Introduction to CUDA Parallel Programming
CUDA平行計算導論

https://ceiba.ntu.edu.tw/1092Phys8061_CUDA

Professor Ting-Wai Chiu (趙挺偉)
Email: twchiu@phys.ntu.edu.tw
Physics Department
National Taiwan University

Introduction to CUDA parallel programming
CUDA平行計算導論

This course introduces the **C**ompute **U**nified **D**evice **A**rchitecture (CUDA) parallel programming model, and its applications in science and engineering.
The topics include:

- An Overivew of Nvidia GPUs, and GPU Accelerated Computation with CUDA
- CUDA Programming Model
- Threads and Blocks
- Shared Memory, Constant Memory, and Texture Memory
- Solving Partial Differential Equation with CUDA
- CUDA Libraries: cuBLAS, cuFFT, MAGMA (LAPACK)
- Multi-GPUs with OpenMP
- Linear System
- Conjugate Gradient with Mixed Precision
- Monte Carlo Simulation, The Ising Model

Grading

- Homework 60%
- Final Project 30%
- Class Participation and Discussions 10%
- Total 100%

Guidelines

- You are allowed to discuss HW assignments with other students in the class.
- However, copying other's code is not acceptable
- Also, copying some library code that directly gives the solution of a HW/Project is not acceptable

TA: 蘇冠銘 Guang-Ming Su <d07222009@ntu.edu.tw>

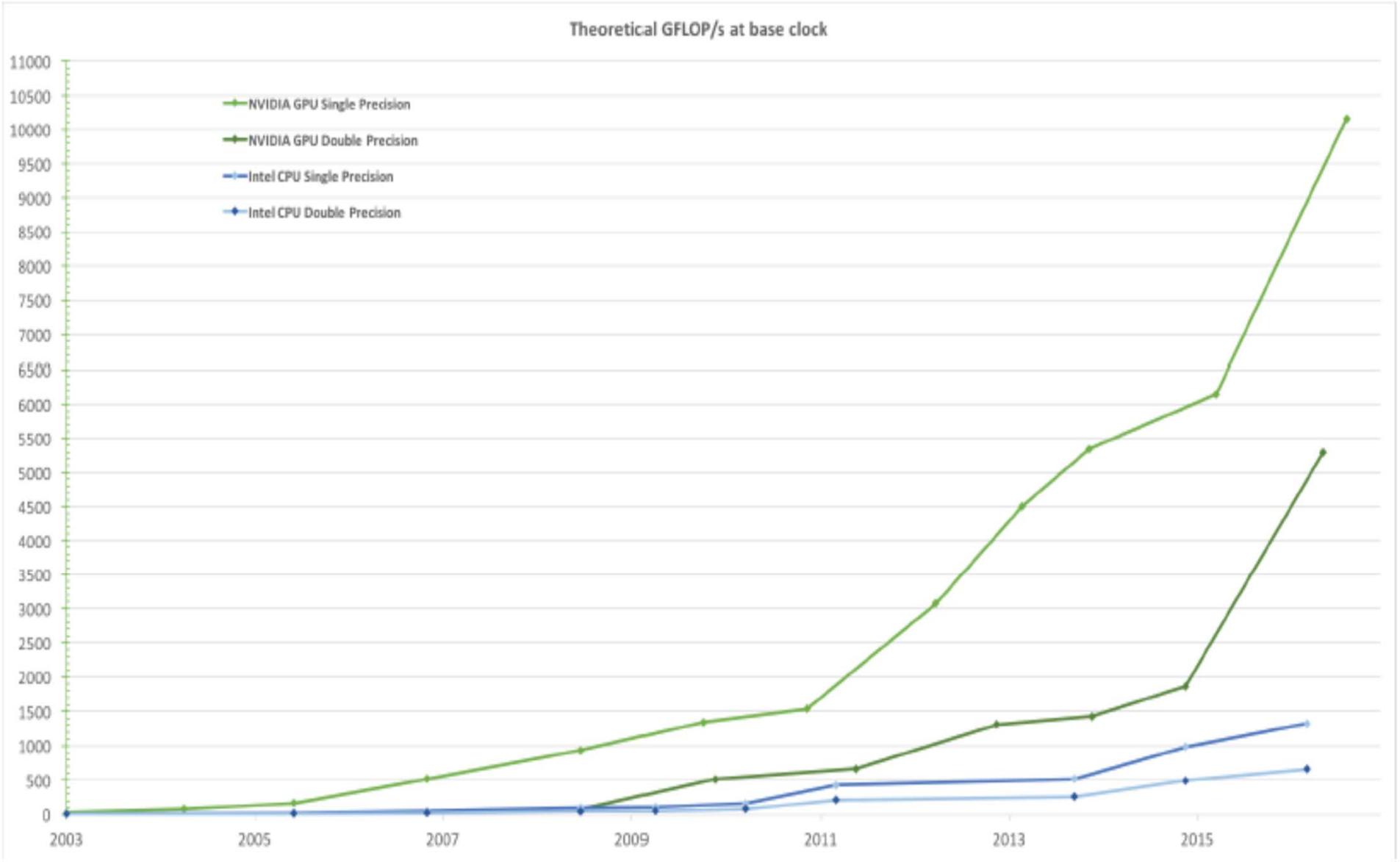
GPU Accelerated Computation

A graphic card (e.g., Nvidia GTX1060) is capable to deliver > 350 Gflops (sustained) with the price less than NT$6,000. It gives a speed up 10x –100x comparing with a single CPU.

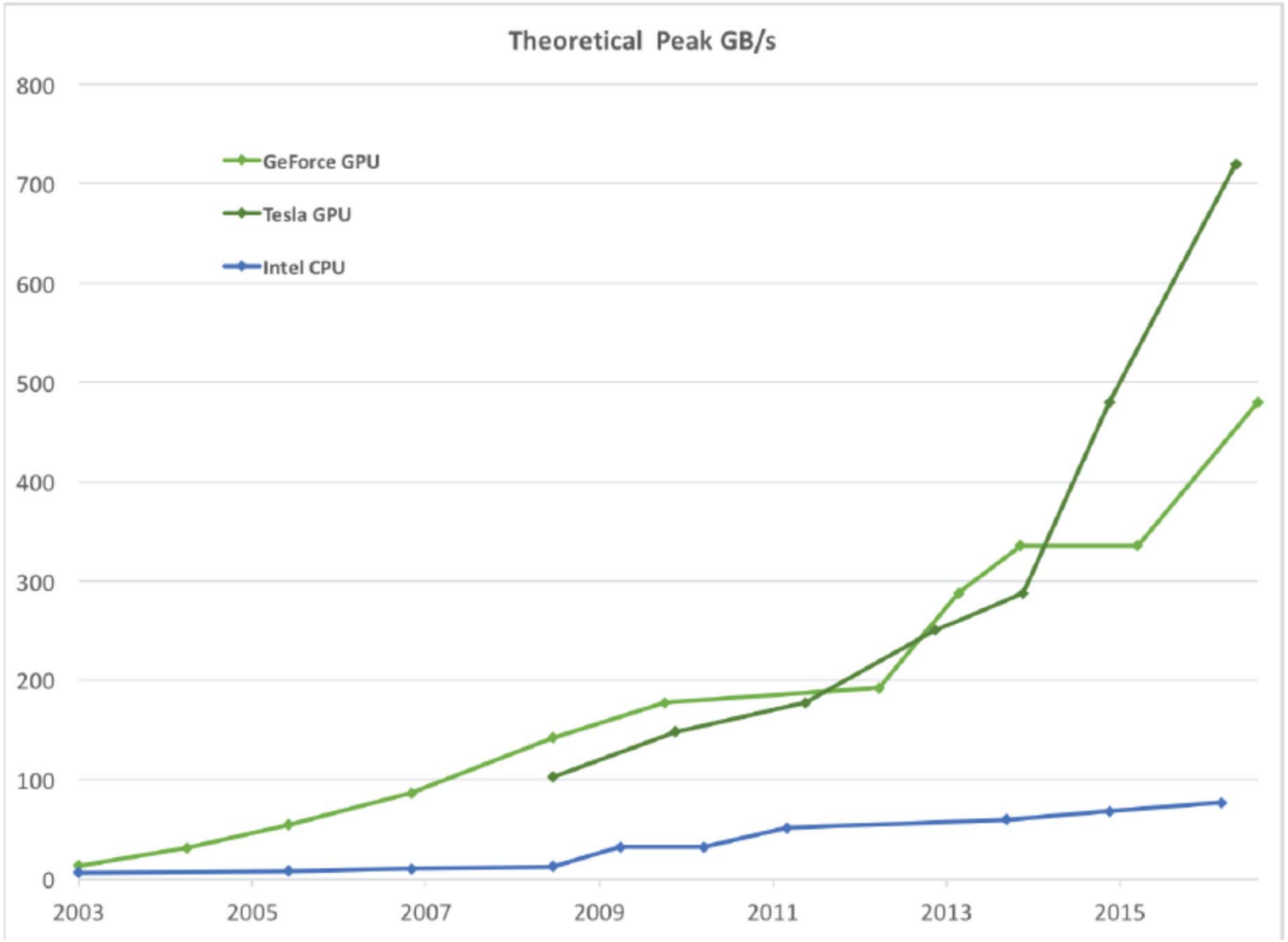
Two Nvidia GPU cards on the motherboard

- **This opens up a great opportunity for many scientific and engineering problems which require enormous amount of number-crunching power.**
- **Recall that in the past 60 years, each 10x jump in computing power motivated new ways of computing, which in turn led to many scientific breakthroughs.**

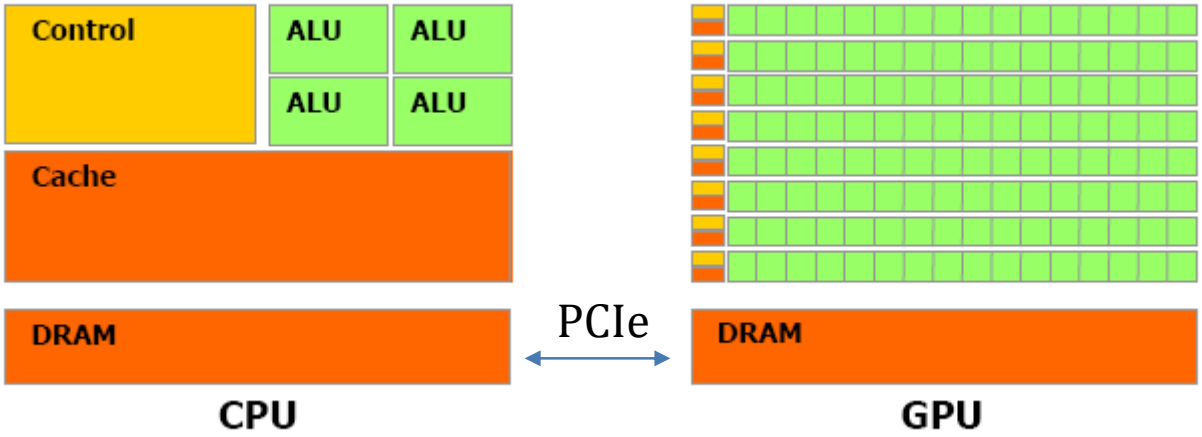
GPU vs. CPU --- GFLOPS Comparison



GPU/CPU Memory Bandwidth Comparison

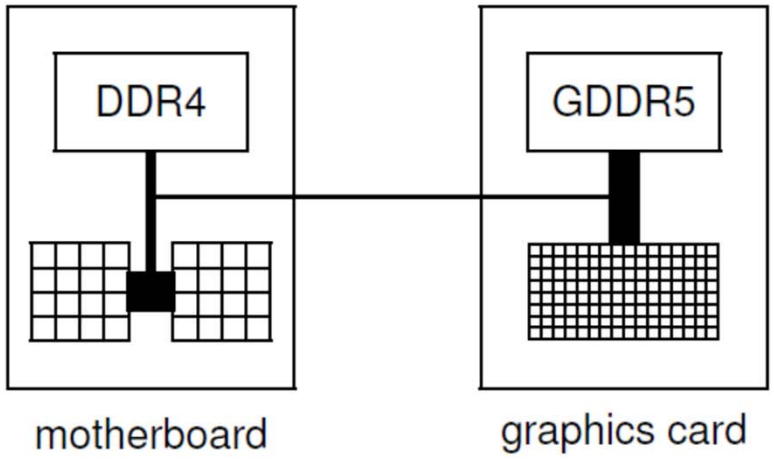


GPU Devotes More Transistors to Data Processing



The reason behind the discrepancy in floating-point capability between the CPU and the GPU is that the **GPU is specialized for compute-intensive, highly parallel computation** – exactly what graphics rendering is about – and therefore designed such that more **transistors are devoted to data processing rather than data caching and flow control.**

GPU Accelerated Computation – The Hardware View

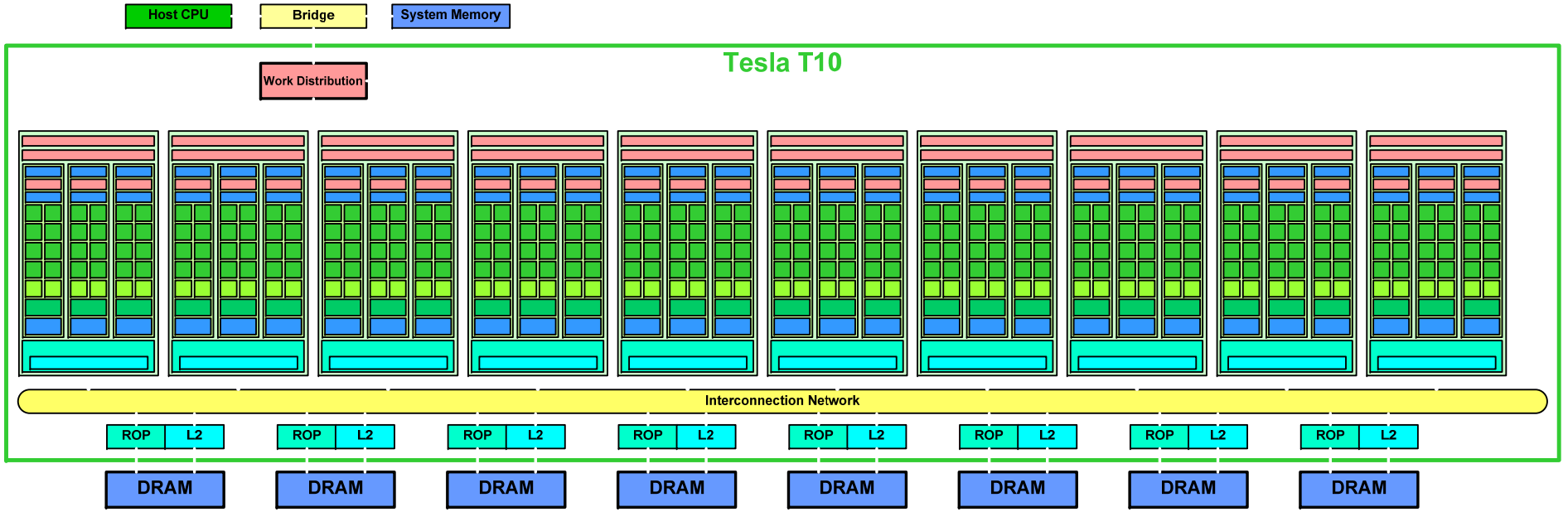


GPU Accelerated Computation – The Basic Idea

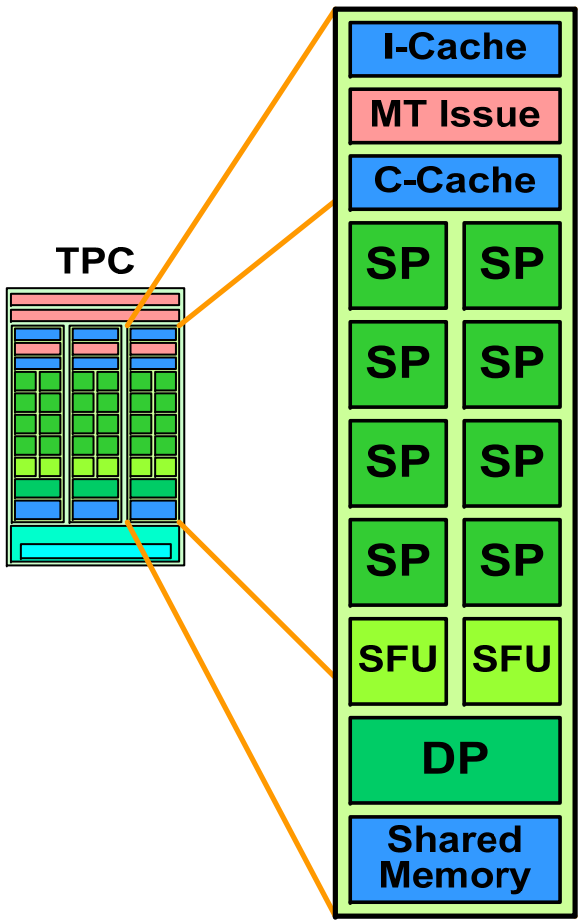
- The GPU is connected to the CPU by the PCIe bus
- The idea is to use the **GPU as a co-processor**
  - CPU controls the execution of the program, and prepares the initial data for GPU
  - Send **big** parallelizable computations to the GPU
  - Need to copy data to the GPU, and fetch results back. This works well if the data transfer is overshadowed by the number crunching done using that data.

Tesla 1070 Architecture (2008)

- Massively parallel general computing architecture
- 30 multiprocessors @ 1.45 GHz with 4 GB of RAM
  - 1 TFLOPS single precision (IEEE 754 floating point)
  - 87 GFLOPS double precision (crucial for science/engineering problems)



Multiprocessor of Tesla T10



- 8 SP Thread Processors
  - IEEE 754 32-bit floating point
  - 32-bit float and 64-bit integer
  - 16K 32-bit registers
- 2 SFU Special Function Units
- 1 Double Precision Unit (DP)
  - IEEE 754 64-bit floating point
  - Fused multiply-add
- Scalar register-based ISA
- Multithreaded Instruction Unit
  - 1024 threads, hardware multithreaded
  - Independent thread execution
  - Hardware thread scheduling
- 16KB Shared Memory
  - Concurrent threads share data
  - Low latency load/store

P100 Architecture (2016)

Streaming Multiprocessor of P100

Nvidia GTX1060

Two Nvidia GTX1060 on the motherboard

deviceQuery (GTX1060)

```
Device 0: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6078 MBytes (6373179392 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1759 MHz (1.76 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),
                                                 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  ...
  ...
```

deviceQuery (GTX1060)

```
Device 0: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6078 MBytes (6373179392 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1759 MHz (1.76 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),
                                                 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  ...
  ...
```

deviceQuery (GTX1060)

```
Device 0: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6078 MBytes (6373179392 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1759 MHz (1.76 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),
                                                 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  ...
  ...
```

deviceQuery (GTX1060)

```
Device 1: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6078 MBytes (6373572608 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1759 MHz (1.76 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),
                                                 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  ...
  ...
> Peer access from GeForce GTX 1060 6GB (GPU0) -> GeForce GTX 1060 6GB (GPU1) : Yes
> Peer access from GeForce GTX 1060 6GB (GPU1) -> GeForce GTX 1060 6GB (GPU0) : Yes
```

deviceQuery (GTX1060)

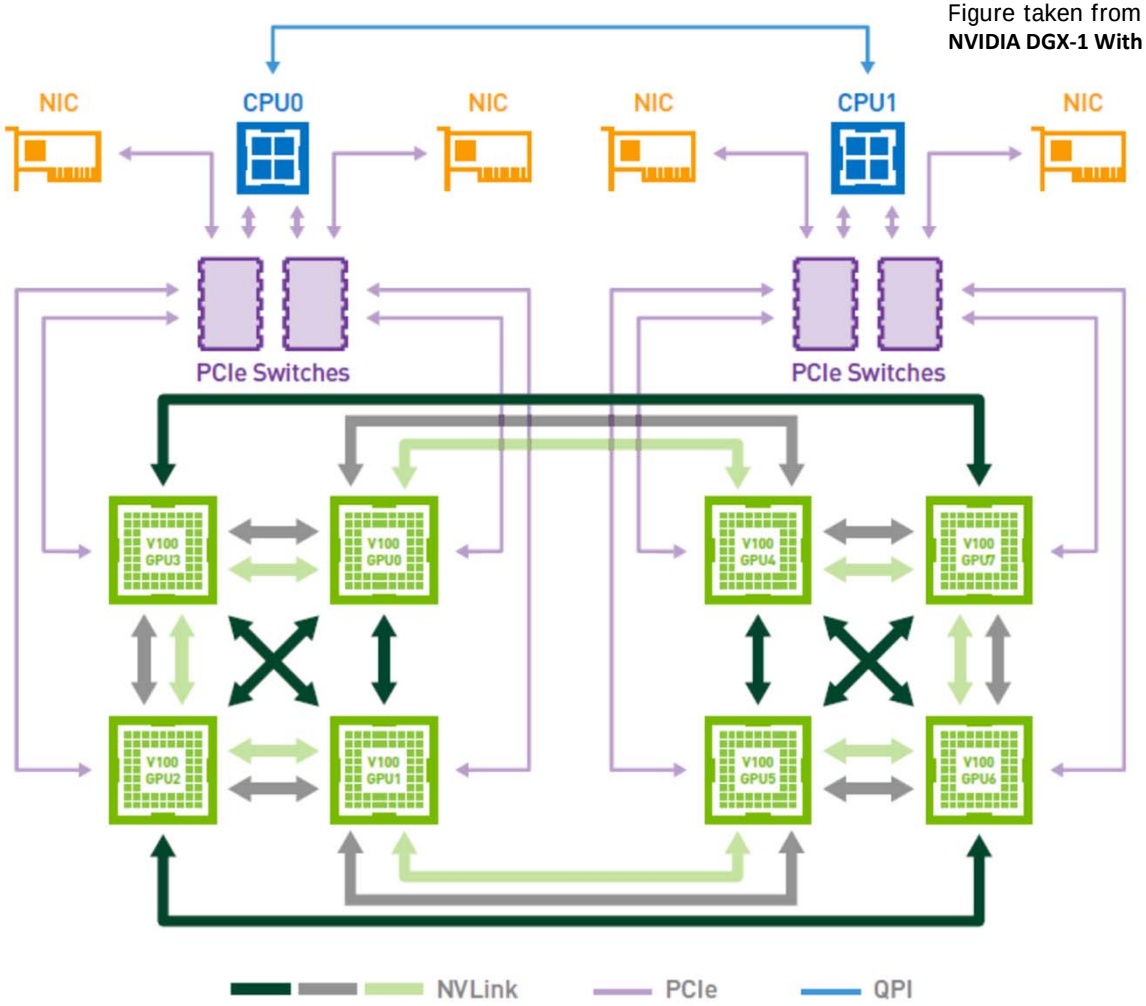
```
Device 1: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6078 MBytes (6373572608 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1759 MHz (1.76 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),
                                                 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  ...
  ...
> Peer access from GeForce GTX 1060 6GB (GPU0) -> GeForce GTX 1060 6GB (GPU1) : Yes
> Peer access from GeForce GTX 1060 6GB (GPU1) -> GeForce GTX 1060 6GB (GPU0) : Yes
```

P2P is crucial for using multi-GPUs via PCIe bus

Nvidia DGX-1 (8 V100+NVLink)

Nvidia DGX-1 (8 V100+NVLink)

Figure taken from the White Paper
NVIDIA DGX-1 With Tesla V100 System Architecture



NVLink 2.0, data rate ~ 300 GB/s

Nvidia DGX-1 (8 V100+NVLink)

- **Connection topology**: with 8 GPUs sitting at the corners of a cube, NVLink only connects the edges and 2 face diagonals.

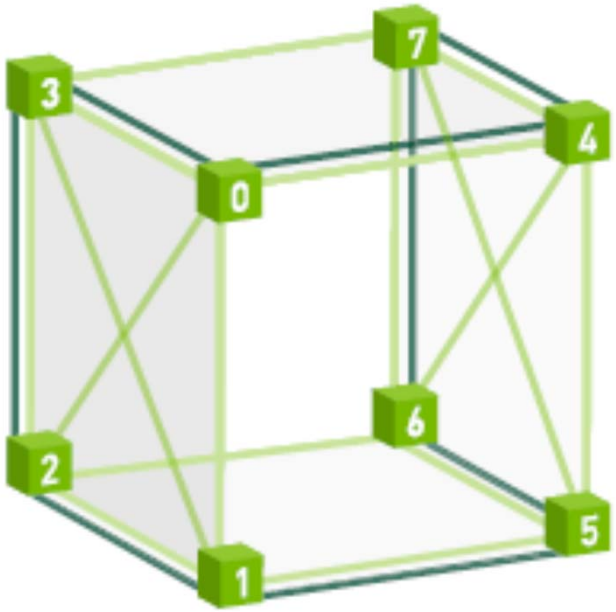


Figure taken from the White Paper
NVIDIA DGX-1 With Tesla V100 System Architecture

deviceQuery (DGX-1)

Detected 8 CUDA Capable device(s)

Device 0: "Tesla V100-SXM2-32GB"

CUDA Driver Version / Runtime Version	10.0 / 10.0
CUDA Capability Major/Minor version number:	7.0
Total amount of global memory:	32480 MBytes (34058272768 bytes)
(80) Multiprocessors, (64) CUDA Cores/MP:	5120 CUDA Cores
GPU Max Clock rate:	1530 MHz (1.53 GHz)
Memory Clock rate:	877 Mhz
Memory Bus Width:	4096-bit
L2 Cache Size:	6291456 bytes
Maximum Texture Dimension Size (x,y,z)	1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
Maximum Layered 1D Texture Size, (num) layers	1D=(32768), 2048 layers
Maximum Layered 2D Texture Size, (num) layers	2D=(32768, 32768), 2048 layers
Total amount of constant memory:	65536 bytes
Total amount of shared memory per block:	49152 bytes
Total number of registers available per block:	65536
Warp size:	32

deviceQuery (cont.)

```
Maximum number of threads per multiprocessor:  2048
Maximum number of threads per block:           1024
Max dimension size of a thread block (x,y,z):  (1024, 1024, 64)
Max dimension size of a grid size              (x,y,z): (2147483647, 65535, 65535)
Maximum memory pitch:                          2147483647 bytes
Texture alignment:                             512 bytes
Concurrent copy and kernel execution:          Yes with 6 copy engine(s)
Run time limit on kernels:                     No
Integrated GPU sharing Host Memory:            No
Support host page-locked memory mapping:       Yes
Alignment requirement for Surfaces:            Yes
Device has ECC support:                        Enabled
Device supports Unified Addressing (UVA):      Yes
Device supports Compute Preemption:            Yes
Supports Cooperative Kernel Launch:            Yes
Supports MultiDevice Co-op Kernel Launch:      Yes
Device PCI Domain ID / Bus ID / location ID:   0 / 21 / 0
Compute Mode:
  < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
...
```

deviceQuery (cont.)

> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU1) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU2) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU3) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU4) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU5) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU6) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU0) -> Tesla V100-SXM2-32GB (GPU7) : Yes
> Peer access from Tesla V100-SXM2-32GB (GPU1) -> Tesla V100-SXM2-32GB (GPU0) : Yes
:
> Peer access from Tesla V100-SXM2-32GB (GPU2) -> Tesla V100-SXM2-32GB (GPU0) : Yes
:
> Peer access from Tesla V100-SXM2-32GB (GPU3) -> Tesla V100-SXM2-32GB (GPU0) : Yes
:
> Peer access from Tesla V100-SXM2-32GB (GPU4) -> Tesla V100-SXM2-32GB (GPU0) : Yes
:
> Peer access from Tesla V100-SXM2-32GB (GPU5) -> Tesla V100-SXM2-32GB (GPU0) : Yes
:
> Peer access from Tesla V100-SXM2-32GB (GPU6) -> Tesla V100-SXM2-32GB (GPU0) : Yes
:
> Peer access from Tesla V100-SXM2-32GB (GPU7) -> Tesla V100-SXM2-32GB (GPU0) : Yes

Summit (The 2nd fastest Supercomputer)

Sponsors	U.S. Department of Energy
Operators	IBM
Architecture	9,216 POWER 9 22-core CPUs 27,648 Nvidia Tesla V100 GPUs
Power	13 MW
Storage	250 PB
Speed	200 petaflops (peak)

Summit or **OLCF-4** is a supercomputer developed by IBM for use at Oak Ridge National Laboratory, which was the fastest supercomputer in the world from June 2018 to November 2019, capable of 200 petaflops.
Its current LINPACK benchmark is clocked at 148.6 petaflops. Summit is the first supercomputer to reach exaop (exa operations per second) speed, achieving 1.88 exaops during a genomic analysis and is expected to reach 3.3 exaops using mixed precision calculations.

A Member of NARLabs
National Center for High-performance Computing

臺灣杉二號（TAIWANIA 2）

2,016 NVIDIA Tesla V100 32GB GPU = 252 units of DGX-1

TOP500 #28 (9 PFLOPS) Green500 #19 (11.285 GFLOPS/W)

CUDA
Compute Unified Device Architecture

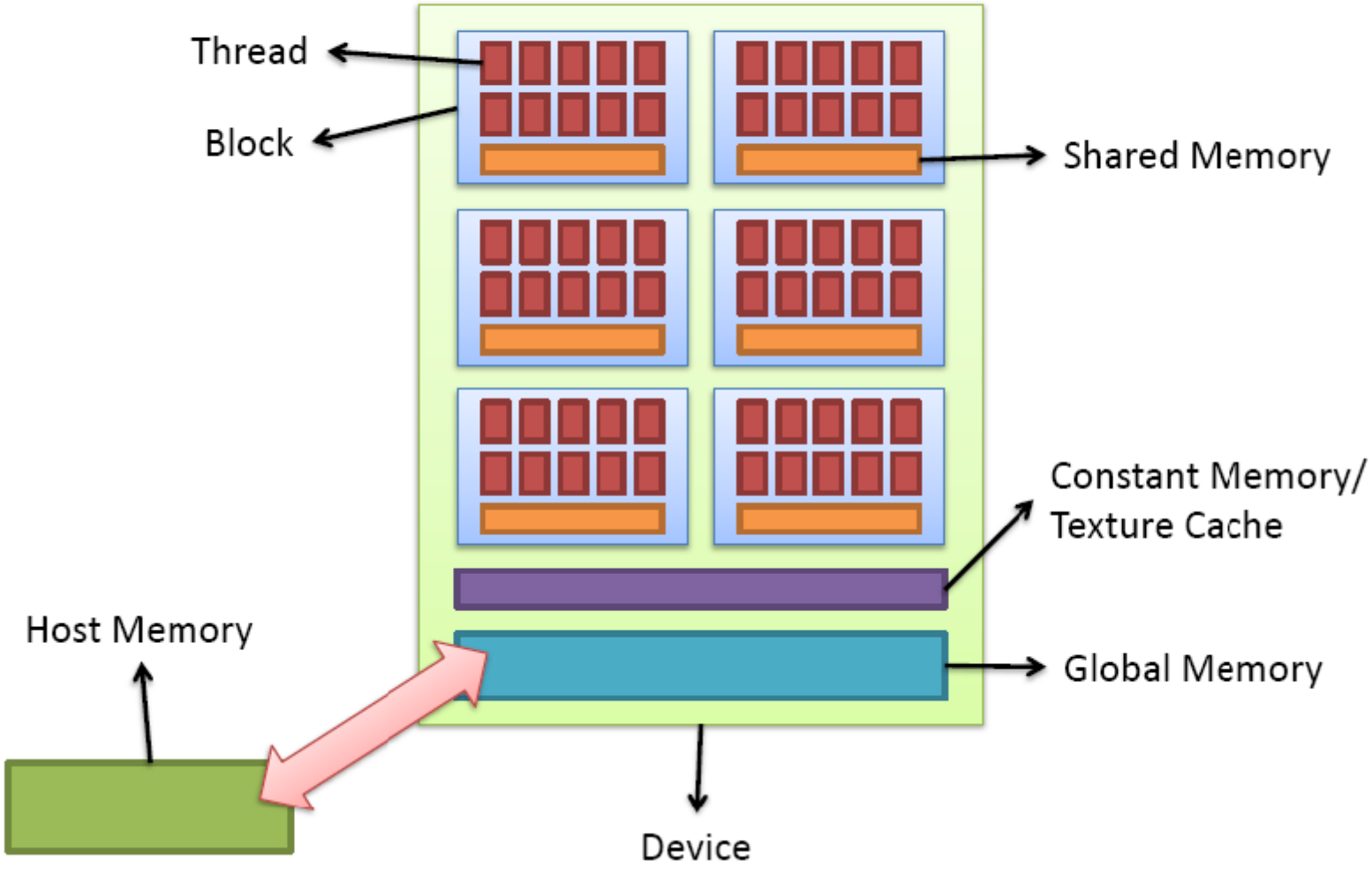
- CUDA is a scalable parallel programming model and software environment designed to develop codes that transparently scales its parallelism to leverage the ever increasing number of cores. CUDA C is just an extension of C.

CUDA Resources

- Nvidia Toolkit Documentation v10.2 (2019)
- Nvidia Developer Resources
- CUDA Programming Guide
- Get Start with CUDA C
- CUDA by Example (code)
- CUDA_by_Example.pdf
- Professional CUDA C programming

https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

CUDA Architecture



Memory Architecture

- Basic features of the Nvidia GPU memory architecture:

	size	access	bandwidth
Global	Large	r/w by all threads and host	Slow
Constant	Small	Read only by all threads	Fast
Texture	Small cache	Read only by all threads	Fast
Shared	Very small	r/w by all threads within one block	Very fast
Register	Very small	r/w by only one thread	Very fast

  - Shared memory may have bank conflict
  - Texture can take care of the locality.
- GPU computing is memory bandwidth bound!

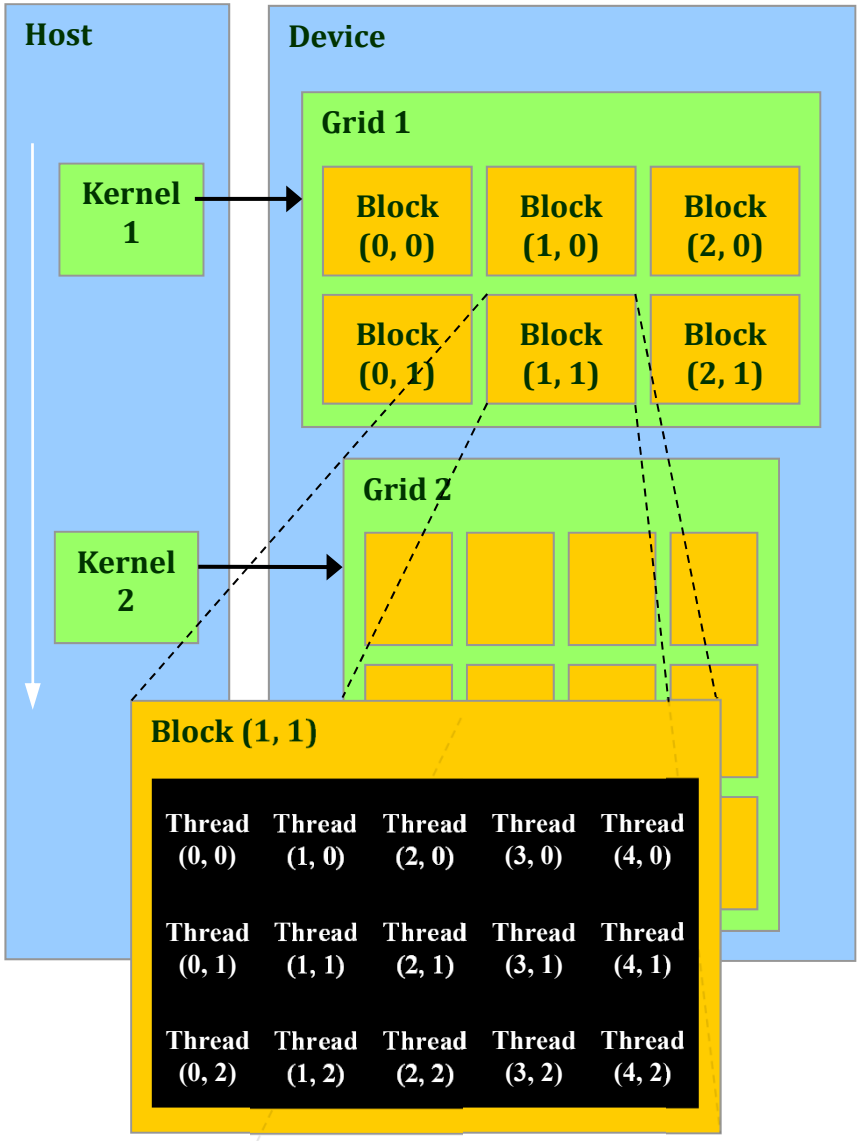
Thread/Block Management

- Parallelize a loop by designating the value of loop counter to each thread.
- Number of threads per block
  - Should be tested to find the best value (may be limited by resource in one block)
  - Must be a multiple of **half-warp**.
- Memory bandwidth bound → Try to **reuse data**.
  - Larger number of blocks does NOT mean better performance!
  - Using loop inside kernel to reduce the number of blocks sometimes runs faster.

CUDA Programming Model

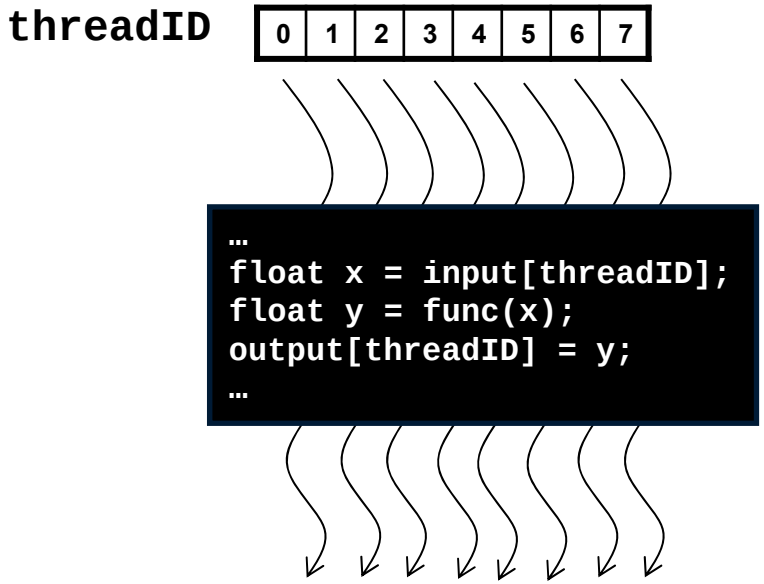
A kernel is executed by a **grid** of thread blocks

- A thread block is a batch of threads that can cooperate with each other by:
  - Sharing data through shared memory
  - Synchronizing their execution
- Threads from different blocks cannot cooperate



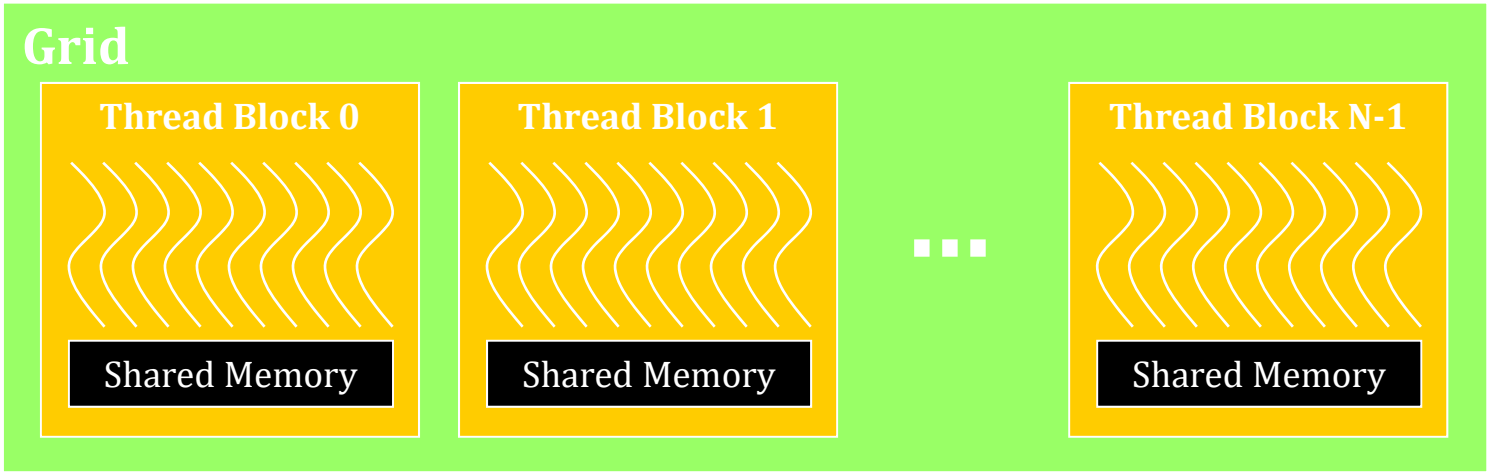
Arrays of Parallel Threads

- **A CUDA kernel is executed by an array of threads**
  - **All threads run the same code**
  - **Each thread has an ID that it uses to compute memory addresses and make control decisions**



Thread Blocks

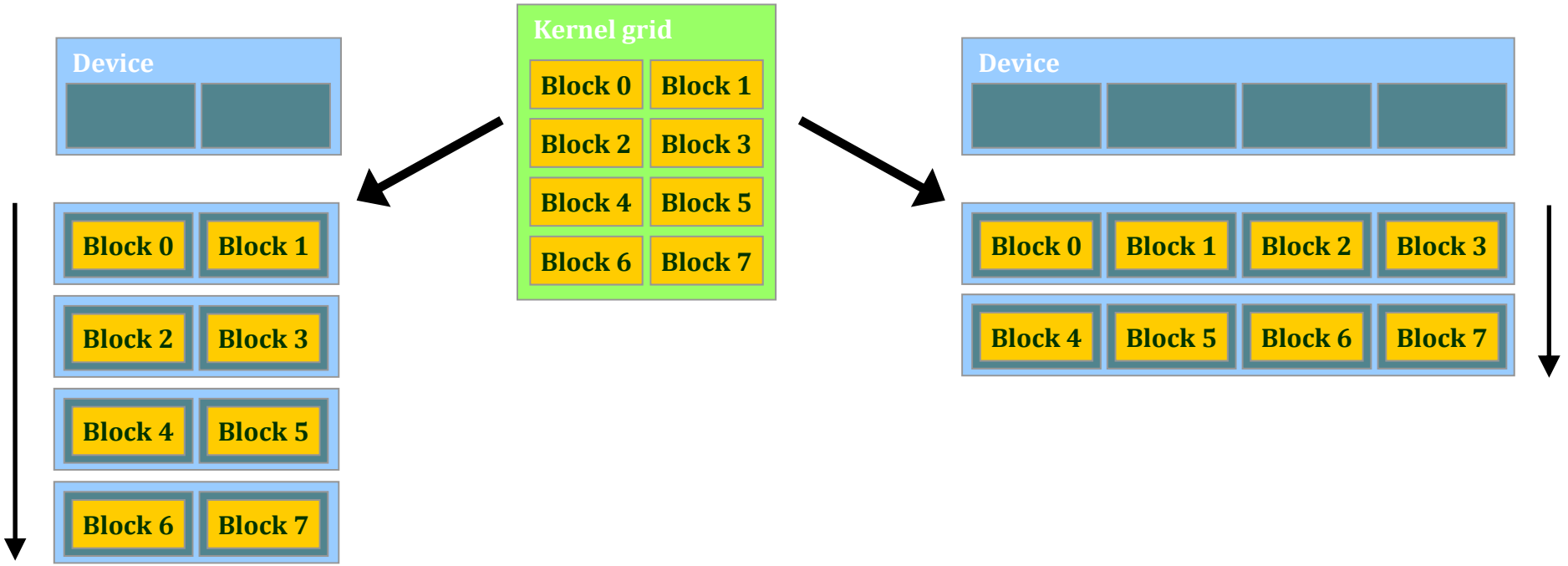
- **Kernel launches a grid of thread blocks**



  - **Threads within a block cooperate via shared memory**
  - **Threads in different block cannot cooperate**
- **Allows programs to *transparently scale* to different GPUs**

Transparent Scalability

- **Hardware is free to schedule thread blocks on any processor**
  - **A kernel scales across parallel multiprocessors**



Salient Features of Threads

Threads within a block can cooperate among themselves by sharing data through some ***shared memory*** *and* ***synchronizing their execution to coordinate memory accesses***. More precisely, one can specify synchronization points in the kernel by calling the **__syncthreads()** intrinsic function; **__syncthreads()** acts as a **barrier** at which all threads in the block must wait before any are allowed to proceed.

Thread blocks are required to execute independently: It must be possible to execute them in any order, in parallel or in series. This independence requirement allows thread blocks to be scheduled in any order across any number of cores, enabling us to write scalable code.

Thread Hierarchy

threadIdx has 3 components : threadIdx.x, threadIdx.y, threadIdx.z

A thread block containing threads, which can be 1, 2, or 3 dimensional

blockDim has 3 components : blockDim.x, blockDim.y, blockDim.z

blockIdx has 3 components : blockIdx.x, blockIdx.y, blockIdx.z

```
__global__ void matAdd(float *A, float *B, float *C)
{
    int i = threadIdx.x;      // assume one single block in 2D grid
    int j = threadIdx.y;
    C[i][j] = A[i][j] + B[i][j];
}

int main(void)
{
   dim3 dimBlock(N, N);
   matAdd<<<1, dimBlock>>>(A, B, C);
}
```

```
__global__ void matAdd(float *A, float *B, float *C, int N)
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int j = blockIdx.y * blockDim.y + threadIdx.y;
   if (i < N && j < N)
      C[i][j] = A[i][j] + B[i][j];
}

int main()
{
   int  N=1024;
   dim3 dimBlock(16, 16);
   dim3 dimGrid((N + dimBlock.x – 1) / dimBlock.x,
                        (N + dimBlock.y – 1) / dimBlock.y);
    matAdd<<<dimGrid, dimBlock>>>(A, B, C, N);
}
```

deviceQuery (GTX1060)

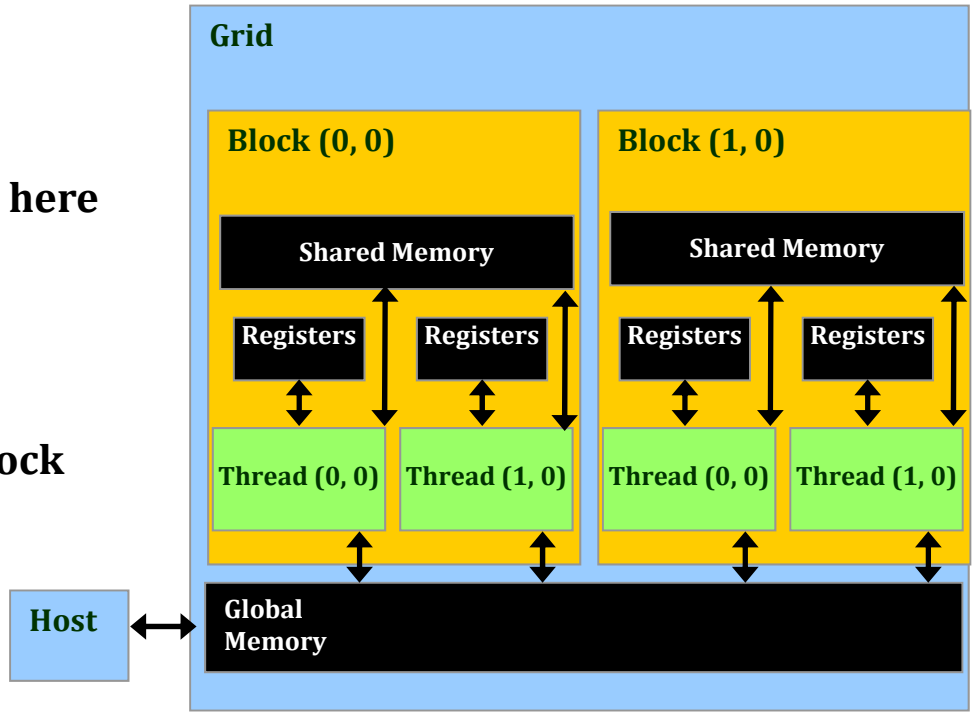
```
Device 0: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          10.2 / 10.2
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6078 MBytes (6373179392 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1759 MHz (1.76 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),
                                                 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  ...
  ...
```

Execution Model

- **Kernels are launched in grids**
- **A block executes on one multiprocessor**
  - **Does not migrate**
- **Several blocks can reside concurrently on one multiprocessor**
  - **The number of blocks is limited by the resources of one multiprocessor**
    - **Registers are partitioned among all resident threads**
    - **Shared memory is partitioned among all resident thread blocks**

CUDA Memory Hierarchy

- **Registers**
- **Global Memory**
  - **Kernel input and output data reside here**
  - **Off-chip, large**
  - **Uncached**
- **Shared Memory**
  - **Shared among threads in a single block**
  - **On-chip, small**
  - **As fast as registers**



The host can read & write global memory but not shared memory

Global, constant, and texture memory spaces are persistent across kernels called by the same application.

CUDA Model Summary

- **Thousands of concurrent threads**
- **Shared memory**
  - **User-managed data cache**
  - **Thread communication / cooperation within blocks**
- **Access to global memory**
  - **Any thread can read/write any location(s)**

Memory	Location	Cached	Access	Scope
Shared	On-chip	N/A	Read/write	All threads in a block
Global	Off-chip	No	Read/write	All threads + host

General Rules for Optimization

- **Optimize memory transfers**
  - **Minimize memory transfers from host to device**
  - **Use shared/constant/texture memory as cache to device memory**
  - **Take advantage of coalesced memory access**
- **Maximize processor occupancy**
  - **Optimize execution configuration (block size)**
- **Maximize computation intensity**
  - **More computation per memory access**
  - **Re-compute may be faster than re-loading data**