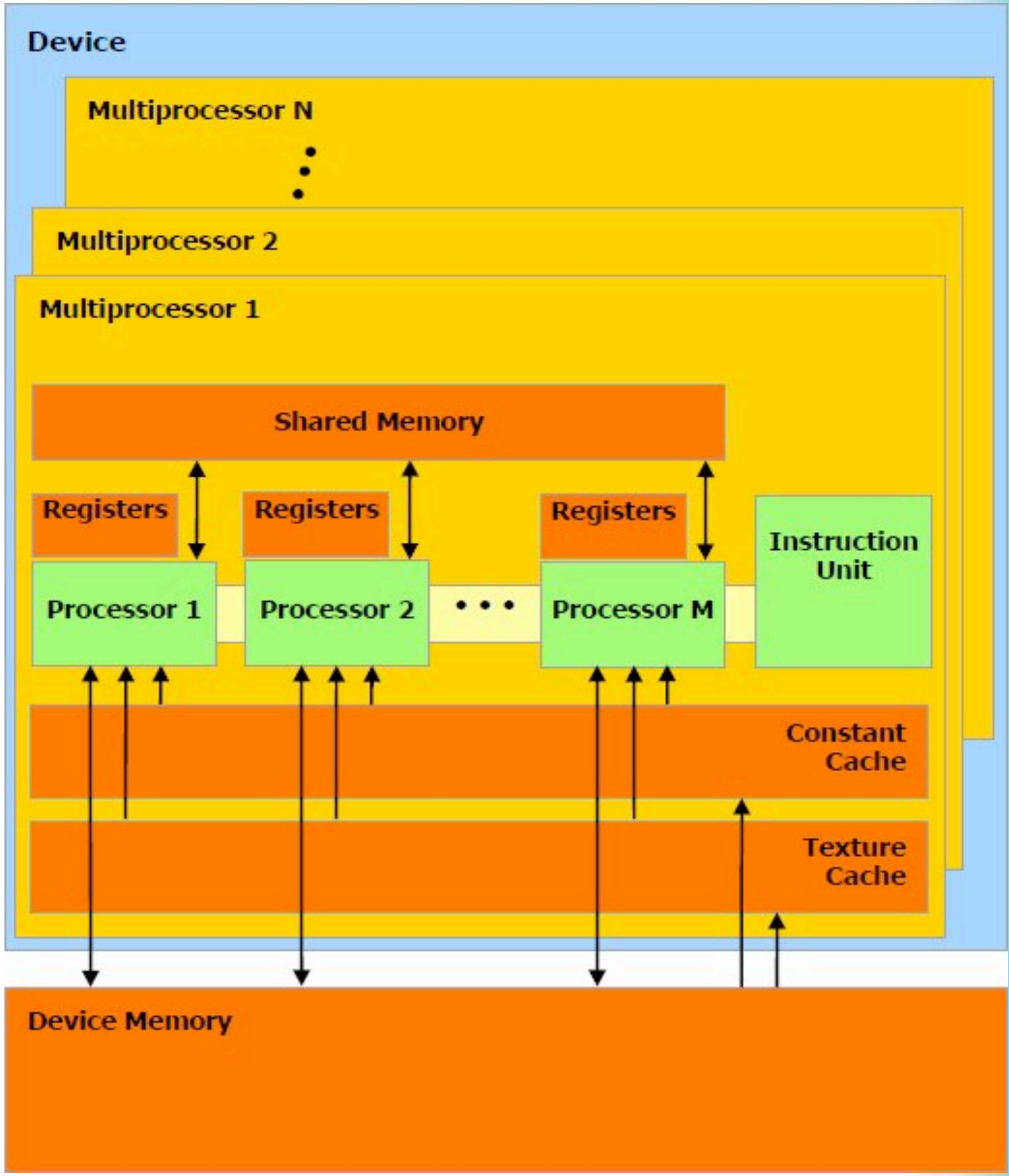
<u>Multiprocessor Structure:</u>



GPU Hardware

<u>Multiprocessor Structure:</u>

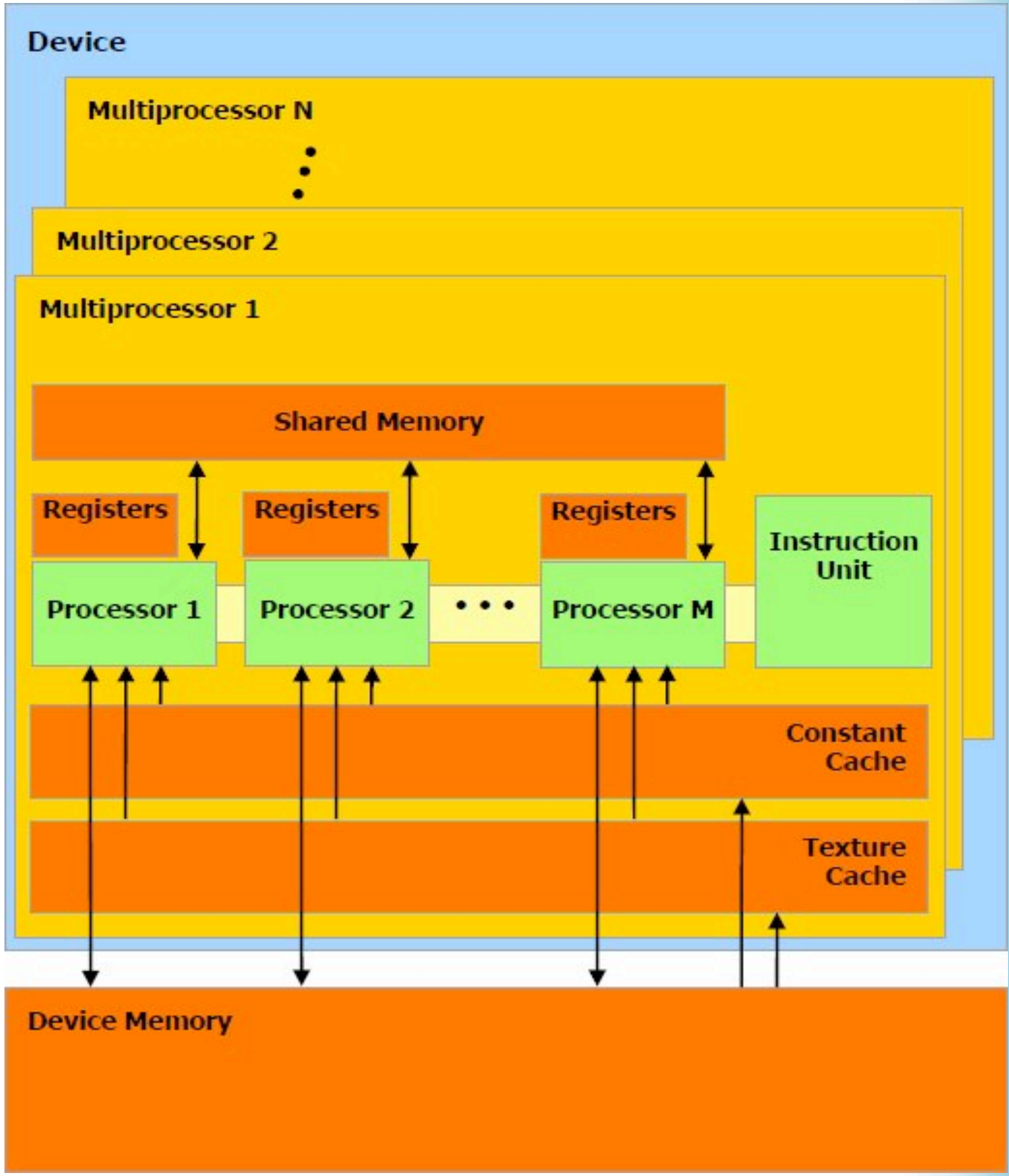
- N multiprocessors with M cores each
- SIMD – Cores share an Instruction Unit with other cores in a multiprocessor.
- Diverging threads may not execute in parallel.



GPU Hardware

Memory Hierarchy:

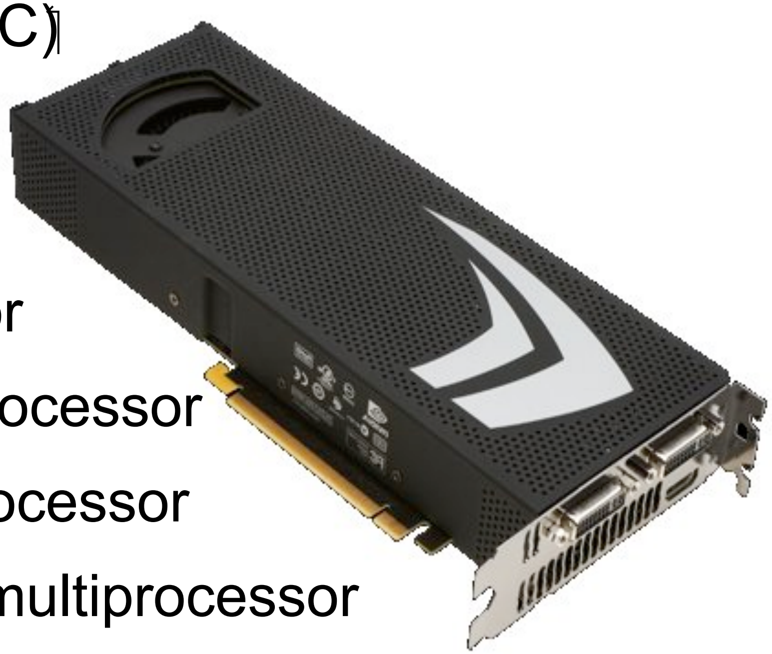
- Processors have 32-bit registers
- Multiprocessors have shared memory, constant cache, and texture cache
- Constant/texture cache are read-only and have faster access than shared memory.



GPU Hardware

NVIDIA GTX280 Specifications:

- 933 GFLOPS peak performance
- 10 thread processing clusters (TPC)
- 3 multiprocessors per TPC
- 8 cores per multiprocessor
- 16384 registers per multiprocessor
- 16 KB shared memory per multiprocessor
- 64 KB constant cache per multiprocessor
- 6 KB < texture cache < 8 KB per multiprocessor
- 1.3 GHz clock rate
- Single and double-precision floating-point calculation
- 1 GB DDR3 dedicated memory



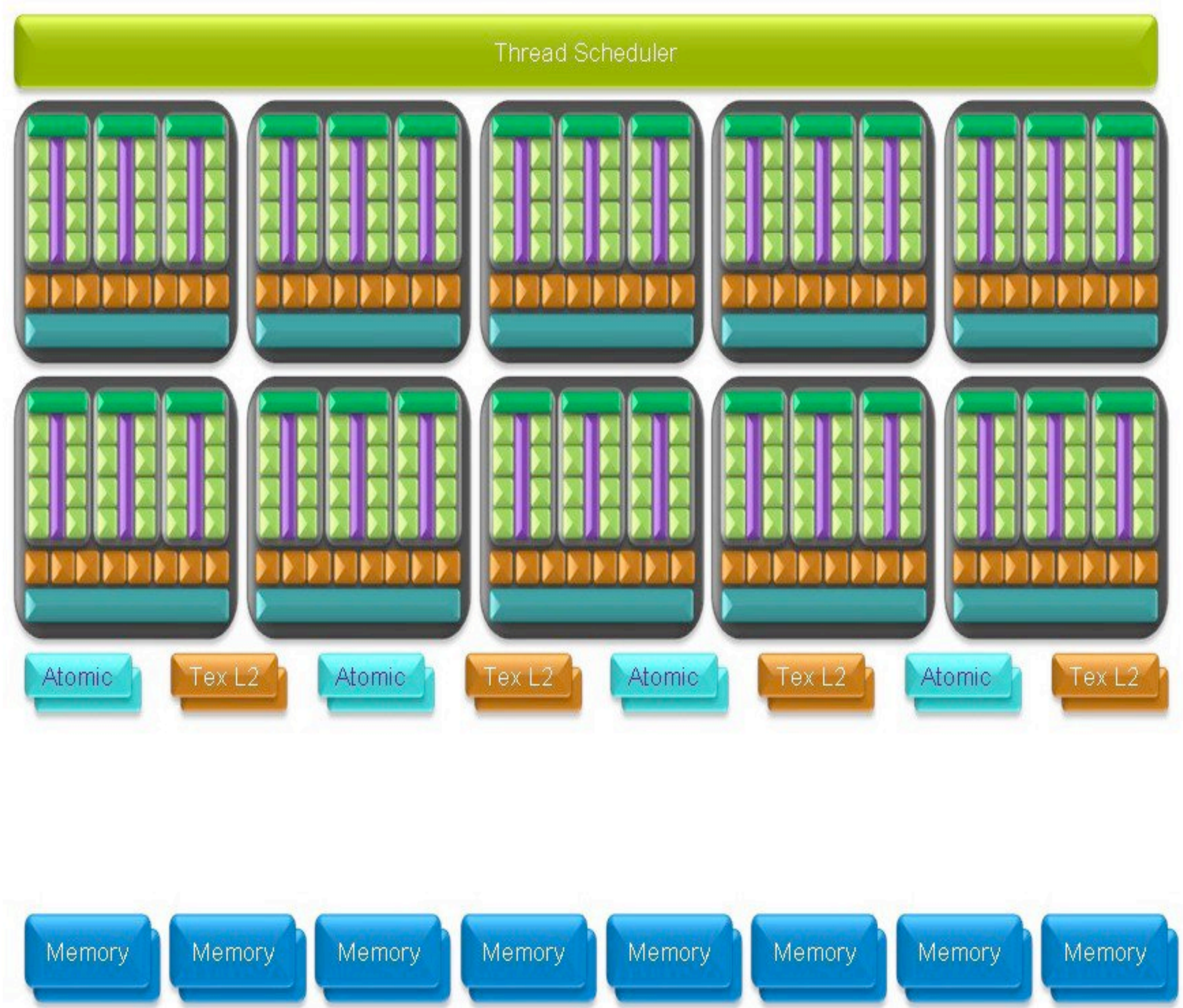
GPU Hardware

- Thread Scheduler
- Thread Processing

Clusters

- Atomic/Tex L2
- Memory

GeForce GTX 280 Parallel Computing Architecture



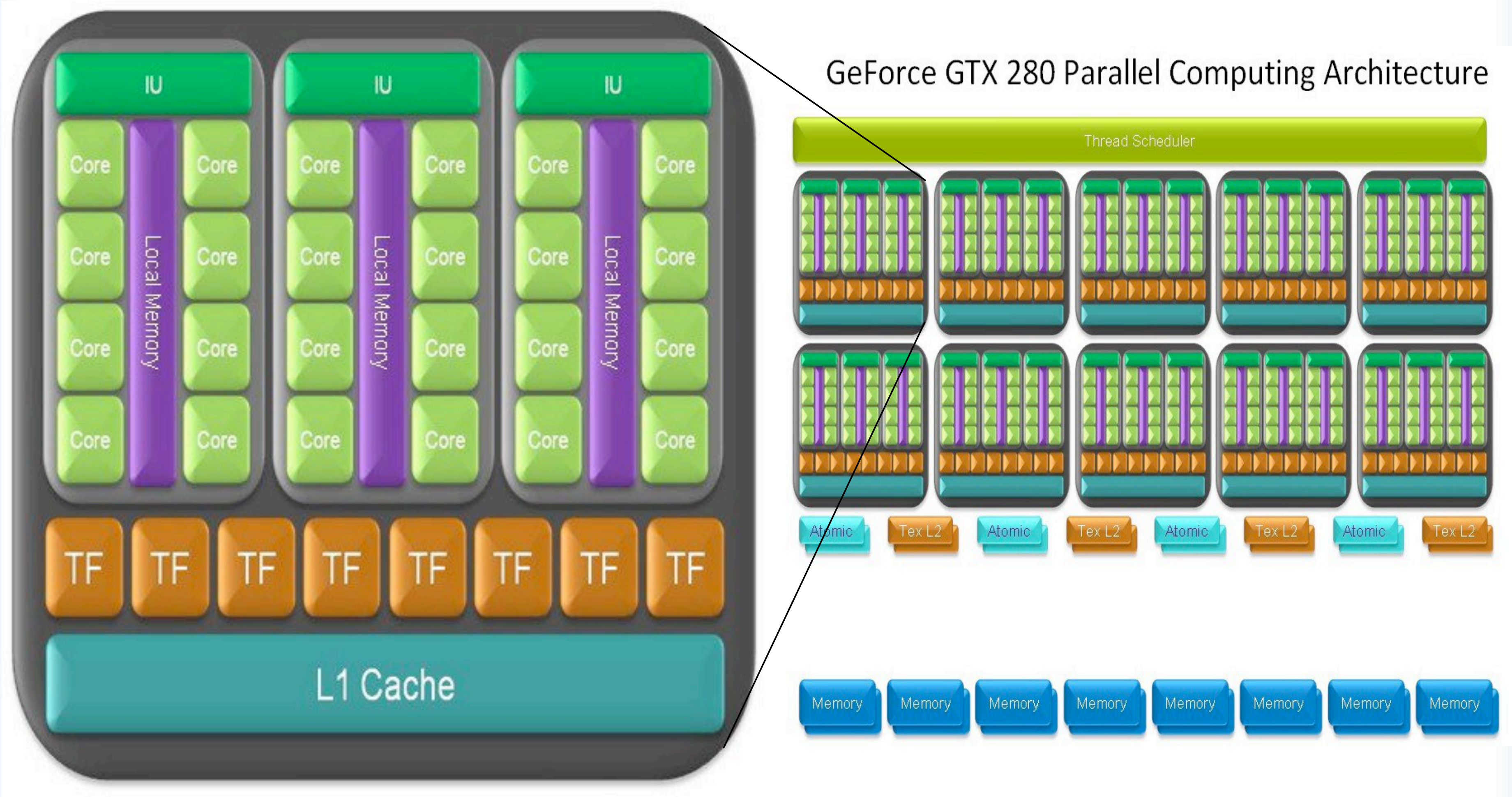
GPU Hardware

Thread Scheduler:

- Hardware-based
- Manages scheduling threads across thread processing clusters
- Nearly 100% utilization: If a thread is waiting for memory access, the scheduler can perform a zero-cost, immediate context switch to another thread
- Up to 30,720 threads on the chip

GPU Hardware

Thread Processing Cluster:



IU - instruction unit TF - texture filtering

GPU Hardware

Atomic/Tex L2:

- Level 2 Cache
- Shared by all thread processing clusters
- Atomic
  - Ability to perform read-modify-write operations to memory
  - Allows granular access to memory locations
  - Provides parallel reductions and parallel data structure management

GPU Hardware

Features	8800 GTX	GTX 280	% Increase
Cores	128	240	87.5 %
TEX	64t/clk	80t/clk	25 %
ROP Blend	12p/clk	32p/clk	167 %
Precision	fp32	fp64	--
GFLOPs	518	933	80 %
FB Bandwidth	86 GB	142 GB	65 %
Texture Fill	37 GT/s	48 GT/s	29.7 %
ROP Blend	7 GBL/s	19 GBL/s	171 %
PCI Express	6.4 GB	12.8 GB	100 %
Video	VP1	VP2	--

GPU Hardware

GT200 Power Features:

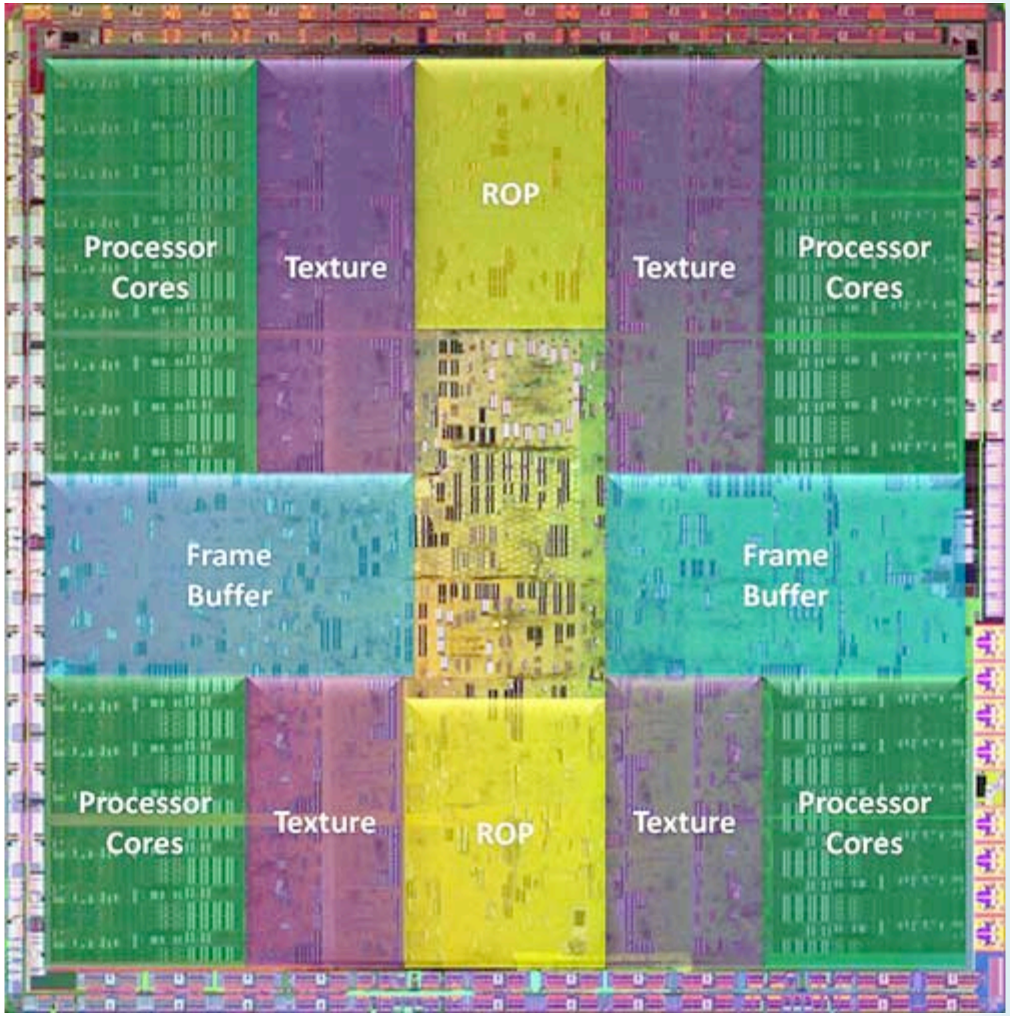
- Dynamic power management
- Power consumption is based on utilization
  - Idle/2D power mode: 25 W
  - Blu-ray DVD playback mode: 35 W
  - Full 3D performance mode: worst case 236 W
  - HybridPower mode: 0 W
    - On an nForce motherboard, when not performing, the GPU can be powered off and computation can be diverted to the motherboard GPU (mGPU)

GPU Hardware

- 10 Thread Processing Clusters(TPC)
- 3 multiprocessors per TPC
- 8 cores per multiprocessor
- ROP – raster operation

  processors (for graphics)
- 1024 MB frame buffer for

  displaying images
- Texture (L2) Cache



Programming Model

Past:

- The GPU was intended for graphics only, not general purpose computing.
- The programmer needed to rewrite the program in a graphics language, such as OpenGL
- Complicated

Present:

- NVIDIA developed CUDA, a language for general purpose GPU computing
- Simple

Programming Model

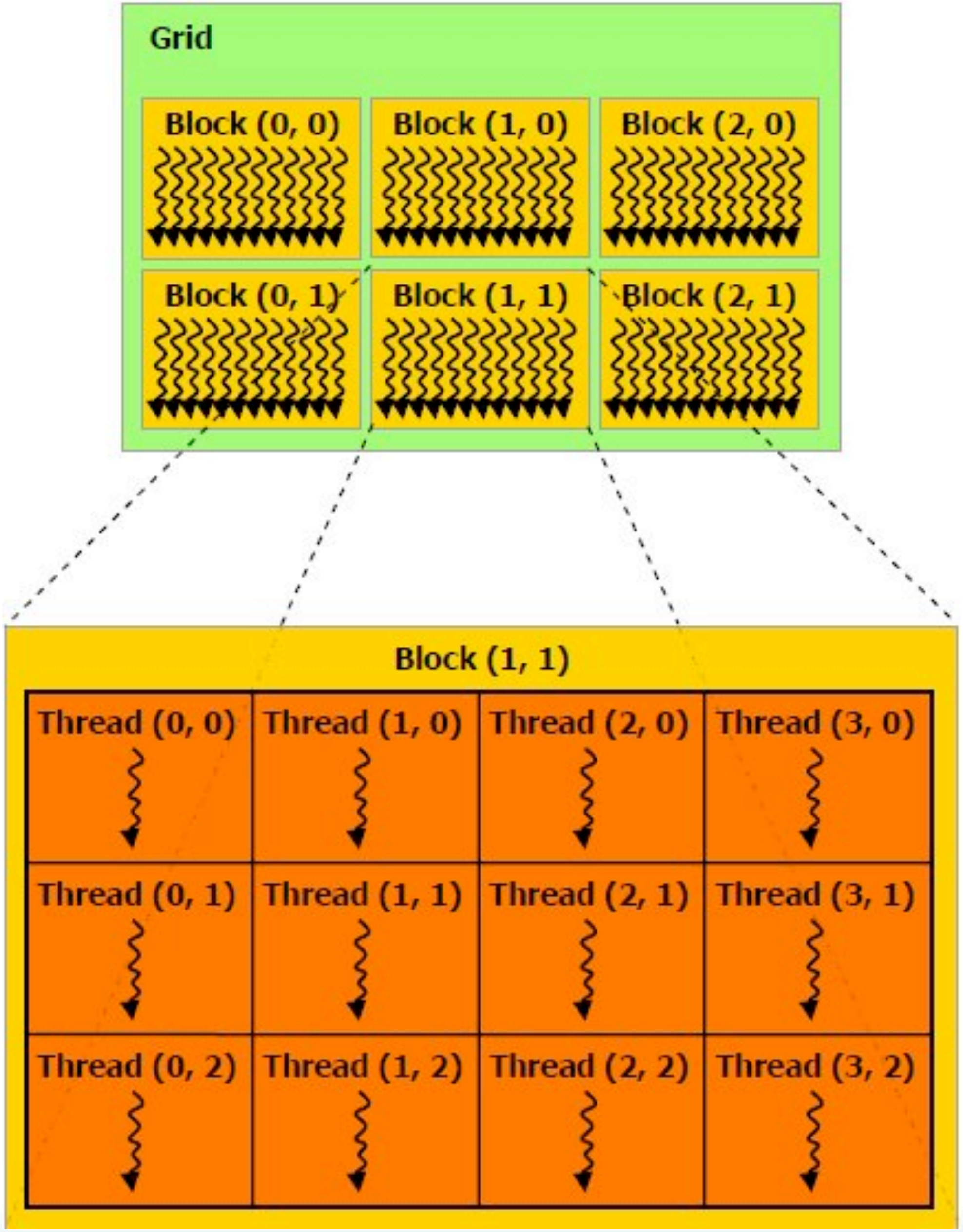
CUDA:

- Compute Unified Device Architecture
- Extension of the C language
- Used to control the device
- The programmer specifies CPU and GPU functions
  - The host code can be C++
  - Device code may only be C
- The programmer specifies thread layout

Programming Model

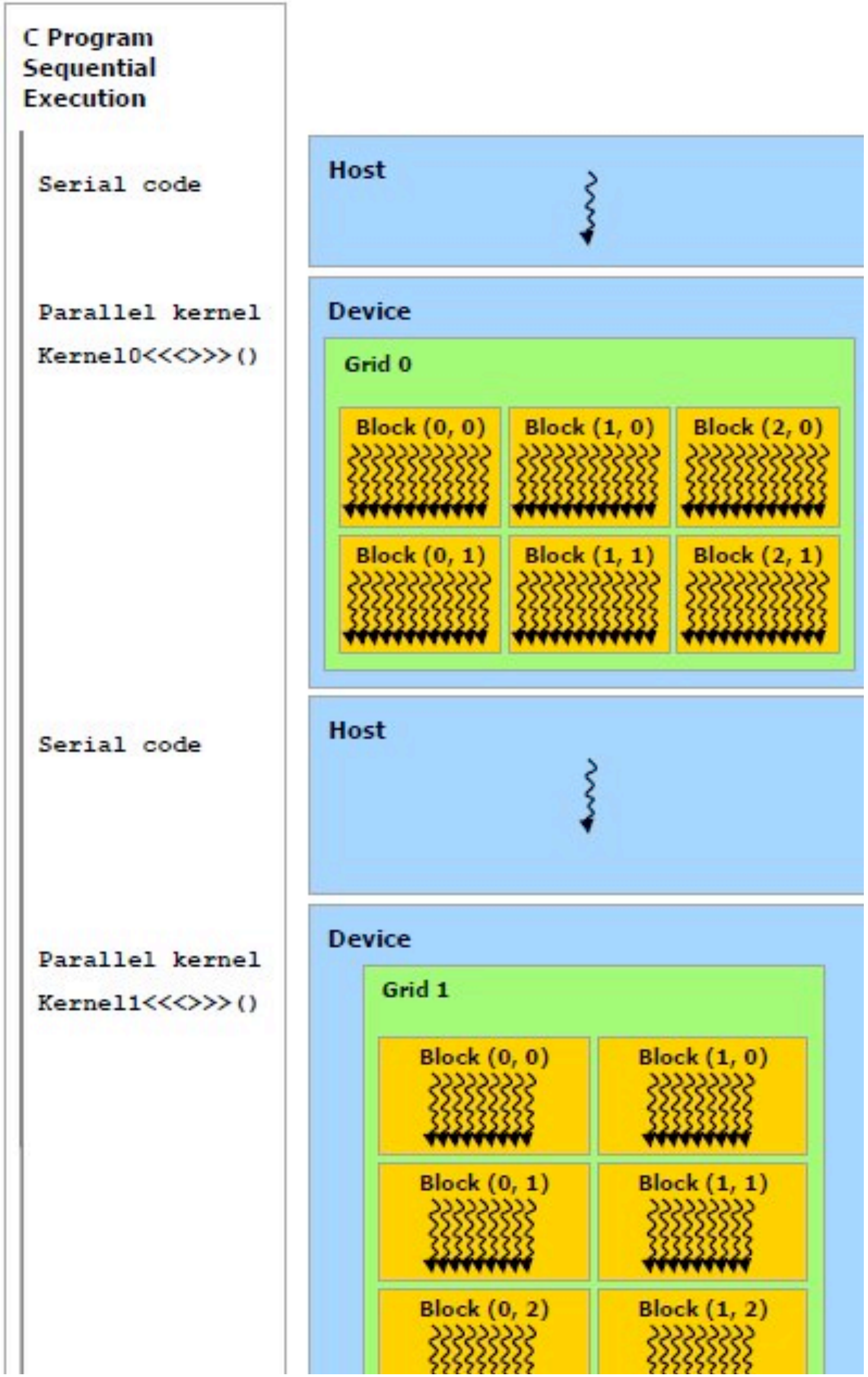
Thread Layout:

- Threads are organized into *blocks*.
- Blocks are organized into a *grid*.
- A multiprocessor executes one block at a time.
- A *warp* is the set of threads executed in parallel
- 32 threads in a warp



Programming Model

- Heterogeneous Computing:
  - GPU and CPU execute different types of code.
  - CPU runs the main program, sending tasks to the GPU in the form of kernel functions
  - Multiple kernel functions may be declared and called.
  - Only one kernel may be called at a time.



Programming Model: GPU vs. CPU Code

CPU C program

```c
void add_matrix_cpu
                (float *a, float *b, float *c, int N)
{
    int i, j, index;
  for (i=0;i<N;i++) {
   for (j=0;j<N;j++) {
      index =i+j*N;
      c[index]=a[index]+b[index];
      }
  }
}
void main()
{
  .....
     add_matrix(a,b,c,N);
}
```

CUDA C program

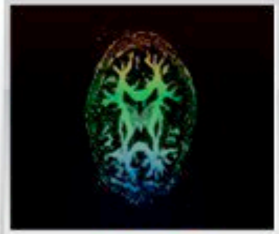
```c
__global__ void add_matrix_gpu
                (float *a, float *b, float *c, int N)
{
   int i=blockIdx.x*blockDim.x+threadIdx.x;
   int j=blockIdx.y*blockDim.y+threadIdx.y;
   int index =i+j*N;
   if( i <N && j <N) c[index]=a[index]+b[index];
}

void main()
{
   dim3 dimBlock (blocksize,blocksize);
   dim3 dimGrid (N/dimBlock.x,N/dimBlock.y);
   add_matrix_gpu<<<dimGrid,dimBlock>>>(a,b,c,N);
}
```

D. Kirk. Parallel Computing: What has changed lately? *Supercomputing*, 2007

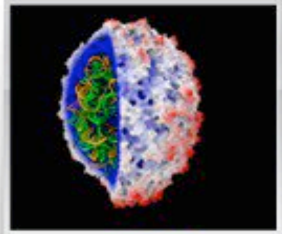
Performance Results

Speedups Using GPU vs CPU



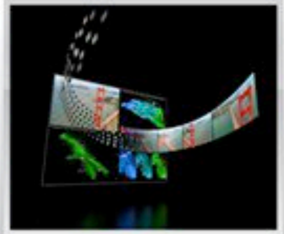
146X

Interactive visualization of volumetric white matter connectivity[1]



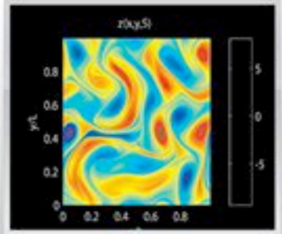
36X

Ionic placement for molecular dynamics simulation on GPU[2]



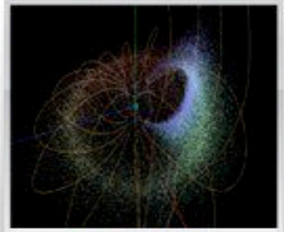
18X

Transcoding HD video stream to H.264 for portable video[3]



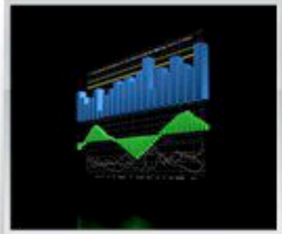
17X

Simulation in Matlab using .mex file CUDA function[4]



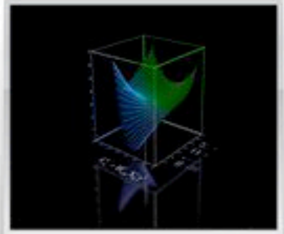
100X

Astrophysics N-body simulation[5]



149X

Financial simulation of LIBOR model

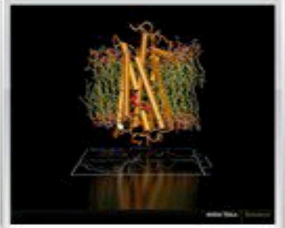


47X

GLAME@lab: M-script API for linear Algebra

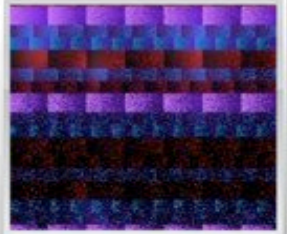
20X

Ultrasound medical imaging for



24X

Highly optimized object oriented

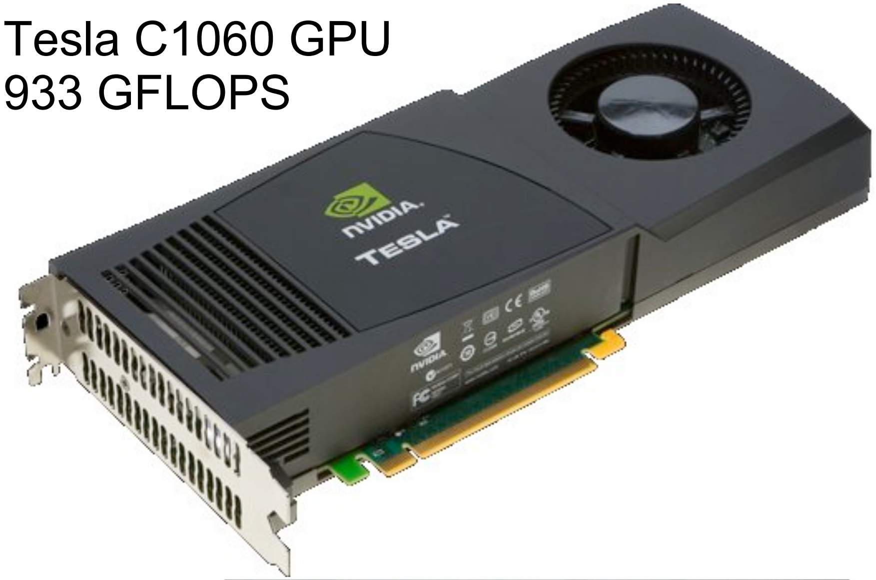


30X

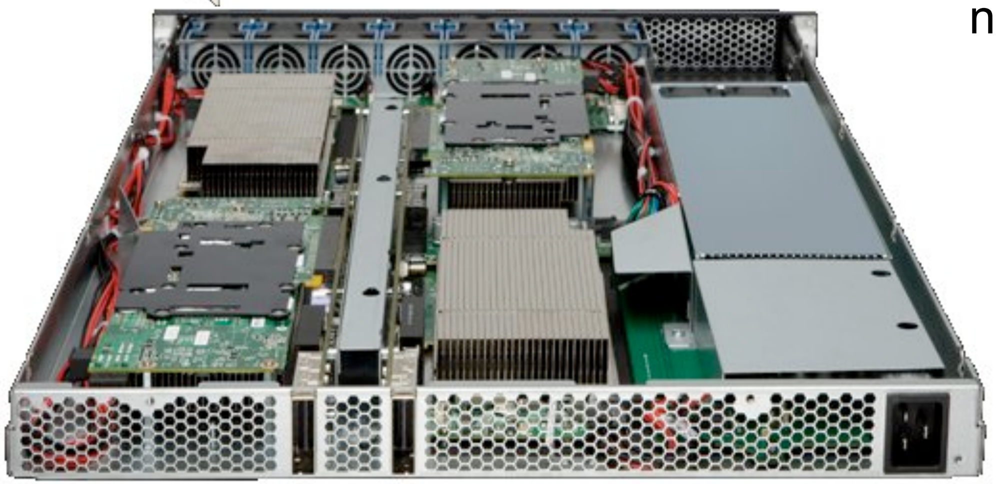
Cmatch exact string matching - find similar

Supercomputing Products

Tesla C1060 GPU
933 GFLOPS



nForce Motherboard



Tesla C1070 Blade 4.14 TFLOPS

Supercomputing Products

Tesla C1060:

- Similar to GTX 280
- No video connections
- 933 GFLOPS peak performance
- 4 GB DDR3 dedicated memory
- 187.8 W max power consumption

Supercomputing Products

Tesla C1070:

- Server Blade
- 4.14 TFLOPS peak performance
- Contains 4 Tesla GPUs
- 960 Cores
- 16GB DDR3
- 408 GB/s bandwidth
- 800W max power consumption

Conclusion

- SIMD causes some problems
- GPU computing is a good choice for fine-grained data-parallel programs with limited communication
- GPU computing is not so good for coarse-grained programs with a lot of communication
- The GPU has become a co-processor to the CPU

References

D. Kirk. Parallel Computing: What has changed lately? *Supercomputing*, 2007.

nvidia.com

NVIDIA. NVIDIA GeForce GTX 200 GPU Architectural Overview. May, 2008.

NVIDIA. NVIDIA CUDA Programming Guide 2.1. 2008.