GPU Basics

Introduction to GPU

PANCHATCHARAM MARIAPPAN
CFD SOFTWARE DEVELOPMENT ENGINEER

August 24, 2017

Outline

History of Super Computing

- Supercomputing is a leading edge of the technology
- Today's Supercomputers are tomorrow Desktop PC
- Supercomputing is the driver of many of the technologies of modern-day processors
- NVIDIA GPU-based machine, Titan (CPU+Tesla GPU) was 1st supercomputer in 2010 and 2nd supercomputer now.
- Titan has almost 300,000 cores (18688 * 16 cores) and 18688 Tesla GPUs.
- Achieves 10 and 20 petaflops per second

History of Super Computing

- Both Supercomputers and desktop are now using heterogeneous computing
- Heterogeneous computing: Mixing of CPU and GPU technology
- Whatever we use as laptop or desktop today were top 500 list 12 years ago
- Think!? Where will be the computing world in the next decade

Von Neumann Architecture

- Almost all processors work on Von Neumann architecture
- Von Neumann - One of the fathers of computing
- Approach: Fetch instruction from memory, decode and then execute
- Modern processors speed: 4GHz

Have a look at this code

```
void Function()
{
  int a[100];
  for(int i=0;i<100;i++)
  {
      a[i]=i*10
  }
}
```

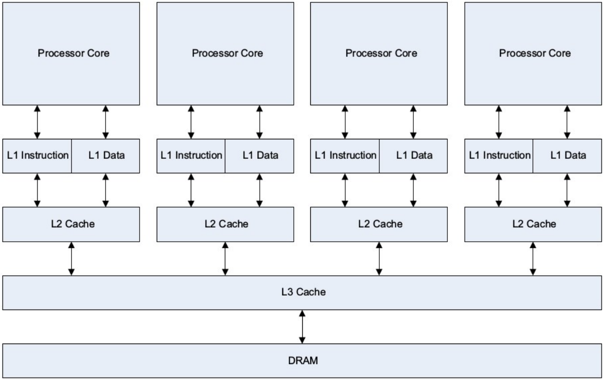
Von Neumann ...

How the processor implement this?

- See the address of `array` loaded into some memory access register
- The parameter `i` would be loaded into another register
- Once the loop exit, 100 is loaded into another register
- Computer iterate around the same instructions 100 times
- For each value, it has control, memory, and calculation instructions fetch and execution

Von Neumann...

- Inefficient as the computer is executing the same instructions but with different values
- Hardware designers implement into just about all processors a small amount of cache
- More complex processors has many levels of cache



Recall

Remember the plumber, toolbox, van, store from Lecture 1

- During fetch from memory, the processor first queries cache
- If data is not in L1 cache, fetch from L2 cache or L3 cache
- If not in any of caches fetch from main memory
- L1 cache runs faster using full processor speed
- L1 cache is only 16 K or 32 K bytes in size
- L2 cache is slower but large in size around 256 K bytes
- L3 cache is in megabytes, but slower than L2

GPU

What is GPU?

GPU

Graphics Processing Unit (GPU) or virutal processing unit (VPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display

What is GPU?

- Manipulate and alter memory to accelerate processes
- Graphics programmers: shaders, texture and fragments
- Parallel programmers: Streams, kernels, scatter and gather
- Stream processing, related to SIMD
- SIMD: Single Instruction Multiple Data

Where is GPU?

GPUs are used in

- Embedded systems
- Mobile Phones
- Personal computers
- Workstations
- Game consoles
- Present on video card or motherboard (intel)

Why GPU?

- GPUs are very efficient at manipulating computer graphics
- Has highly parallel structure
- More effective than general purpose CPUs for algorithms
- Large blocks of data is done in parallel

Let us revisit this later in detail

History of GPUs

1980s

- Intel made the iSBX 275 video graphics controller multimodule board
  - Based on 827220 Graphics Display controller
  - Used to draw lines, arcs, rectangles, bitmaps
- 1985: Commodore Amiga, the first PC with GPU
  - Came with stream processor called blitter
  - Used for accelerated movement
- 1986: Texas, TMS34010, a microprocessor with on chip graphics
- 1987: IBM 8514, one of the first video card

1990s

- 1991: S3 graphics
- 2D GUI acceleration evolved
- CPU assisted real-time 3D graphics become popular
- Fifth generation video games came with play stations
- OpenGL appeared in early 90s as graphics API (Application Program Interface)

NVIDIA

- 1999: The term GPU was popularized by Nvidia
- GeForce 256, the world's first GPU
- GeForce 256 : A single-chip processor with integrated transform, lighting, rendering engines
- Able to construct 10 million polygons per second
- Rethink?! Line drawing using hands at the beginning of the Lecture 1

Note: The term VPU was coined by ATI Technologies

2000-present

- OpenGL, DirectX added programmable shading
- Nvidia produced a chip capable programmable shading, GeForce 3
- October 2002: ATI Radeon, the world's first Direct 3D
- Used to implement looping and lengthy floating point math
- GeForce 8 series was produced by Nvidia
- GPGPU (General Purpose GPU) introduced
- CUDA introduced on June 23, 2007
- OpenCL introduced on August 28, 2009

GPU companies

- Intel
- Nvidia
- AMD/ATI
- S3 Graphics
- Matrox

S. Sundar & M. Panchatcharam

History

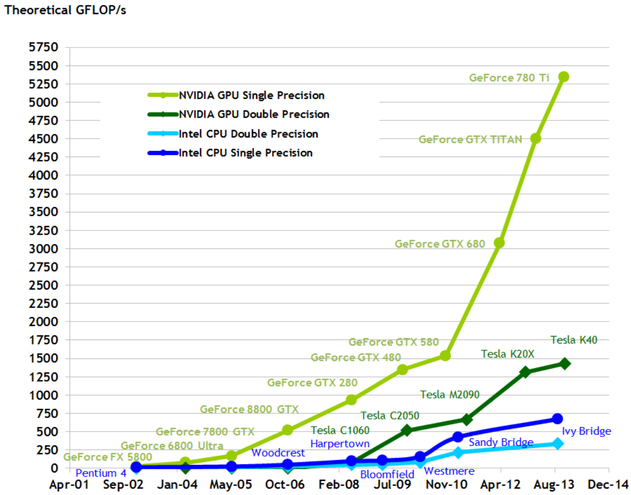
NVIDIA Time line History

- 1993: Funded by Huang, Malachowsky, Priem
- 1995: First product NV1
- 1996: First Microsoft DirectX drivers
- 1997: Riva drivers, 1 million unit sold in 4 months
- 1999: Invents the GPU
- 2000: Graphics Pioneer 3DFx

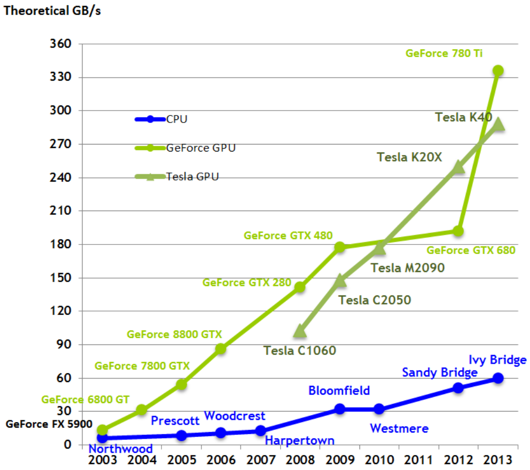
NVIDIA Time line History

- 2001: Enters in Graphics market with NFORCE
- 2005: Develops processor for sony playstation 3
- 2006: CUDA architecture is unveiled
- 2008: Tegra mobile processor launched
- 2009: Fermi architecture launched
- 2010: World's fastest super computer
- 2012: Launches Kepler architecture base GPUs
- 2013: Tegra 4 family mobile processors

Floating Point Operations per second for the CPU and GPU



Memory Bandwidth for the CPU and GPU



CPU vs GPU

- GPU is specialized for compute intensive highly parallel computation
- More transistors are devoted to data processing rather than data caching and flow control

Moore's law: Revisit

Moore's Law

Number of transistors per square inch on integrated circuits had doubled every two years since the integrated circuit was invented

- Scale gets smaller and smaller
- Chip makers came up against law of physics
- The increase in number of transistors in a CPU increase the performance
- CPU architects diminishes where as GPU makers benefit from Moore's law

GPU vs CPU

GPU Basics

S. Sundar & M. Panchatcharam

Super Computing

GPU

History of GPUs

History

Why GPU

GPU vs CPU

GPU Computing

GPU architecture G80 and GT200

Fermi Architecture

Kepler Architecture

CPU

- Designed to get maximum performance from a stream of instructions
- Later, parallelism of instructions came with certain conditions
- Number of unused calculating units increased
- Needs more cache

GPU

- Operation is simple
- Clever technique of handing groups of pixels and polygons simultaneously
- Allot a large part to calculating units
- Does not need more cache

CPU vs GPU

CPU use task parallelism	GPU use data parallelism
Multiple tasks map to multiple threads	SIMD model
Tasks run different instructions	Same instruction on different data
10s of relatively heavyweight threads run on 10s of cores	10,000s of light weight threads on 100s of cores
Each thread managed and scheduled explicitly	Threads are managed and scheduled by hardware
Each thread has to be individually programmed	Programming done for batches of threads

CPU vs GPU

- GPU is specialized for compute intensive highly parallel computation
- More transistors are devoted to data processing rather than data caching and flow control
- Earlier GPU and CPU were separate world
- CPUs were used for office/internet applications
- GPUs were used for drawing nice pictures

CPU vs GPU

- CPU has often called the brain of the PC
- Now PC is enhanced by another part called GPU, which is its soul
- The CPU is composed of only a few cores with lot of cache memory that can handle a few software threads at at time
- A GPU is composed of hundreds of cores that can handle thousands of threads simultaneously
- A GPU with 100+ cores to process thousands of threads can accelerate sofme software by 100x over a CPU alone
- Combination of CPU with GpU can deliver the best value of system performance, price and power

GPU Computing

GPU Computing or GPGPU

GPGPU

GPU accelerated computing is the use of GPU together with a CPU to accelerate scientific, engineering and enterprise applications

Earlier GPGPU Drawbacks

- More complex and precise data types
- Operated with 8 bit integers
- Computational units on GPU in a restrictive way
- Texture unit for read only, frame buffer for write memory
- Vertex and pixel shaders used to execute the kernels

NVIDIA targeted these drawbacks.

GPU Methods

- Mapping: It applies the kernel function to every element in the stream. E.g. constant multiple of each value in the stream
- Reduction: Calculating smaller stream from larger stream
- Stream Filtering: A non-uniform reduction
- Scatter: An operation in vertex processor to adjust the position of vertex
- Gather: A processor to read textures, gather information from any grid cell
- Sort, Search, Data structures, Dense arrays, Sparse arrays, etc.

Applications

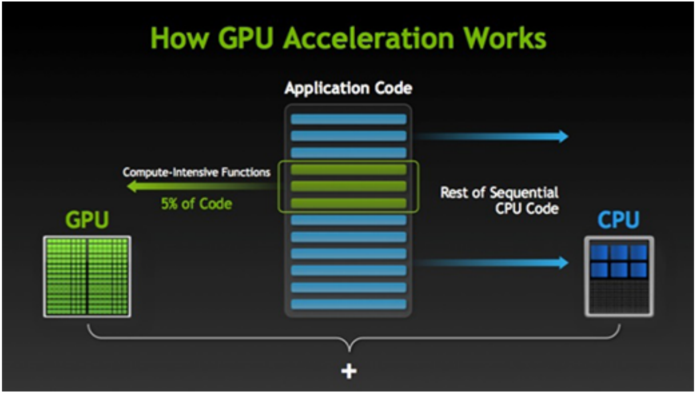
- Neighbor Algorithm
- Grid Computing
- Statistical Physics, CFD,
- Fast Fourier Transform
- Audio signal, Digital Image, video processing
- Bioinformatics, Medical Imaging, Neural Networks, etc

GPU Performance

- CPU comes with dual/quad/hexa/octo cores
- GPU has several generations
- Performance per dollar and performance per watt
  - An exascale computing in USA requires 2 gigawatts of power for petaflop supercomputer.
  - Same exascale computing in NVIDIA Kepler K20 processors requires 150 megawatts power.
  - Also, it performs a quintillion floating point calculations per second
  - 1000 times faster than a petaflop supercomputer

How applications accelerate with GPUs

- GPU computing loads compute intensive portions of the applications
- Remainder of the code still runs on the CPU



Summary

- GPUs use stream processing to achieve high throughput
  - GPUs designed to solve problems that tolerate high latencies
  - High latency $\Rightarrow$ Lower cache requirements
  - Less transistor area for cache $\Rightarrow$ More area for computing units
  - More computing units $\Rightarrow$ 10,000s of SIMD threads and high throghput
- In addition
  - Threads managed by hardware $\Rightarrow$ Not required to write code for each thread and manage them
  - Easier to increase parallelism by adding more processors

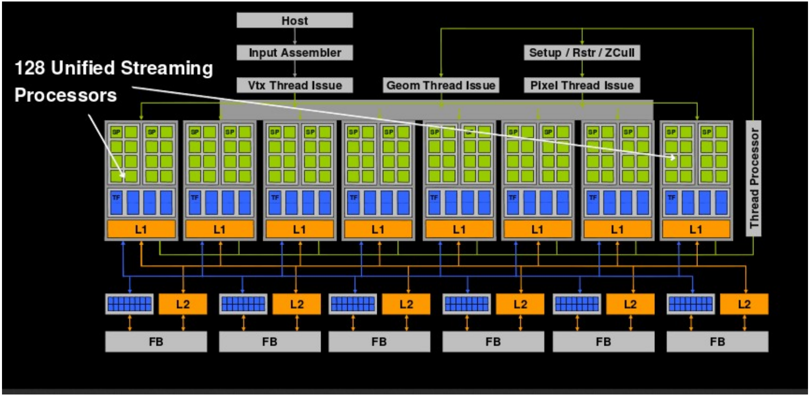
Hence, Fundamental unit of modern GPU is a stream processor

GPU architecture

G80 architecture

- High throughput computing $\Rightarrow$ Programmable streaming processor
- Architecture built around the unified scalar stream processing cores
- GeForce 8800 GTX (G80) was the first GPU architecture built with these features
- It has 16 stream multiprocessors, each with 8 unified streaming processors
- In total 128 streaming processors

G80 architecture

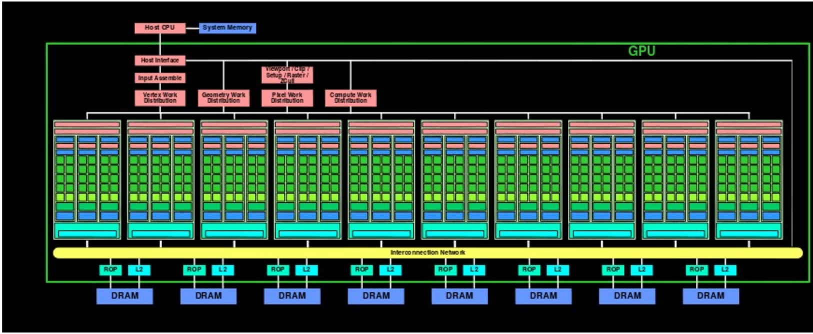


GT200 architecture

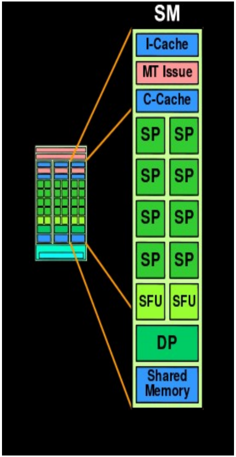
GT200 architecture has

- 1.4 billion transistors
- 240 steaming processors (SPs)
- cache memory
- instruction scheduler
- Two special function units

GT200 architecture



SM



Inside a SM

- Scalar register based ISA
- Multithreaded Instruction unit
  - Up to 1024 concurrent threads
  - Hardware thread scheduling
- 8 SP :Thread Processors
  - IEEE 754 32-bit floating point
  - 32/64-bit integer
  - 16K 32-bit integer
- 2 SFU: Special Function Units: sin,cos...
- Double precision unit
- Fused multiply add
- 16KB shared memory

Memory Hierarchy

- SM can directly access device memory (video memory)
  - Not cached
  - Read & write
  - GT200: 140 GB/s peak
- SM can access device memory via texture unit
  - Cached
  - Read-only, for textures and constants
  - GT200: 48 GTexels/s peak
- On-chip shared memory shared among threads in an SM
  - important for communication amongst threads
  - provides low-latency temporary storage
  - G80 & GT200: 16KB per SM

SIMT

Single Instruction Multiple Data

- Group 32 threads (vertices, pixels or primitives) into warps
  - Threads in warp execute same instruction at a time
  - Shared instruction fetch/dispatch
  - Hardware automatically handles divergence (branches)
- Warps are the primitive unit of scheduling
  - Pick 1 of 24 warps for each instruction slot
- SIMT execution is an implementation choice
  - Shared control logic leaves more space for ALUs
  - Largely invisible to programmer

Summary of G80 and GT200

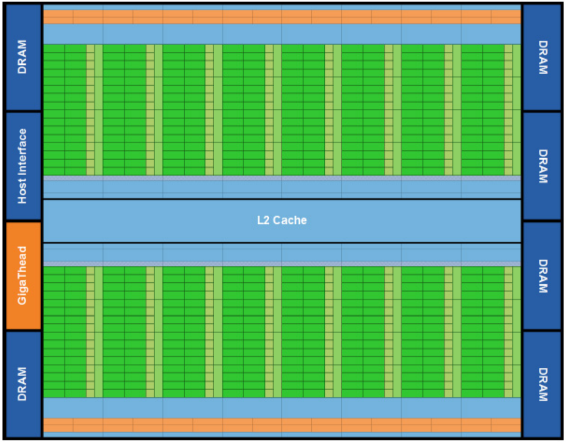
- Execute in blocks can maximally exploits data parallelism
  - Minimize incoherent memory access
  - Adding more ALU yields better performance
- Performs data processing in SIMT fashion
  - Group 32 threads into warps
  - Threads in warp execute same instruction at a time
- Thread scheduling is automatically handled by hardware
  - Context switching is free (every cycle)
  - Transparent scalability. Easy for programming
- Memory latency is covered by large number of in-flight threads
  - Cache is mainly used for read-only memory access (texture, constants

Fermi

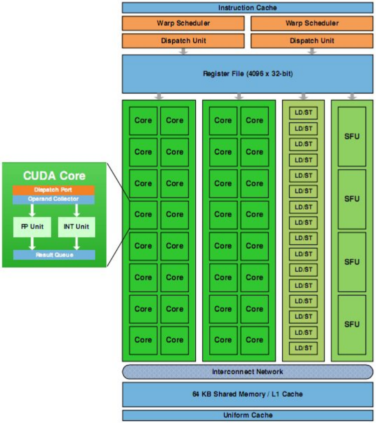
Fermi Architecture

- With 3.0 billion transistors
- 512 CUDA cores
- A CUDA core executes a floating point or integer instruction per clock for a thread
- 512 cores in 16SMs of 32 cores each
- six 64-bit memory partitions
- 6GB GDDR5 DRAM
- Third Generation Streaming Processor

Fermi architecture



Fermi SM



Fermi Streaming Multiprocessor (SM)

SM in Fermi architecture

- It is a third generation streaming multiprocessor
- Each CUDA processor has ALU and FPU (Floating Point Unit)
- IEEE 754-2008 floating point arithmetic
- FMA (Fused Multiply Add) instruction for both float and double
- FMA: Multiply and add instruction are done with a single final rounding step
- 16 Load/Store units
- 4 SFU
- Designed for Double Precision

Dual Warp Scheduler

- SM schedules threads in groups of 32 parallel threads called warps
- Each SM has two warp schedulers
- Each SM has two instruction dispatch units
- Two warps to be issued and executed concurrently
- Fermi achieves peak hardware performance

Shared Memory

- 64 KB Shared Memory
- Shared Memory enables threads within the same thread block to cooperate
- Useful for high performance CUDA applications
- 48 KB L1 cache

Summary table

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

Kepler

Kepler Architecture

- The fastest, most efficient HPC architecture ever built
- It has 7.1 billion transistors
- Provides 1 TFlop (Tera Flop) of double precision throughput with greater than 80% DGEMM efficiency
- Offers huge leap forward in power efficiency
- Delivers 3x performance per watt of Fermi

Kepler Architecture

Kepler has the following features

- Dynamic Parallelism
- Hyper Q
- Grid Management Unit
- GPU Direct
- new SMX architecture
- 15 SMX units and six 64-bit memory controllers
- ECC,L1, L2 cache

Kepler

PCI Express 3.0 Host Interface
GigaThread Engine
SMX
Memory Controller
L2 Cache

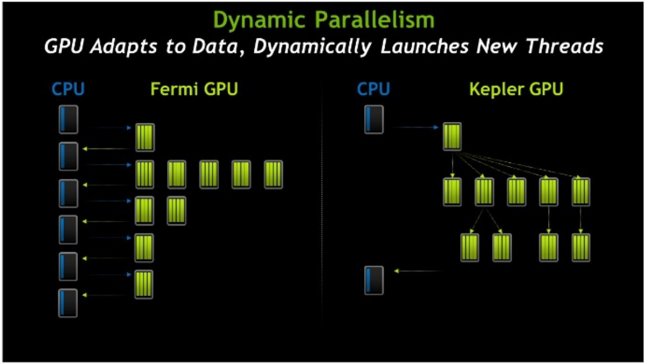
Quad warp scheduler

- SMX schedules threads in groups of 32 parallel threads called warps
- Each SMX has four warp schedulers and eight instruction dispatch units
- Each SMX allows four warps to be issued and executed concurrently
- Selects four warps, two independent instructions per warp per cycle
-

Dynamic Parallelism in GPU

- It is a new feature in GK110, which allows GPU to generate new work to itself synchronize results, control the scheduling of that work via dedicated accelerated hardware paths without CPU
- GK110 job can launch other jobs
- Recursion is possible
- It frees CPU for additional tasks
- Nested loops with differing amounts of parallelism is possible

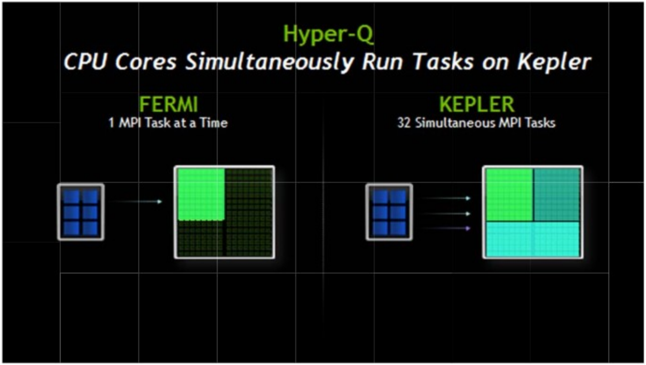
Dynamic Parallelism



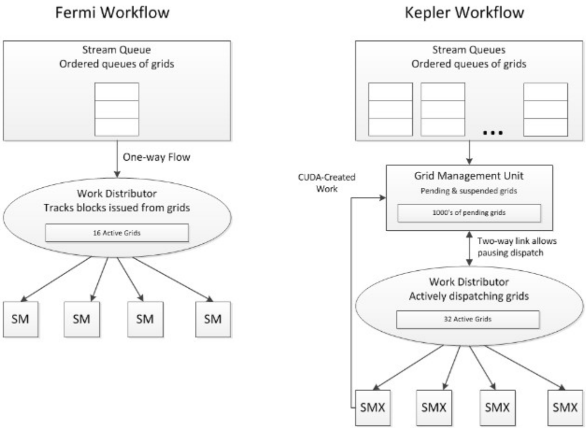
Hyper - Q

- GPU supplied with an optimally scheduled load of work from multiple streams
- Fermi supports 16-way concurrency of kernel launches from separate streams but the streams were all multiplexed into the same hardware work queue
- Hyper-Q increases the total number of connections between the host and the CUDA distributor
- It is a flexible solution that allows connections from multiple CUDA streams, from MPI or even from multiple threads
- Gets 32x performance without any changes in code

Kepler Work Flow



Kepler Work Flow



Summary of Kepler

	FERMI GF100	FERMI GF104	KEPLER GK104	KEPLER GK110
Compute Capability	2.0	2.1	3.0	3.5
Threads / Warp	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16
32-bit Registers / Multiprocessor	32768	32768	65536	65536
Max Registers / Thread	63	63	63	255
Max Threads / Thread Block	1024	1024	1024	1024
Shared Memory Size Configurations (bytes)	16K 48K	16K 48K	16K 32K 48K	16K 32K 48K
Max X Grid Dimension	2^16-1	2^16-1	2^32-1	2^32-1
Hyper-Q	No	No	No	Yes
Dynamic Parallelism	No	No	No	Yes

Compute Capability of Fermi and Kepler GPUs

GPU Compute enables Future Applications

- Enriching the user experience via GPU compute
- Delivering heterogeneous, energy-efficient computing
- Allows developers to unlock the potential of complex applications for consumers

GPU and CT Scans

- CPUs: 2 hours (unusable)
- GPUs: 2 minutes (clinically practical)
- Est. 28000 people/year get cancer from CT scans
- Advanced CT reconstruction reduces radiation by 35-70x

THANK YOU