Master Educational Program
“Information technology in applications”

Mathematical computations with GPUs

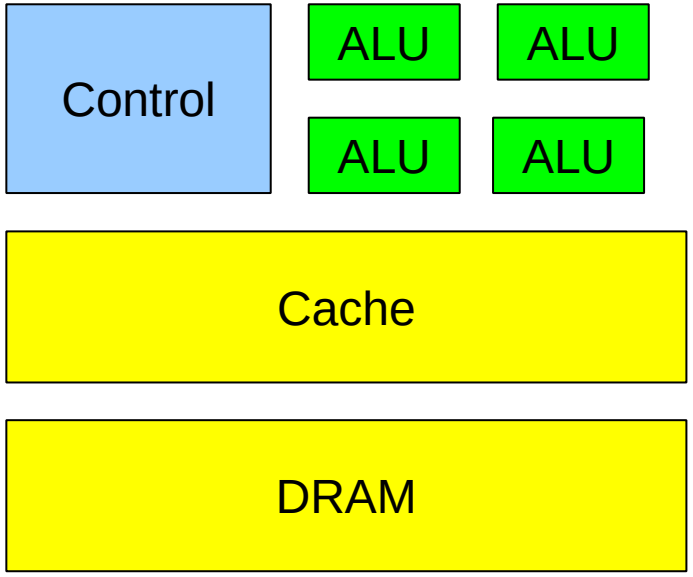
GPU architecture

Alexey A. Romanenko
arom@ccfit.nsu.ru
Novosibirsk State University

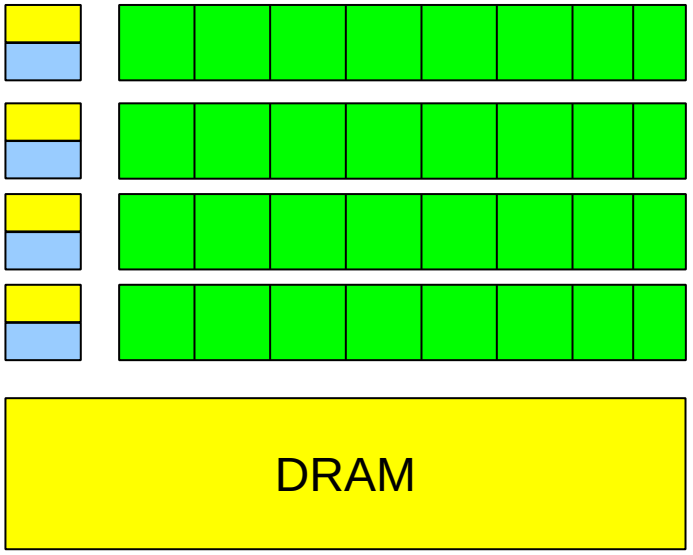
GPU — Graphical Processing Unit

* GPU is a CPU on video-card. GPU has its own parallel architecture (SIMD)
* Program on GPU can't communicate with the host
* Program on GPU can't update host memory
* Uploading/downloading data to/from GPU is performed via PCI Express. Host initiate data exchange process.
* GPU is a co-processor for CPU

CPU vs. GPU



CPU

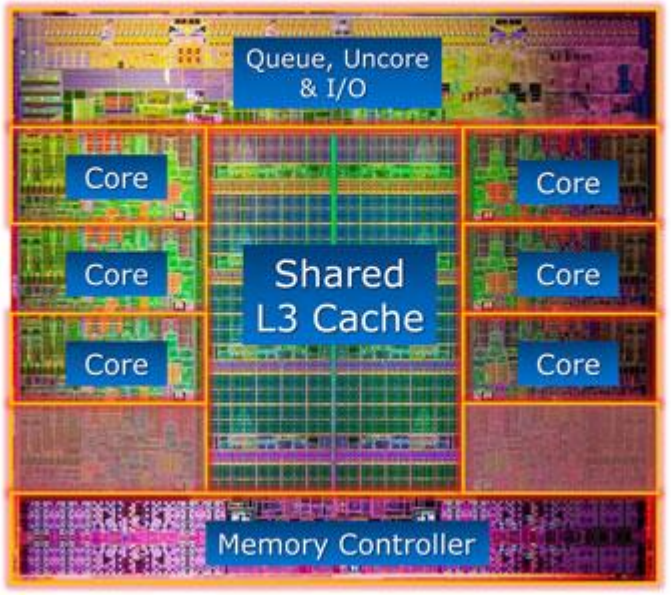


GPU

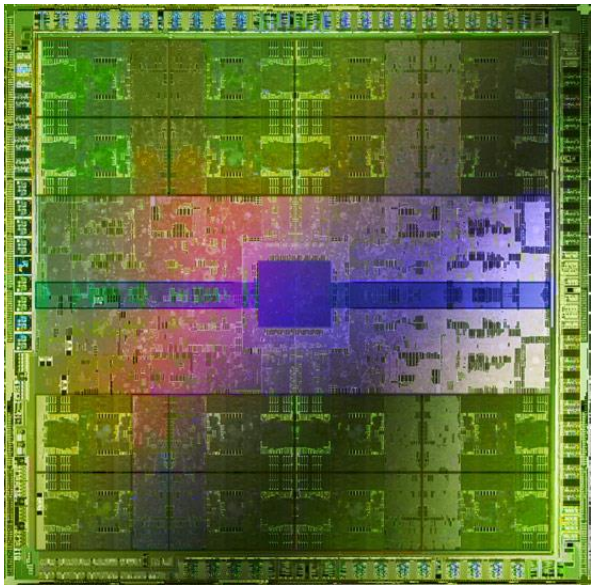
* Less die space for control and cache
* More die space for ALU

CPU vs. GPU

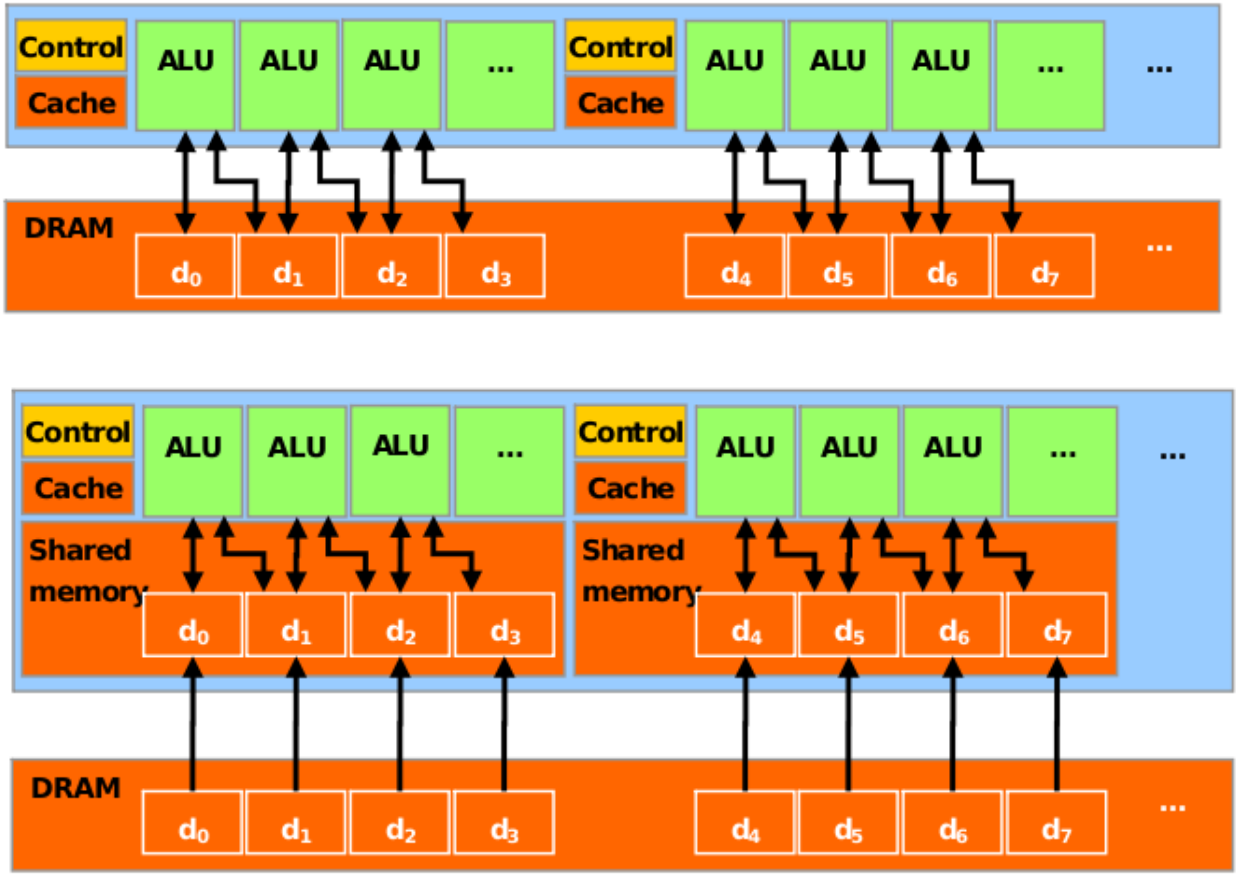
Intel® Core™ i7-3960X Processor Die Detail



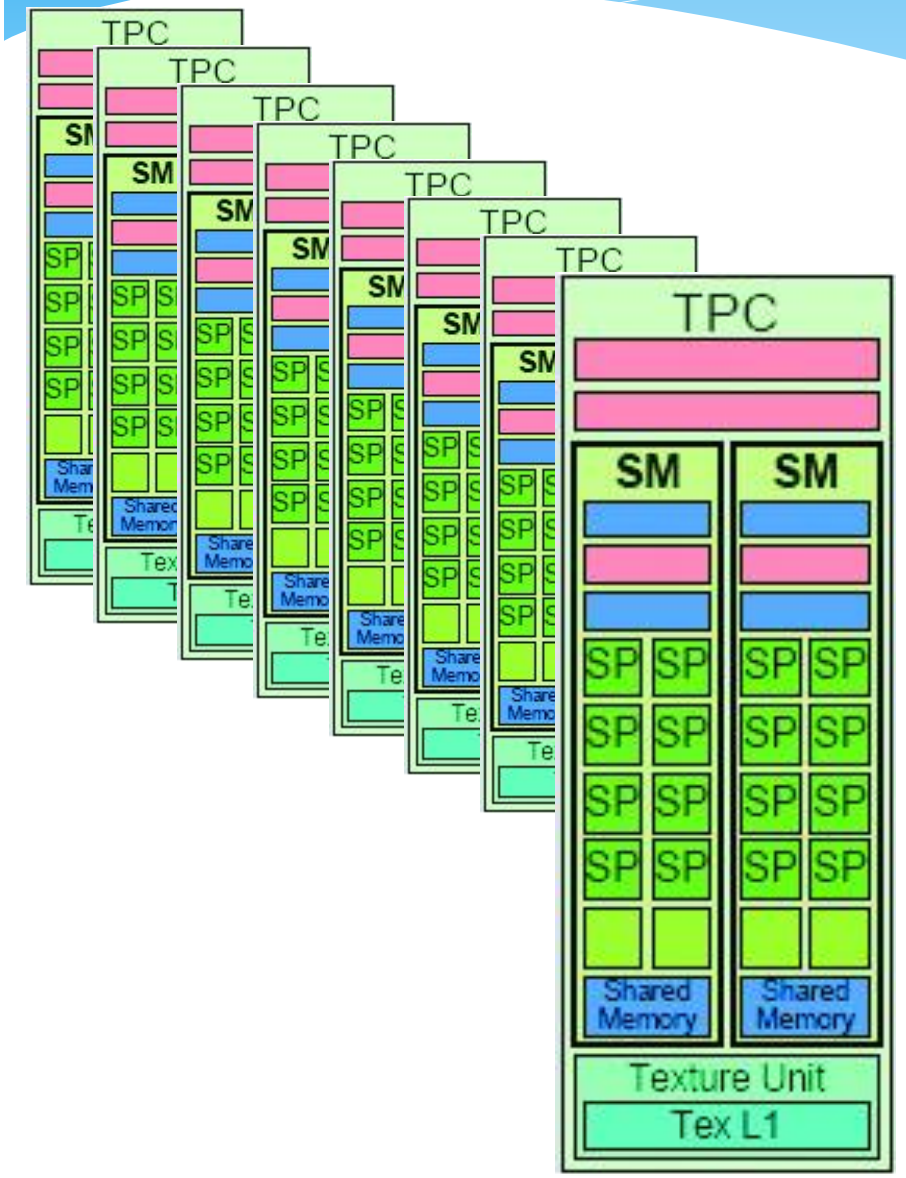
Fermi processor die



Memory access



Architecture of GPU (GT200)



* TPC - Texture Processor Cluster
* SM — Streaming Multiprocessor
  * Multi-threaded processor core
  * Fundamental processing unit for CUDA thread block
* SP — Streaming Processor
  * Scalar ALU for a single CUDA thread

	Number of SM
GeForce 8800 GTX	16
GeForce 8800 GTS	12
Tesla D870	2x16
Tesla S870	4x16
Tesla C1060, GT200, Tesla T10	30
Tesla S1070	4x30

Compute capatibility

* Compute Capability 1.0+
  * Asynchronous kernel execution
* Compute Capability 1.1+
  * Asynchronous copy engine (single engine). Property **asyncEngineCount**
  * Atomic operation
* Compute Capability 1.3+ (e.g. C1060)
  * Double precision arithmetic
* Compute Capability 2.0+ (e.g. C2050)
  * Parallel kernel execution on GPU (property **concurrentKernels**)
  * Two asynchronous copy engines (property **asyncEngineCount**)
* Compute Capability 3.0+ (e.g. K10)
  * Warp shuffle functions
* Compute Capability 3.5+ (e.g. K20)
  * Funnel shift
  * Dynamic parallelism

http://en.wikipedia.org/wiki/CUDA

Tesla C1060

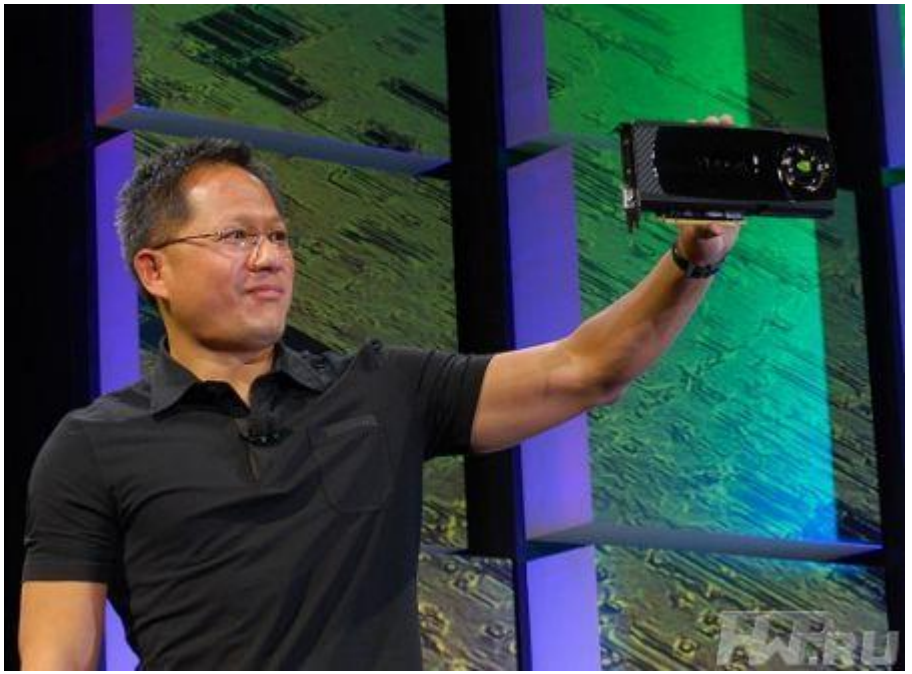
1 TFlops

Tesla S1070

4 TFlops

Fermi

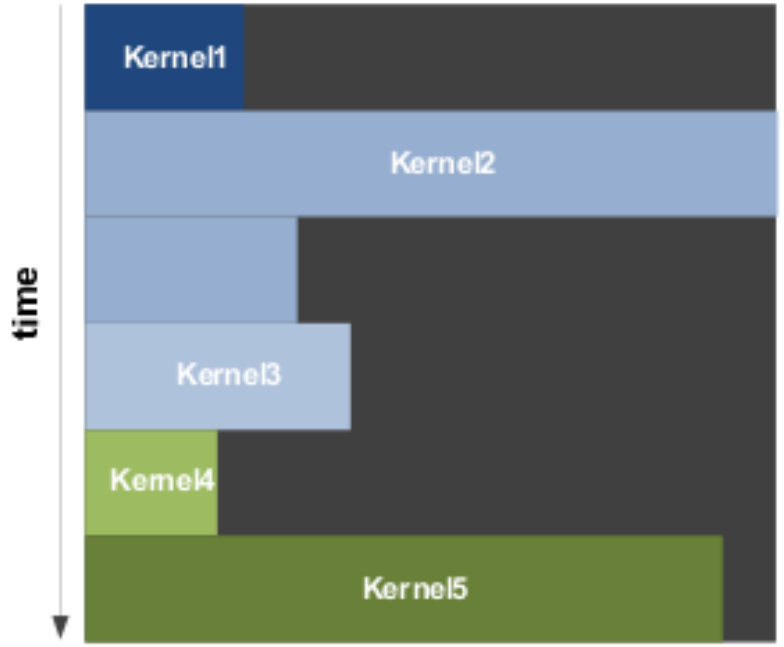
* Announced in Sept., 2009
* May 2010 – start of selling new videocard GT300 series
* Codename - Fermi



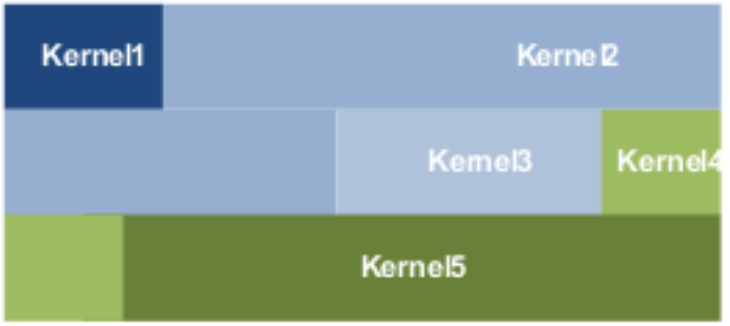
Fermi. Features

* 3 billion transistors, 40-nm
* 512 CUDA cores, support IEEE 754-2008, 16 SM
* Clock rate is about 1,5 GHz
* 128 texture blocks
* 384-bit memory controler GDDR5 (6x64 bit)
* Memory bandwidth is about 192 Gbit/s
* 1,5 TFlopsSP, 750 GFlops DP
* Interface - PCI Express x16 2.0
* C++, in addition to C, Fortran, Java, Python, OpenCL, DirectCompute.
* ECC

Fermi. Features

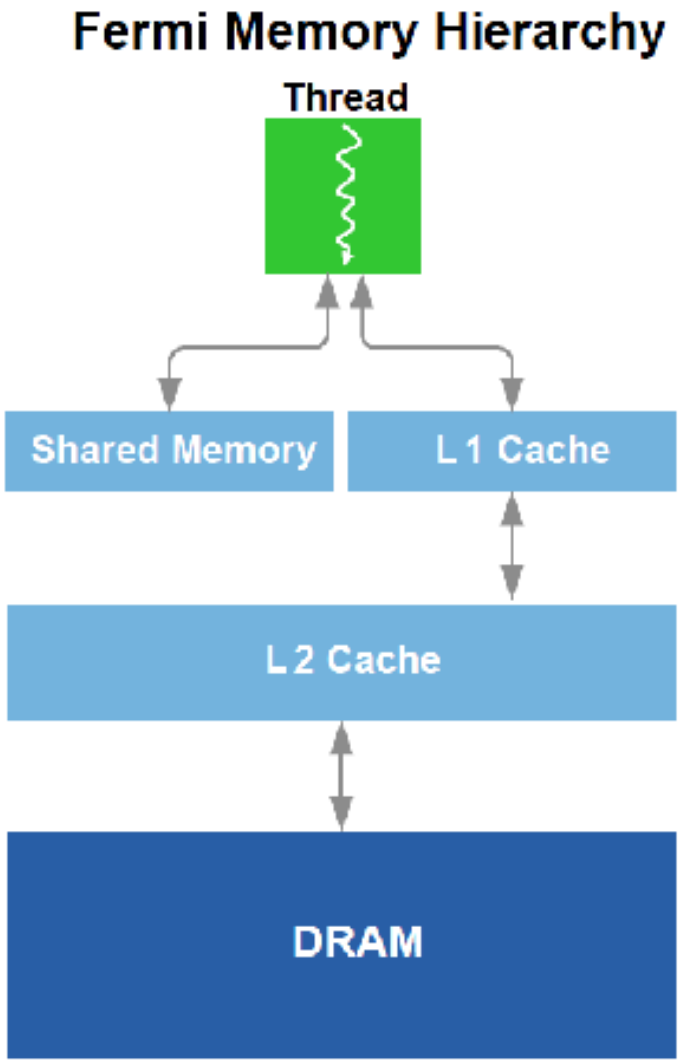


Serial Kernel Execution



Concurrent Kernel Execution

Fermi. Features

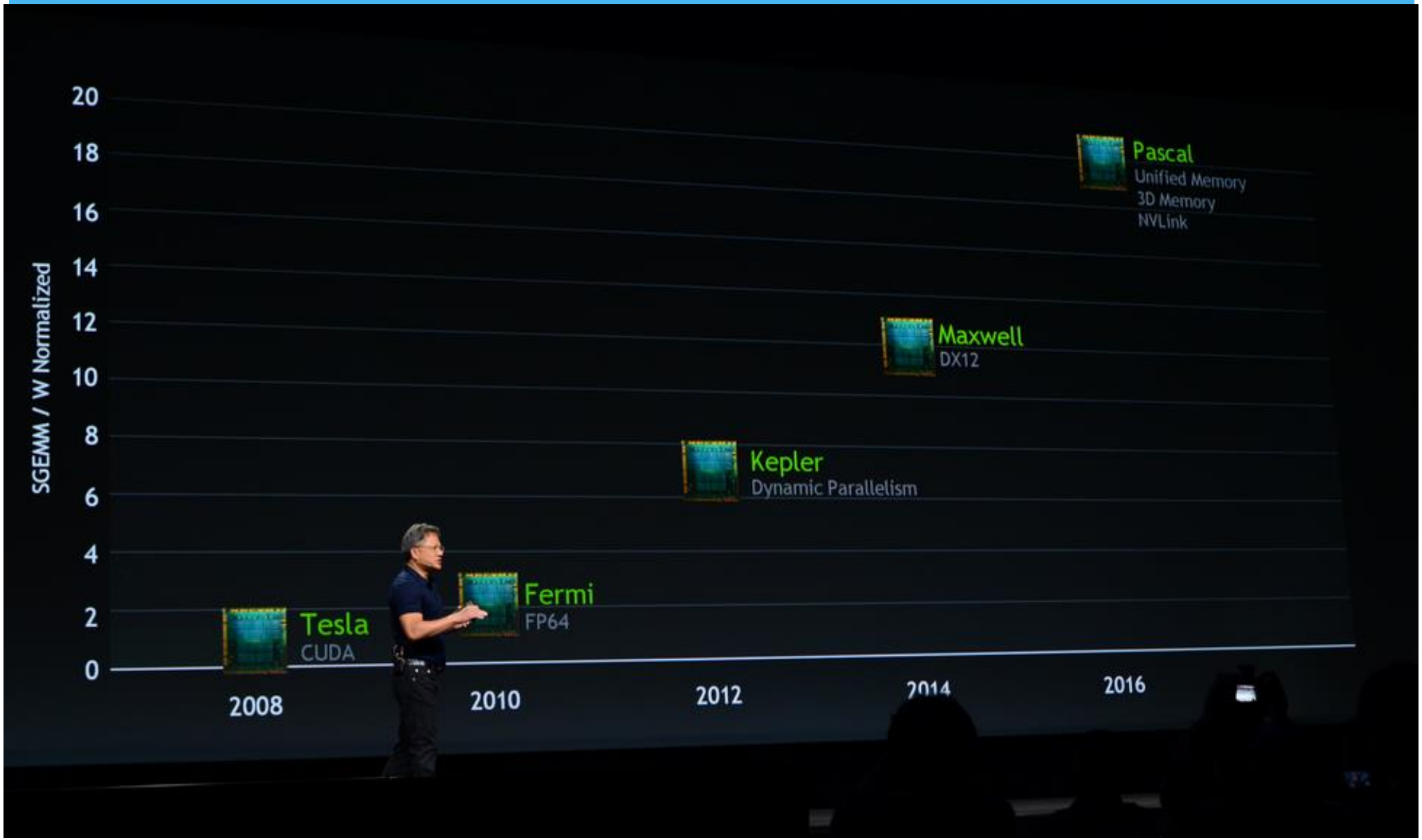


* NVIDIA Parallel DataCache™ - the first hierachical cache on GPU

GPU specification

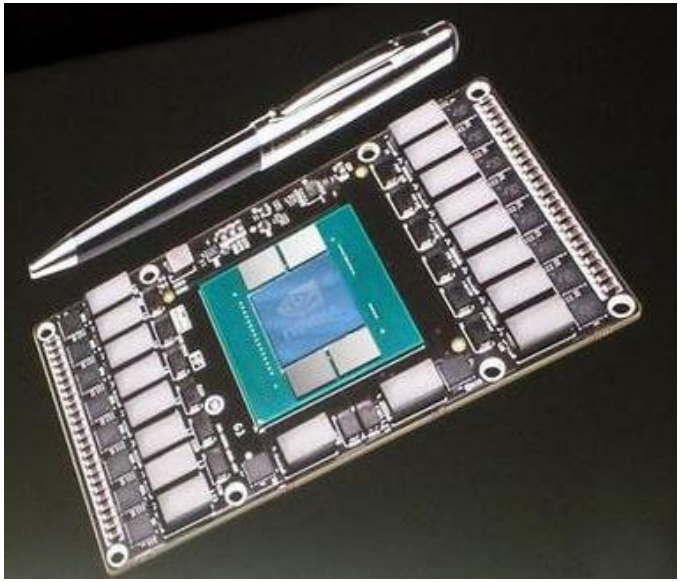
Tech. Specification	Tesla M2090	K10	K20	K20x	K40
Performance DP	0.6 TFlops	0.19 TFlops	1.17 Tflops	1.31 Tflops	1.43 Tflops
Performance SP	1.3 TFlops	4.58 TFlops	3.52 Tflops	3.95 Tflops	4.29 Tflops
Number of GPUs	Fermi	2x GK104s	GK110	GK110	GK110B
Memory (GDDR5)	5GB	2x 4GB	5GB	6GB	12GB
Number of CUDA cores	512	2x 1536	2496	2688	2880
Bandwidth (ECC off)	177 GB/s	320 GB/s	208 GB/s	250 GB/s	288 GB/s
Architecture features		SMX	SMX, Dynamic Parallelism, Hyper-Q		

GTC 2014


SGEMM / W Normalized
20
18
16
14
12
10
8
6
4
2
0
2008
2010
2012
2014
2016
Tesla
CUDA
Fermi
FP64
Kepler
Dynamic Parallelism
Maxwell
DX12
Pascal
Unified Memory
3D Memory
NVLink

GTC 2014 announcements

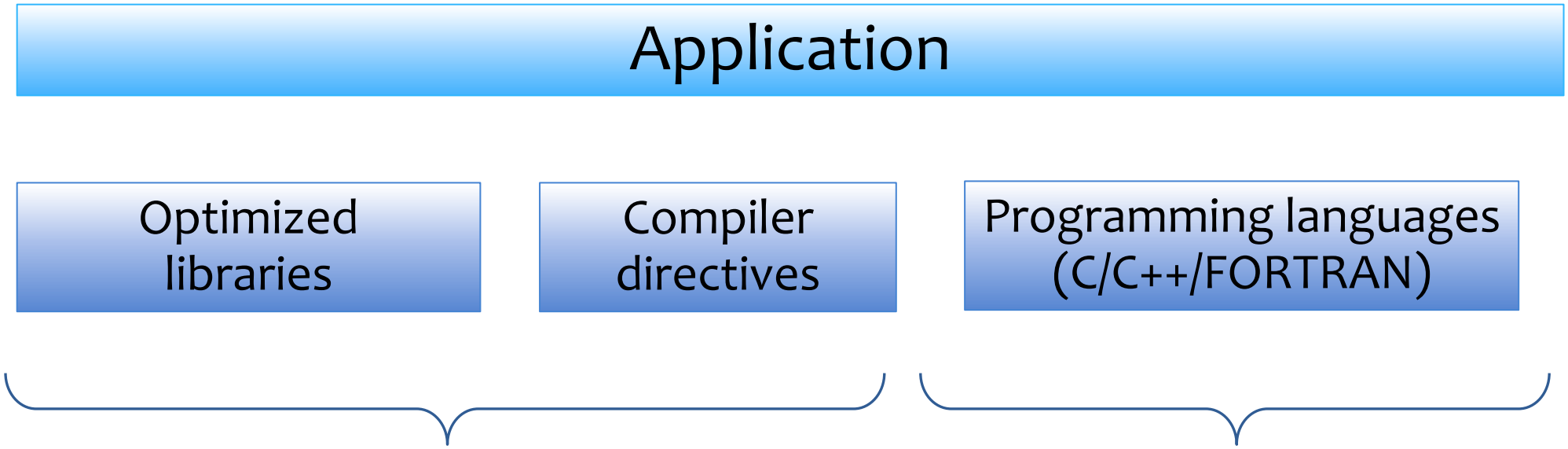
* Titan Z
  * 2x GK110
  * 5760 CUDA cores
  * 8 TFlops (SP)
  * 12GB
* Pascal architecture
  * Start selling 2016
  * 3D memory
  * Bandwidth 5-12x vs. PCIe 3.0
  * NVLink



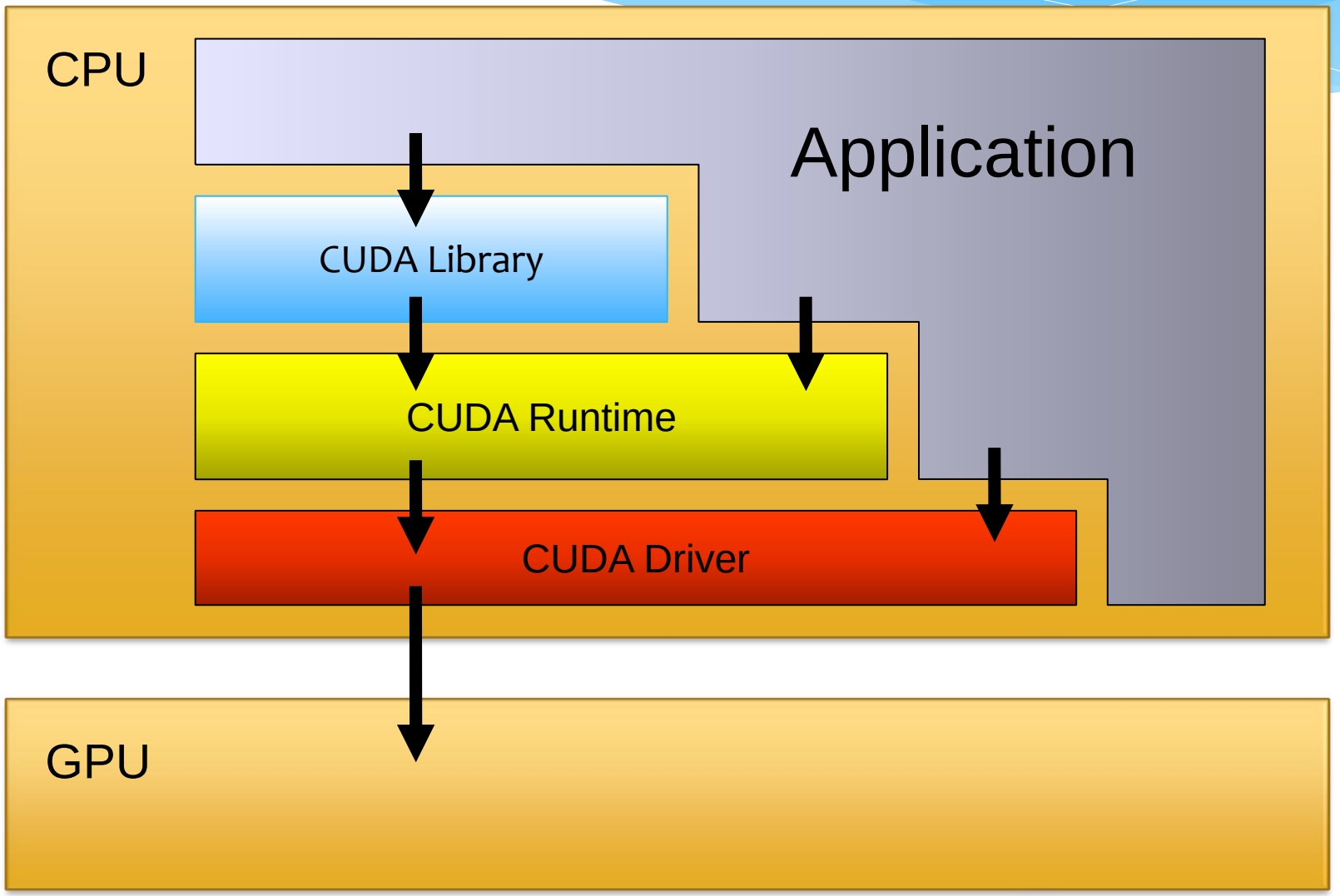
Maxwell

GPU	GK107 (Kepler)	GM107 (Maxwell)
CUDA Cores	384	640
Base Clock	1058 MHz	1020 MHz
GPU Boost Clock	N/A	1085 MHz
GFLOP/s	812.5	1305.6
Compute Capability	3.0	5.0
Shared Memory / SM	16KB / 48 KB	64 KB
Register File Size / SM	256 KB	256 KB
Active Blocks / SM	16	32
Memory Clock	5000 MHz	5400 MHz
Memory Bandwidth	80 GB/s	86.4 GB/s
L2 Cache Size	256 KB	2048 KB

Approaches to GPU programming



CUDA - Compute Unified Device Architecture



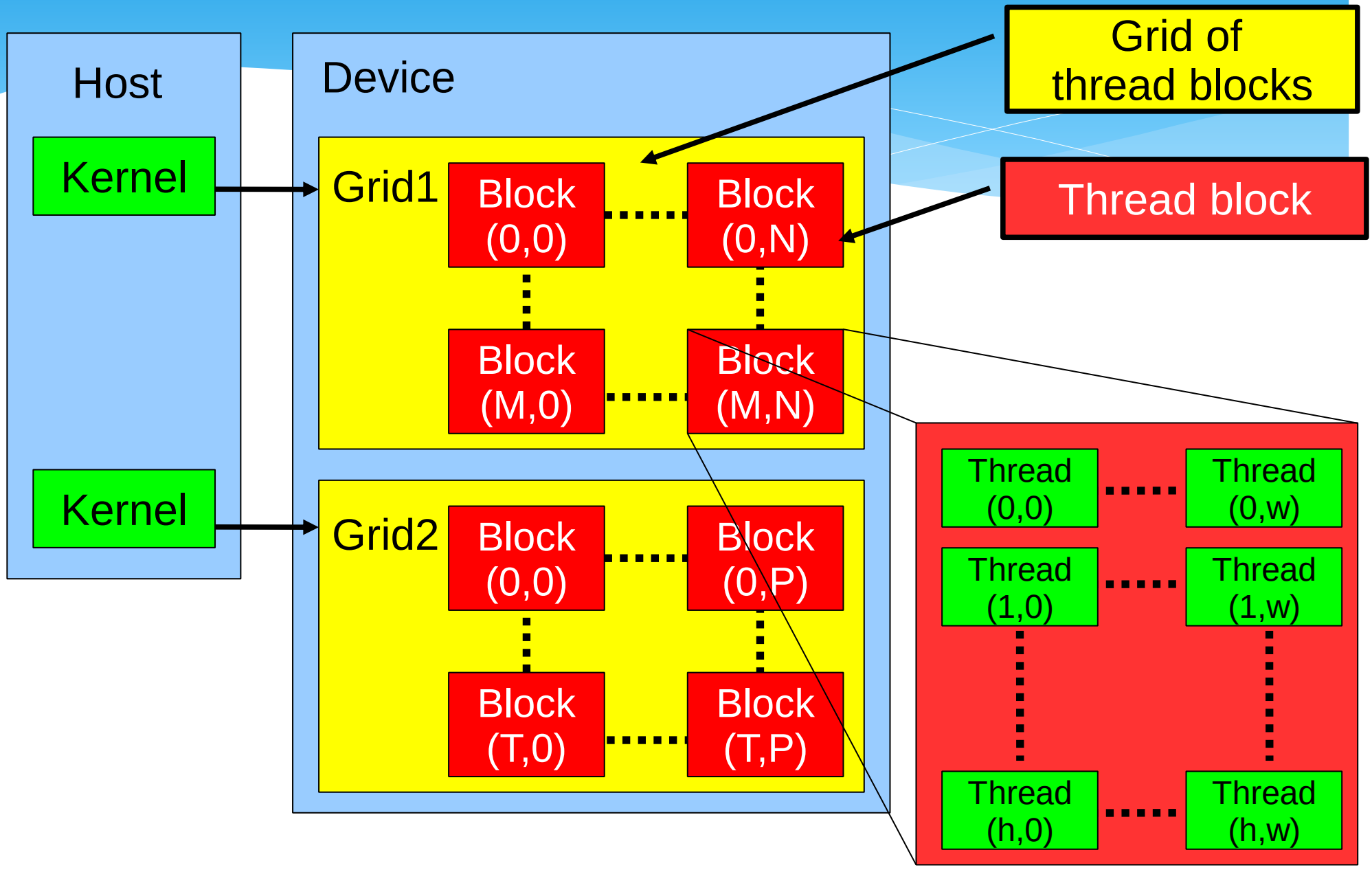
Definitions

* **Thread** — execution unit of instruction flow
* **Thread block** — logical set of **threads.**
* **Warp**— group of **threads** inside **thread block** which are physically executed concurrently (32 threads)
* **Grid** — set of **thread blocks**

Programming model

* GPU has its own memory
* Program is split into threads and executed on SP
* SP has an access to the shared memory inside SM and global memory of GPU
* Thread synchronization allowed only inside SM (thread block)
* Execution is organized as a GRID of thread blocks
* Program on GPU is called «kernel»

Kernel execution



Thread block

* Each thread has its own ID— threadIdx
* Threads can be mapped to 1D, 2D, or 3D grid. threadIdx variable is a structure with 3 fields (x,y,z)
* Thread block size is configured beforee the launch.
* Threads in thread block ≤ 512 or 1024 (depends on GPU)

Grid of thread blocks

* Each thread block has its own ID— blockIdx
* Thread blocks can be mapped to 1D or 2D grid. blockIdx variable is a structure with 3 fields (x,y,z)
* Size and structure of the Grid is configured before the kernel launch

Example

* Let's 2D grid is configured (HxW thread blocks). Each thread block is 2D grid also (MxK threads)
* Thus, whole computational area covered with
  * H*M threads vertically
  * W*K threads horizontally
* The coordinate of the thread is (blockID.x*M + threadID.x, blockID.y*K + threadID.y)

Execution model

* Thread block is executed on Stream Multiprocessor
  * One thread block can use only one SM
  * Schedule of thread block execution is not defined
* Number of thread blocks processed by SM defines by number of registers and shared memory available
* Active thread block is a block currently executed
* Each active thread block is split into SIMD groups — warps. Each warp contains equal number of threads
* Scheduler pass the control from one warp to another one periodically
* Distribution of threads over warps is not depends on the launch

Candidates for GPU parallelization

* Good candidates
  * Task which is well-suited to SIMD parallelization
    * Data-parallel computations
* Bad candidates
  * Task which is not well-suited to SIMD parallelization
    * Task-parallel computations
  * Task which is naturally sequential

Questions