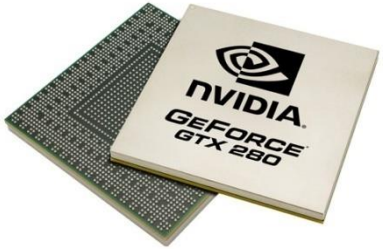
CUDA 硬體介紹

Graphic Processor Unit



Traditional Graphics Pipeline

- 傳統的繪圖管線(nvidia G80之前的GPU)
  - vertex shader和pixel shader的比例是固定的，但需求比例不盡相同，造成閒置的情形無法充份利用processors

Unified Shader Pipline

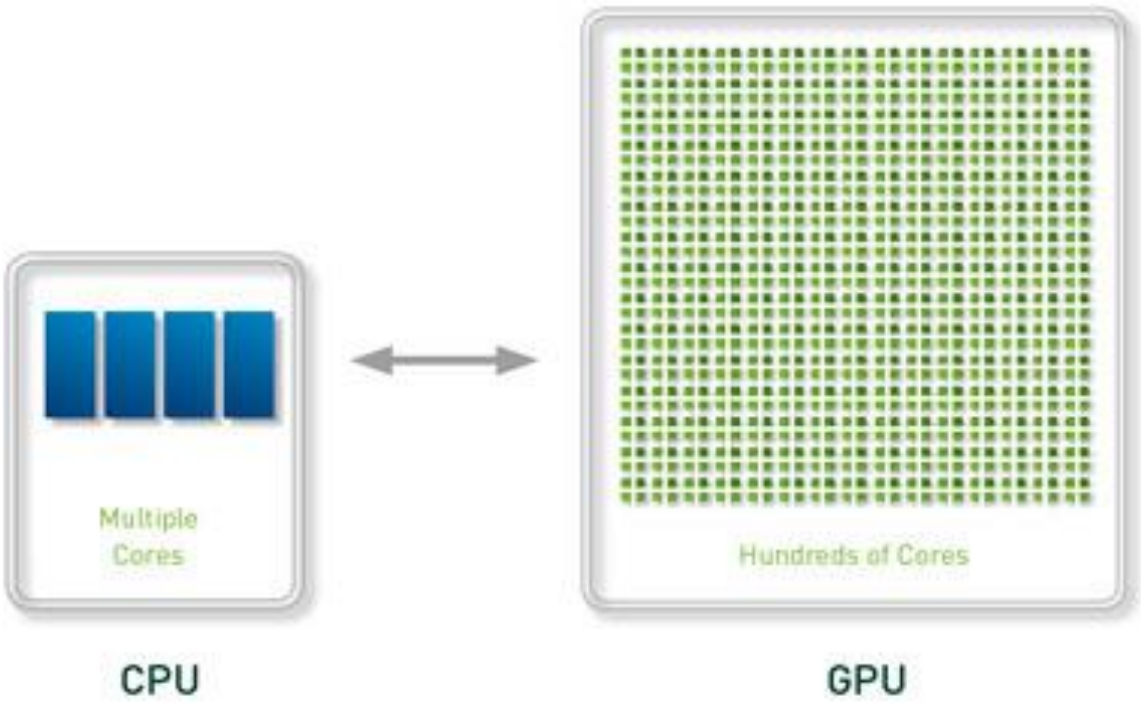
- 2007年後(nvidia G80)有統一化繪圖管線(Unified Shader Pipline)設計產生
  - Unified Shader 依不同比例需求配置給processors，避免閒置情況發生

CUDA - GPU

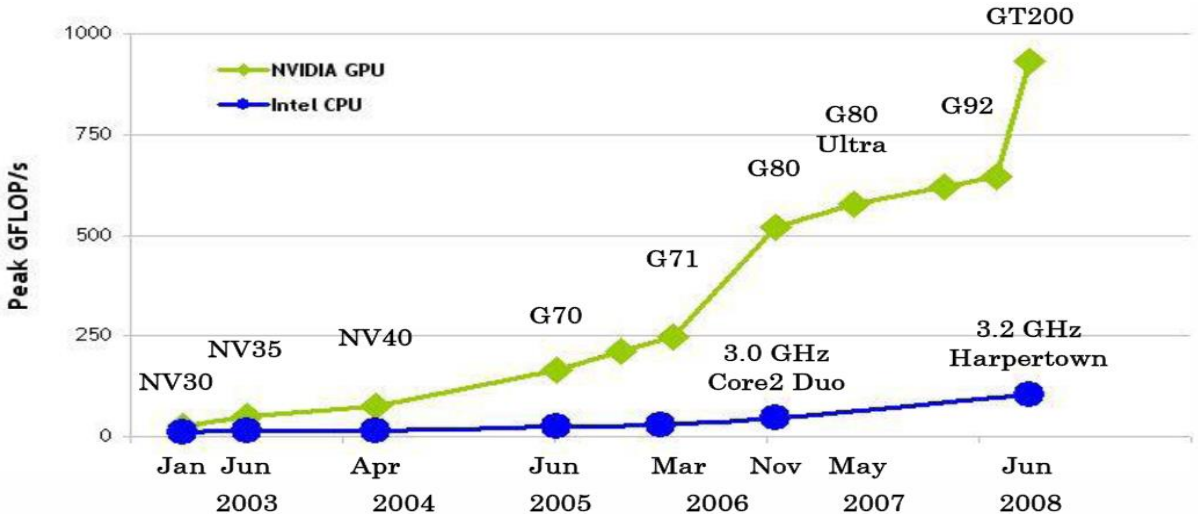
- NVIDIA公司目前已支援CUDA運算架構的GPU硬體設備
  - Tesla、Quadro、GeForce
  - 科學計算、專業繪圖及3D視覺化、遊戲

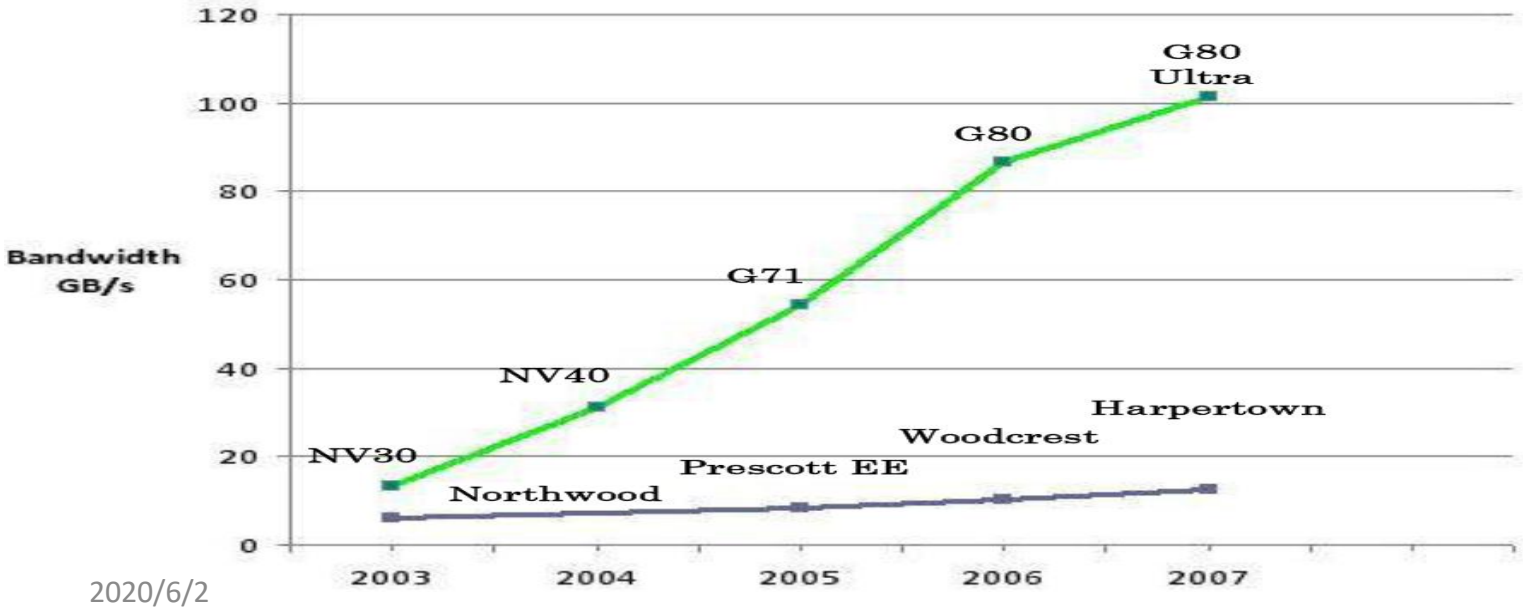
- 大量運算核心設備

Architectures



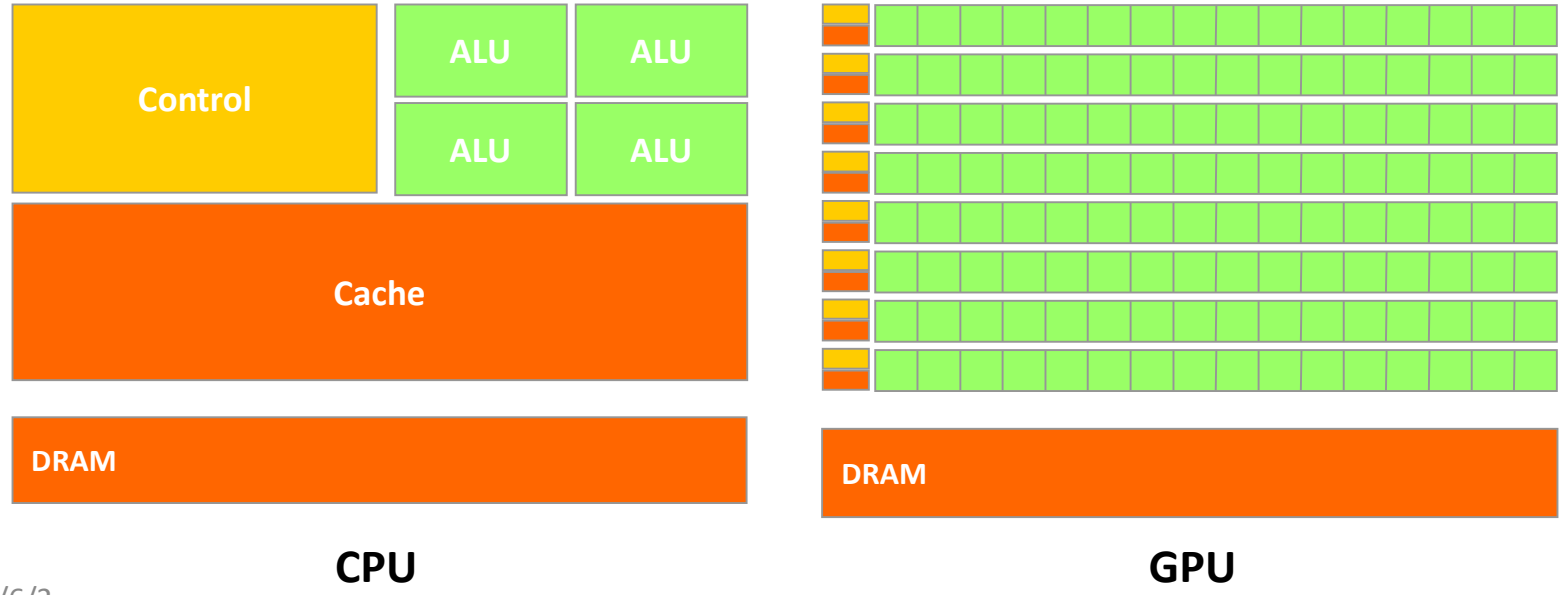
CPU 和GPU 運算能力和記憶體頻寬演進





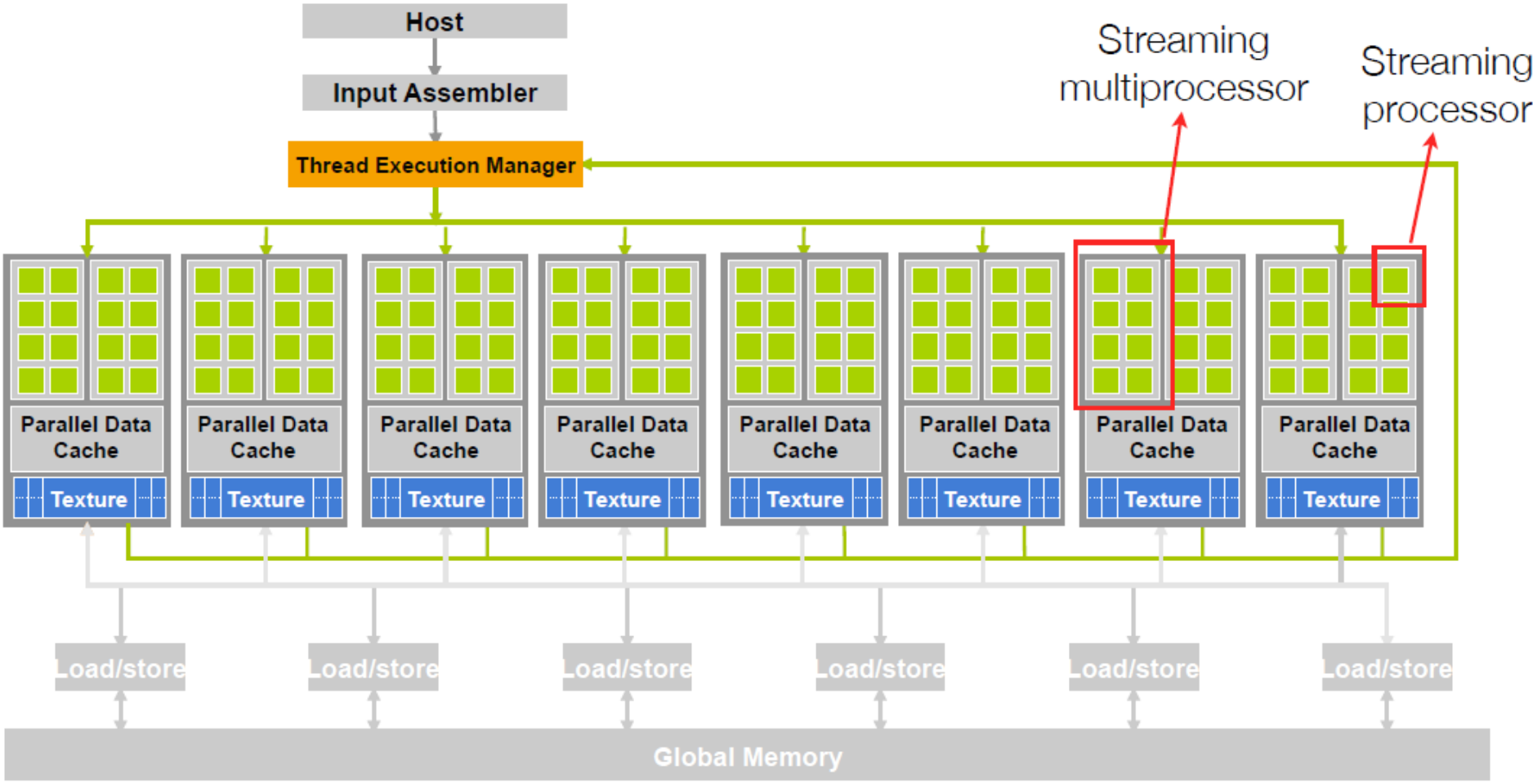
為何會如此演進?

- GPU 架構是為密集運算、高度平行化的資料設計。
  - 可以有更多的電晶體用於運算而非快取或流程控制。



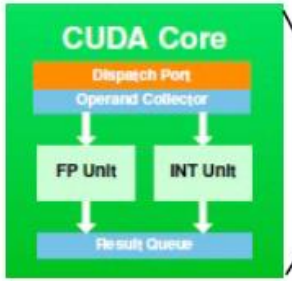
```
kcf@kcf-desktop:/usr/local/cuda-10.0/samples/1_Utilities/deviceQuery$ ./deviceQuery
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA Tegra X1"
  CUDA Driver Version / Runtime Version          10.0 / 10.0
  CUDA Capability Major/Minor version number:    5.3
  Total amount of global memory:                 3963 MBytes (4155600896 bytes)
  ( 1) Multiprocessors, (128) CUDA Cores/MP:     128 CUDA Cores
  GPU Max Clock rate:                            922 MHz (0.92 GHz)
  Memory Clock rate:                             1600 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 262144 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 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: 32768
  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
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            No
  Supports Cooperative Kernel Launch:            No
  Supports MultiDevice Co-op Kernel Launch:      No
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.0, CUDA Runtime Version = 10.0, NumDevs = 1
Result = PASS
```

A glance at the GeForce 8800

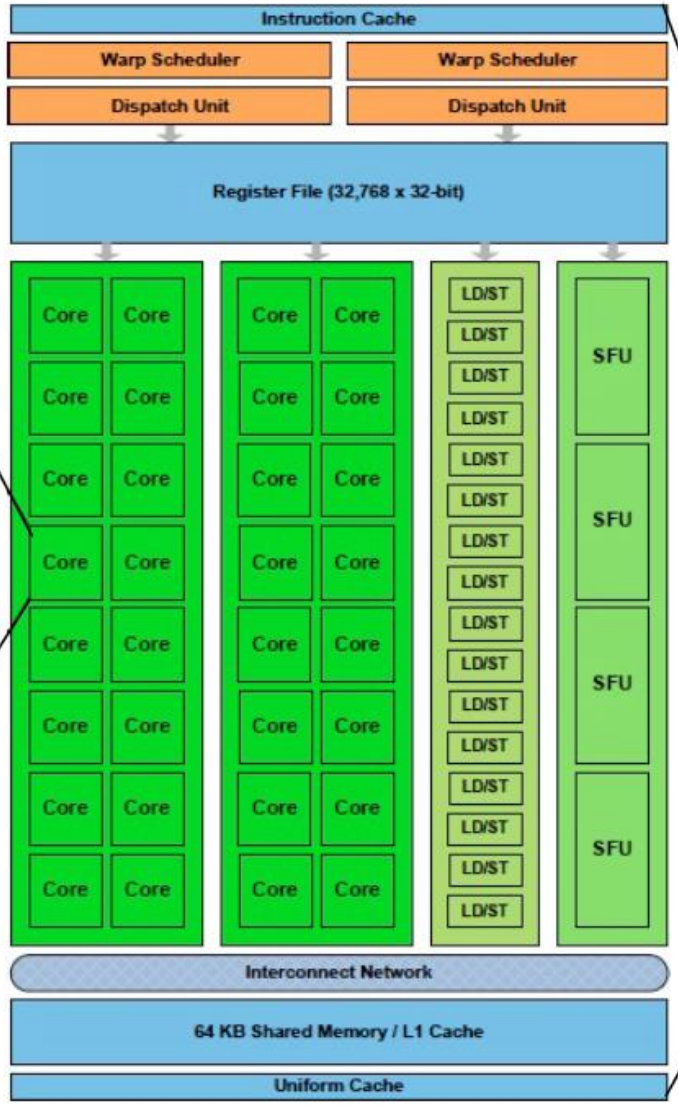


GPU Chip

- Cores?

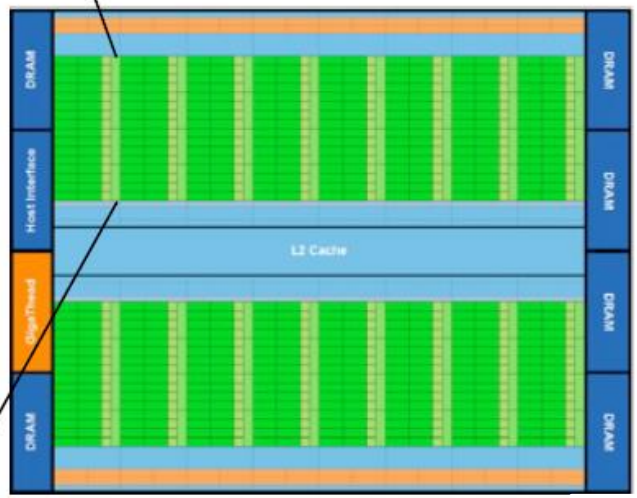


SP



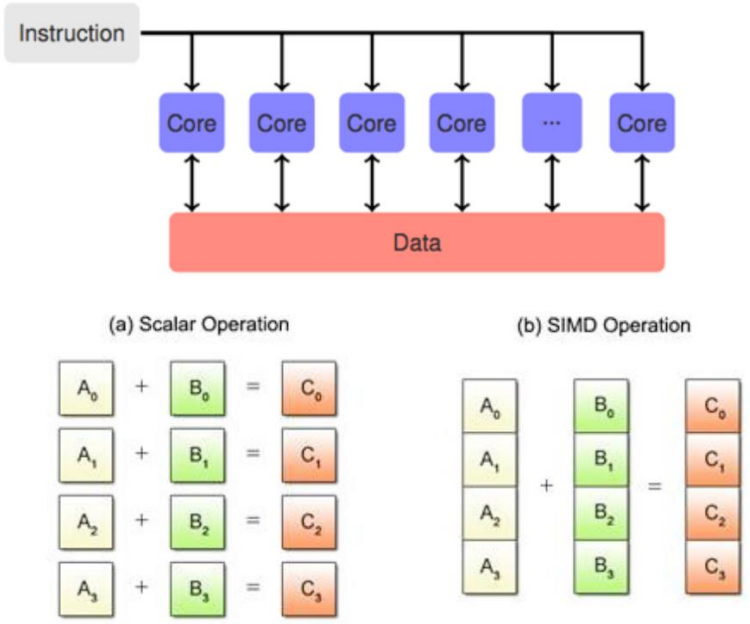
SM

3 billion transistors
512 CUDA cores (32 in 16 SMs)
64kB of on chip RAM
High bandwidth



GPU Chip

- The GPU core is the **stream processor**
- Stream processors are grouped in **stream multiprocessors**
  - SM is basically a **SIMD** processor (Single Instruction Multiple Data)



GPU chip design

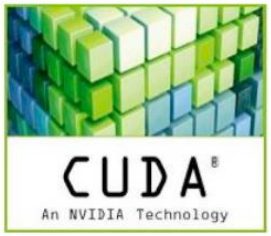
- Core ideas
  - GPUs consist of many “simple” cores
  - Designed for many simpler tasks: **high throughput**
  - Fewer control units: **latency**

Maximize throughput of all threads

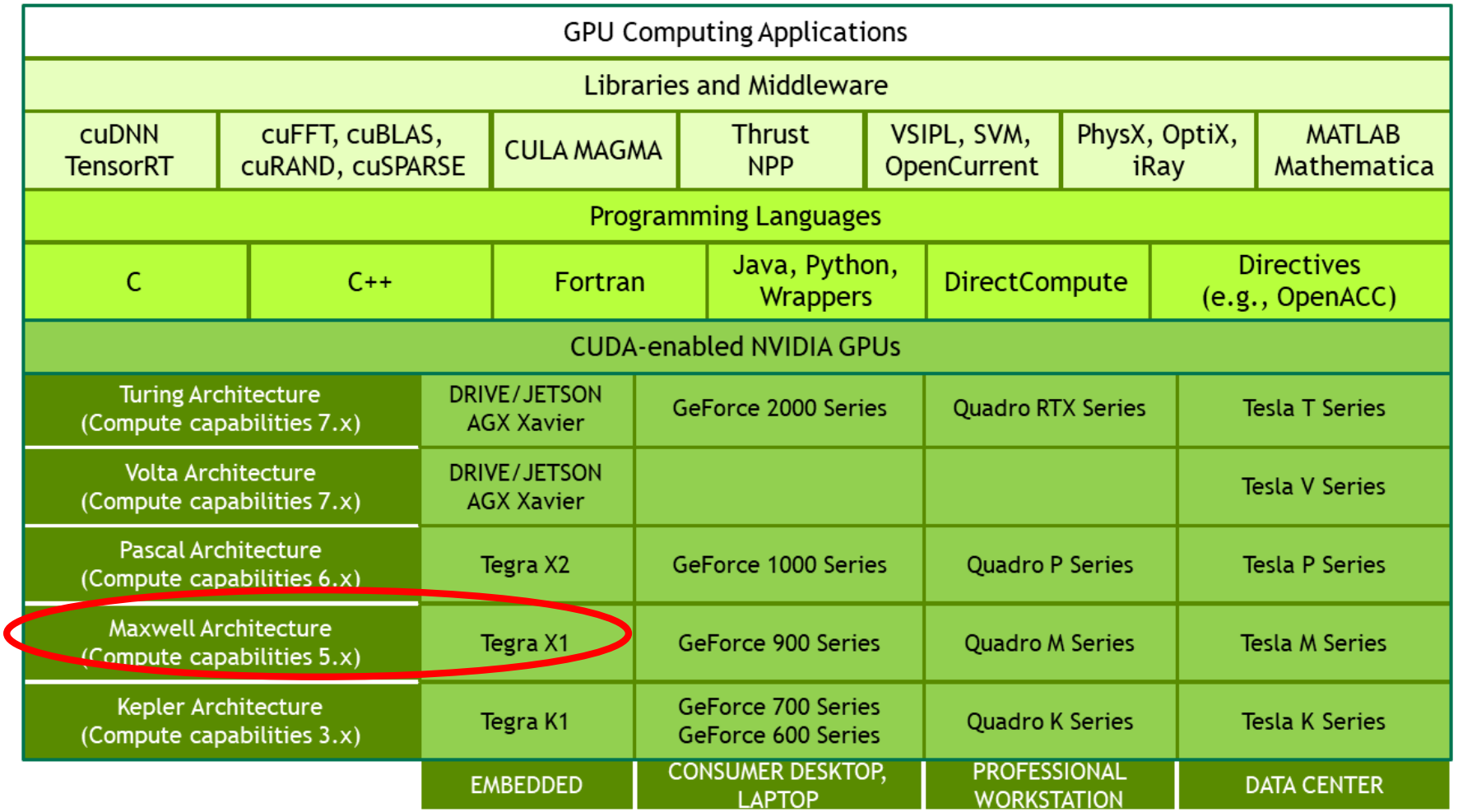
Minimize latency of a thread

Computer Unified Device Architecture (CUDA)

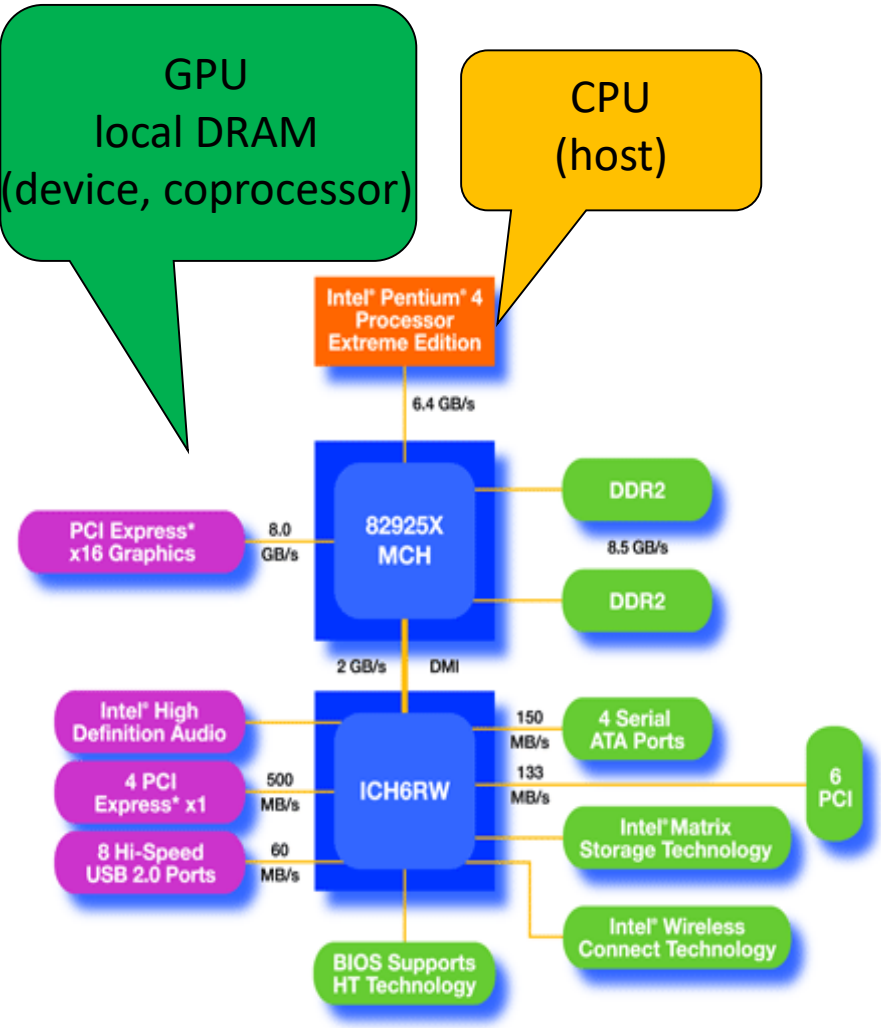
- Parallel computer architecture developed by NVIDIA
- General purpose programming model
- Specially designed for General Purpose GPU (GPGPU) computing:
  - Offers a compute designed API
  - Explicit GPU memory managing



A General-Purpose Parallel Computing Platform and Programming Model



CPU v.s GPU



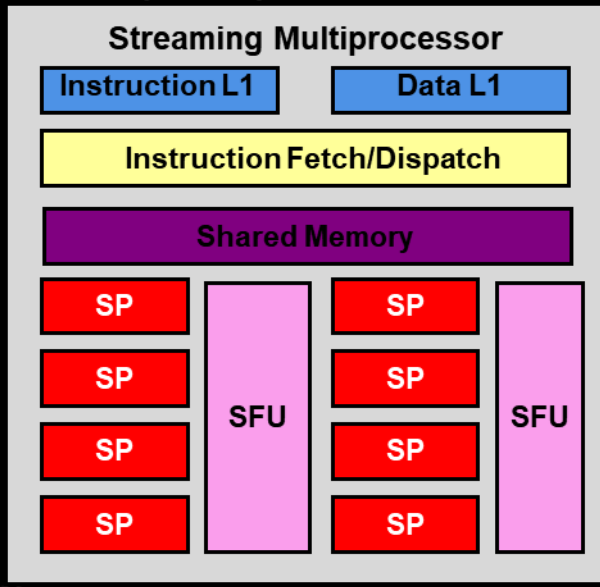
- Integrated programming model
- High speed data transfer(Host ←→ device)
  - up to 5 GB/sec (PCI-E gen.2 x16)
- High data transferring rate
  - Device memory ← →GPU (100GB/s)
  - Host memory ←→CPU(10GB/s)
- Asynchronous data transfer
- Large GPU memory systems

GPU運作細部討論

nVidia GPU Architecture

- GPU最基本的處理單位是*SP(Streaming Processor)*
- GPU擁有許多SP同時進行運算
- 數個SP附加其他單元組成*SM(Streaming Multiprocessor)*

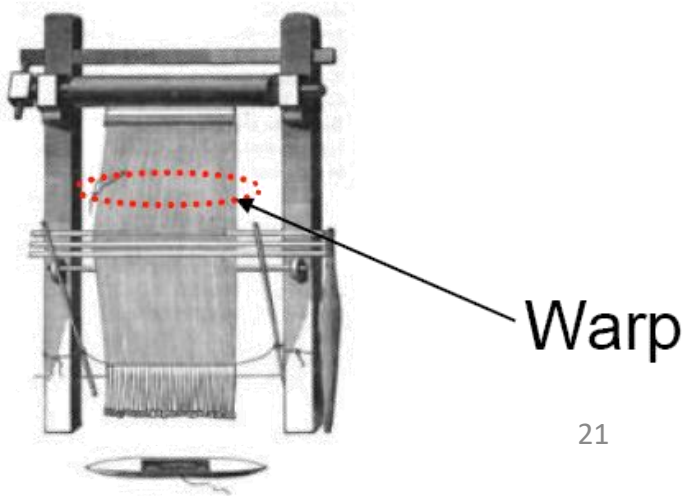
```
 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 1050 Ti"
  CUDA Driver Version / Runtime Version          8.0 / 8.0
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 4096 MBytes (4294967296 bytes)
  ( 6) Multiprocessors, (128) CUDA Cores/MP:     768 CUDA Cores
  GPU Max Clock rate:                            1392 MHz (1.39 GHz)
  Memory Clock rate:                             3504 Mhz
  Memory Bus Width:                              128-bit
  L2 Cache Size:                                 1048576 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
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  CUDA Device Driver Mode (TCC or WDDM):         WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA Runtime Version = 8.0, NumDevs = 1, Device0 = GeForce
 GTX 1050 Ti
Result = PASS
```



```
kcf@kcf-desktop:/usr/local/cuda-10.0/samples/1_Utilities/deviceQuery$ ./deviceQuery
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA Tegra X1"
  CUDA Driver Version / Runtime Version          10.0 / 10.0
  CUDA Capability Major/Minor version number:    5.3
  Total amount of global memory:                 3963 MBytes (4155600896 bytes)
  ( 1) Multiprocessors, (128) CUDA Cores/MP:     128 CUDA Cores
  GPU Max Clock rate:                            922 MHz (0.92 GHz)
  Memory Clock rate:                             1600 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 262144 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 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: 32768
  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
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            No
  Supports Cooperative Kernel Launch:            No
  Supports MultiDevice Co-op Kernel Launch:      No
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.0, CUDA Runtime Version = 10.0, NumDevs = 1
Result = PASS
```

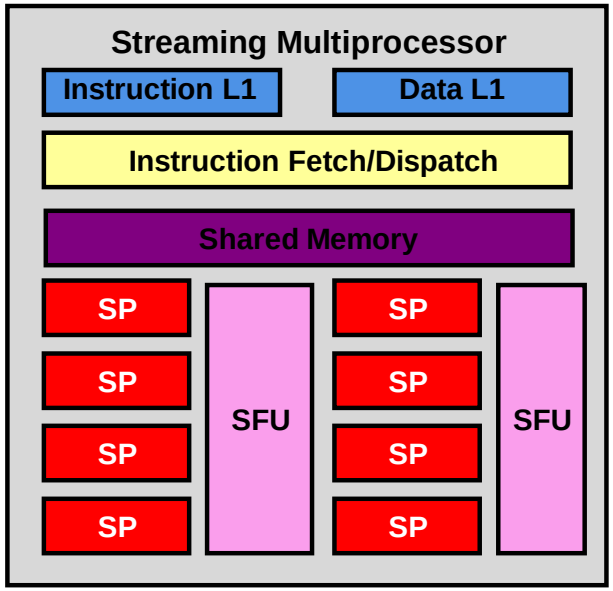
SM, block, warp

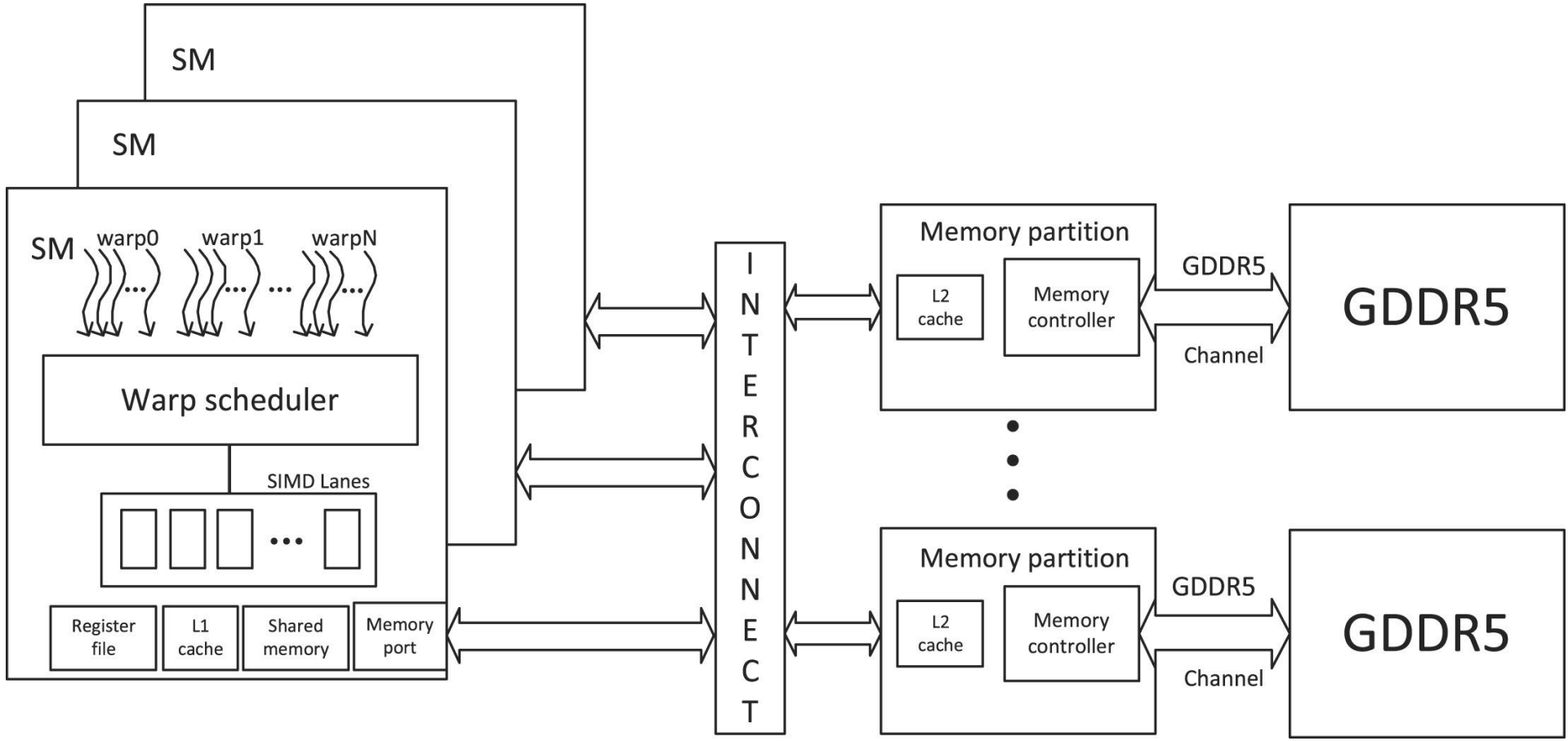
- *SM(Streaming Multiprocessor)*
  - GPU硬體運算時會把CUDA中的多個thread block分配給SM進行，由Warp scheduler負責排程
- block(thread block)
  - Thread block中實際進行運算的threads以*warp* 為單位一起執行同一指令 (32 threads execute one instruction simultaneously.)
- warp
  - 1 warp = 32 threads



Streaming Multiprocessor (SM)

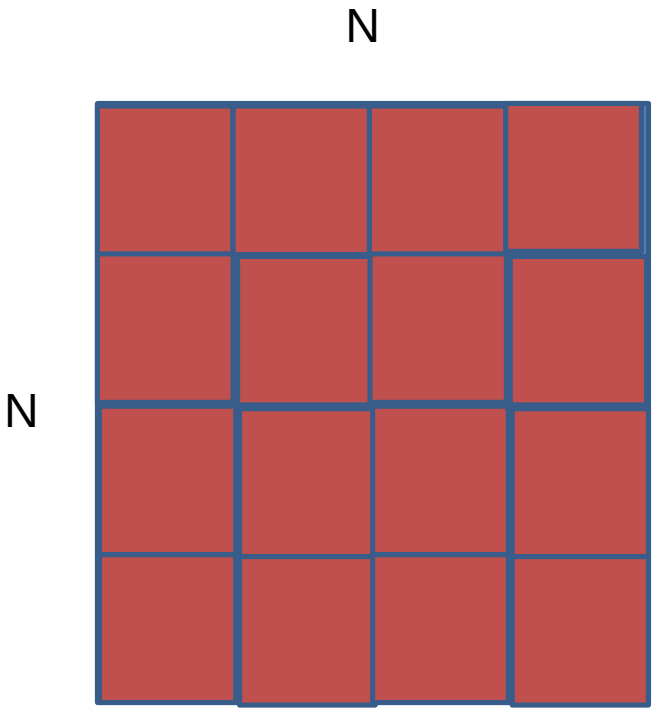
- Streaming Multiprocessor (SM)
  - 8 Streaming Processors (SP)(依據型號不同位有差異)
  - 2 Special Function Units (SFU)
  - 一個SM類似一個CPU，有其執行必要之資源。一個SP中只有整數與浮點數計算單元。
- Multi-threaded instruction dispatch
  - 1 to 512 threads active
  - Shared instruction fetch per 32 threads
  - Cover latency of memory loads
- Registers (32768Reg/Block)
- Shared memory (49152B/Block)
- DRAM access

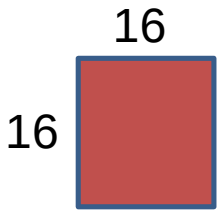


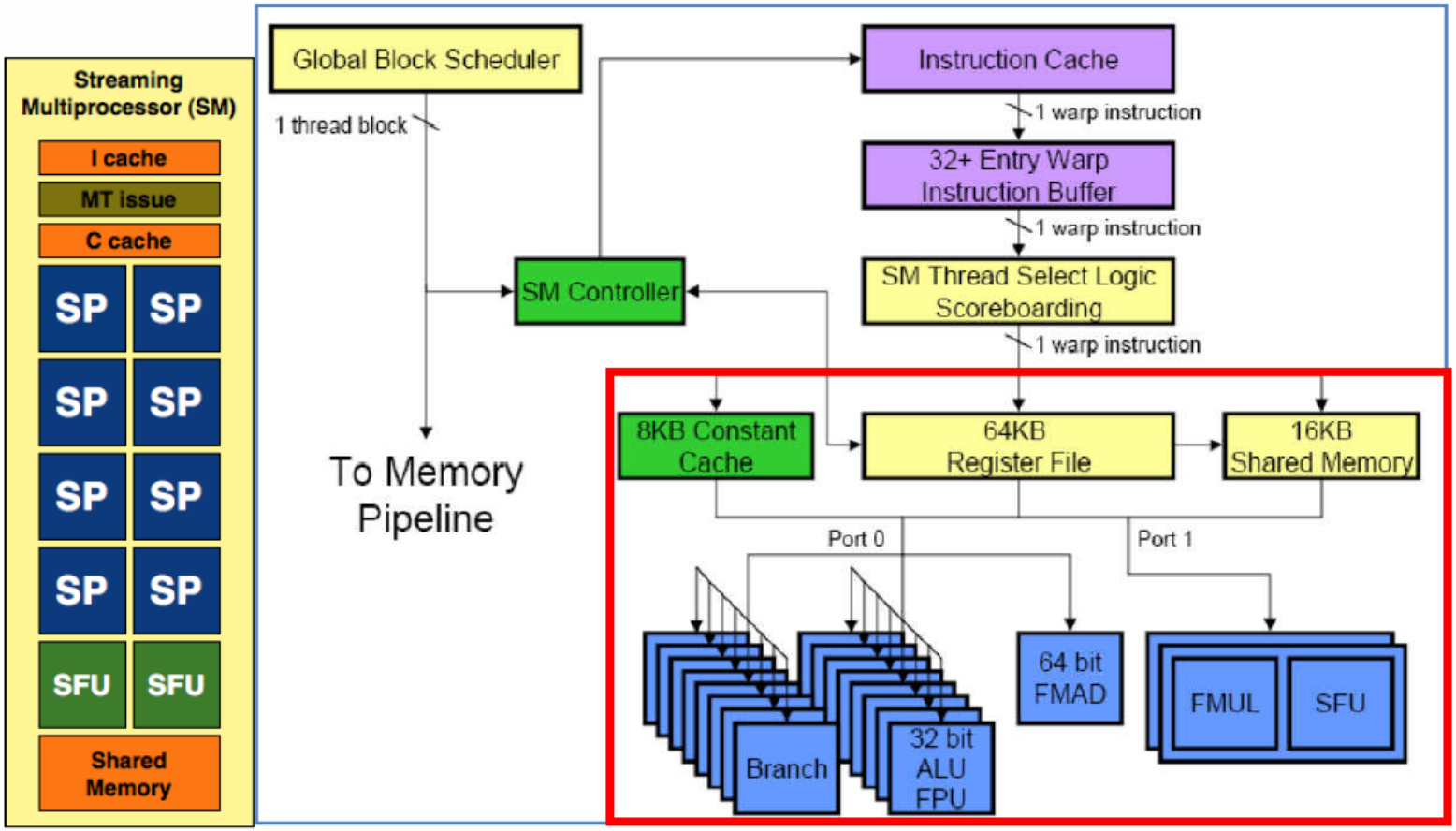

SM
SM
SM
warp0
warp1
warpN
Warp scheduler
SIMD Lanes
Register file
L1 cache
Shared memory
Memory port
INTERCONNECT
Memory partition
L2 cache
Memory controller
GDDR5
Channel
GDDR5
Memory partition
L2 cache
Memory controller
GDDR5
Channel
GDDR5

Sample code

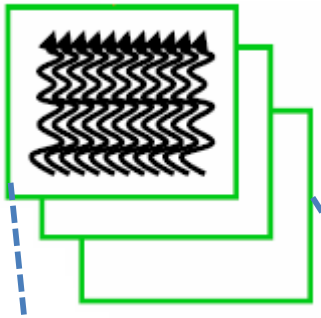
```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int j = blockIdx.y * blockDim.y + threadIdx.y;
   if (i < N && j < N)
      C[i][j] = A[i][j] + B[i][j];
}

int main()
{
   ...
   // Kernel invocation
   dim3 threadsPerBlock(16, 16);    //dim3 是 CUDA 的一種資料型態，表示一個 3D 的向量
   dim3 numBlocks(N / threadsPerBlock.x, N / threadsPerBlock.y);
   MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
   ...
}
```



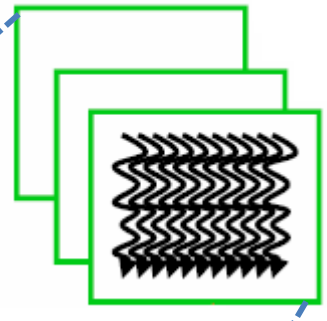



Streaming Multiprocessor (SM)
I cache
MT issue
C cache
SP
SP
SP
SP
SP
SP
SP
SP
SFU
SFU
Shared Memory
Global Block Scheduler
1 thread block
To Memory Pipeline
SM Controller
Instruction Cache
1 warp instruction
32+ Entry Warp Instruction Buffer
1 warp instruction
SM Thread Select Logic Scoreboarding
1 warp instruction
8KB Constant Cache
64KB Register File
16KB Shared Memory
Port 0
Port 1
Branch
32 bit ALU FPU
64 bit FMAD
FMUL
SFU

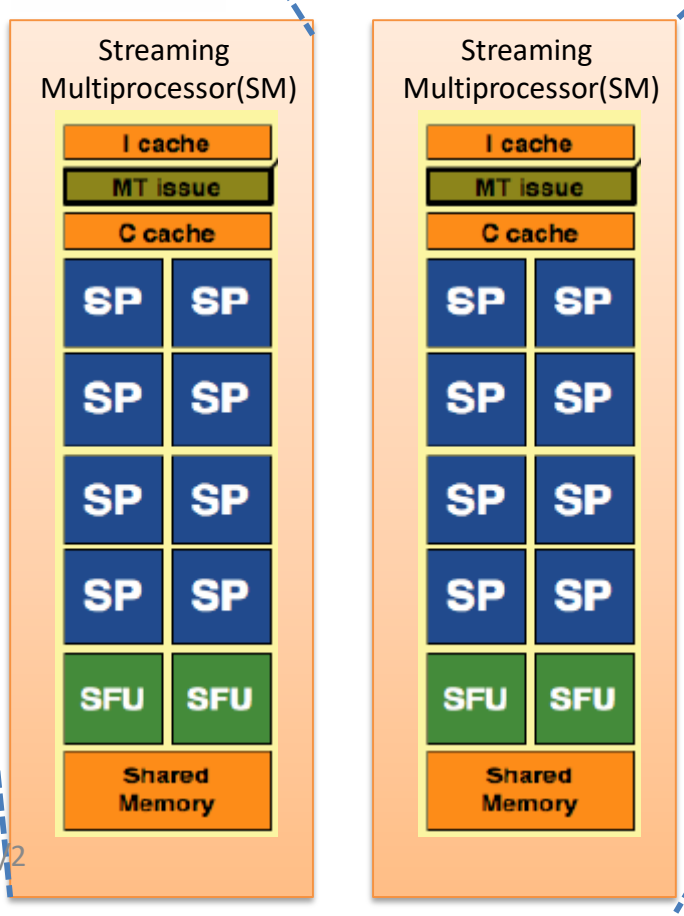
SM Executes Blocks



Blocks



Blocks



多個thread block同時被分配給一個SM，每個block中的thread又以32個為一個warp，作為執行時的基本單位

SM Executes Blocks

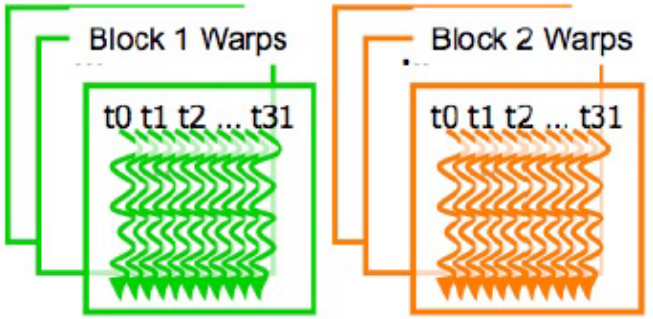
- Threads are assigned to SMs in Block granularity
  - More than one block assigned to each SM as resource allows(block排程)
  - SM in CC 1.2 can take up to 1024 threads
    - Could be 512 (threads/block) * 2 blocks
    - Or 256(threads/block) * 4 blocks, etc

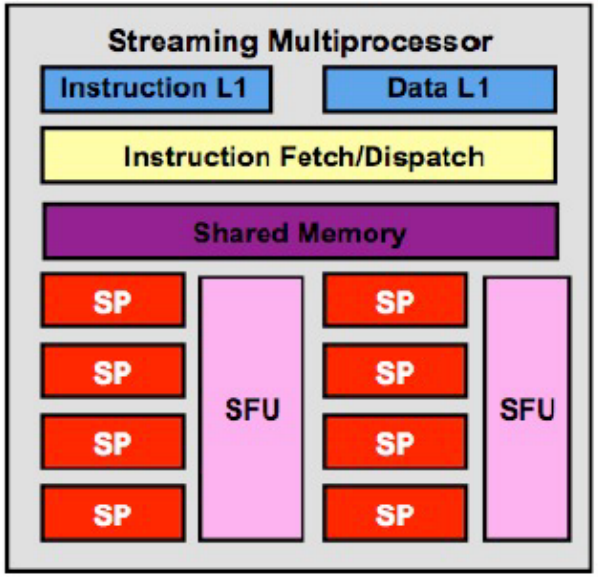
    1024個thread可以分成不同個數的block
- Threads run concurrently
  - SM assigns/maintains thread id
  - SM manages/schedules thread execution

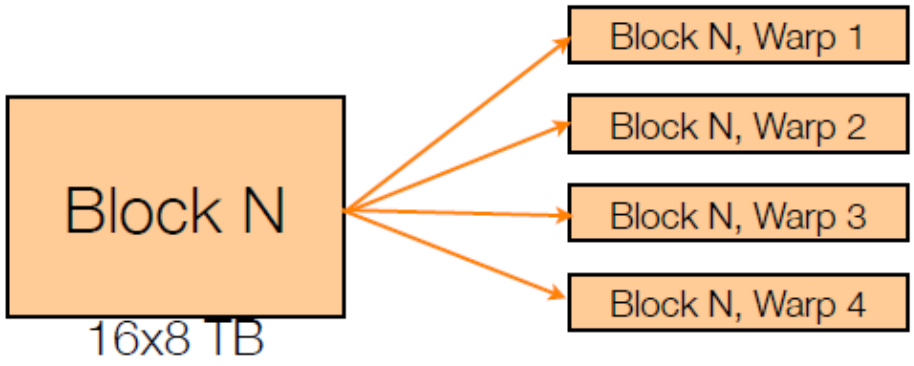
Thread的執行由SM調度管理

Warp designation

- Hardware separates threads of a block into warps
  - All threads in a warp correspond to the same thread block
  - Threads are placed in a warp sequentially
  - Threads are scheduled in warps



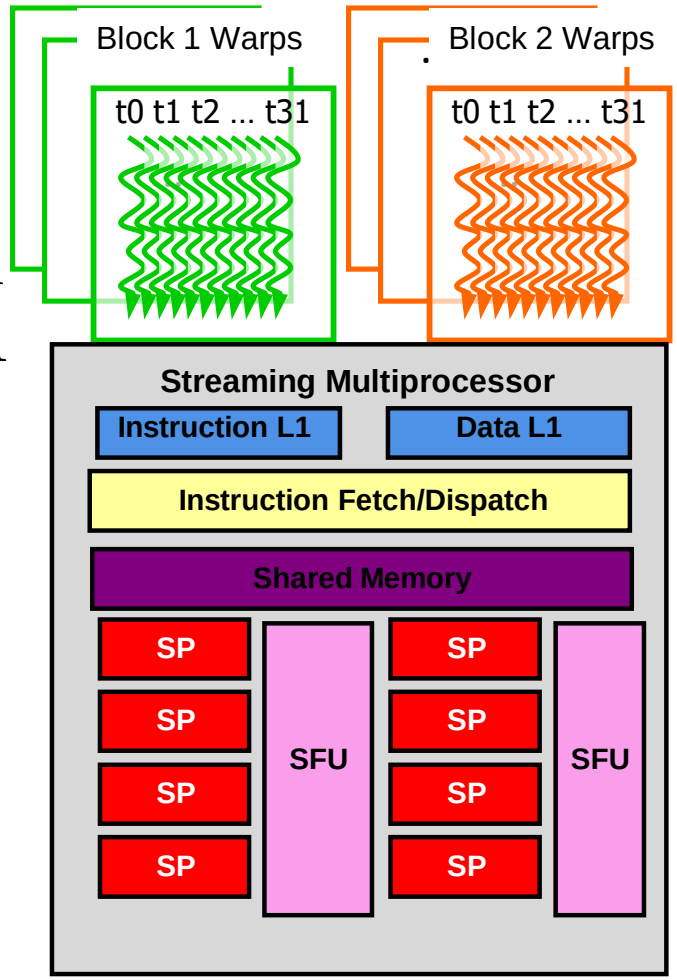




16*8=32*4

Thread Scheduling/Execution(1/3)

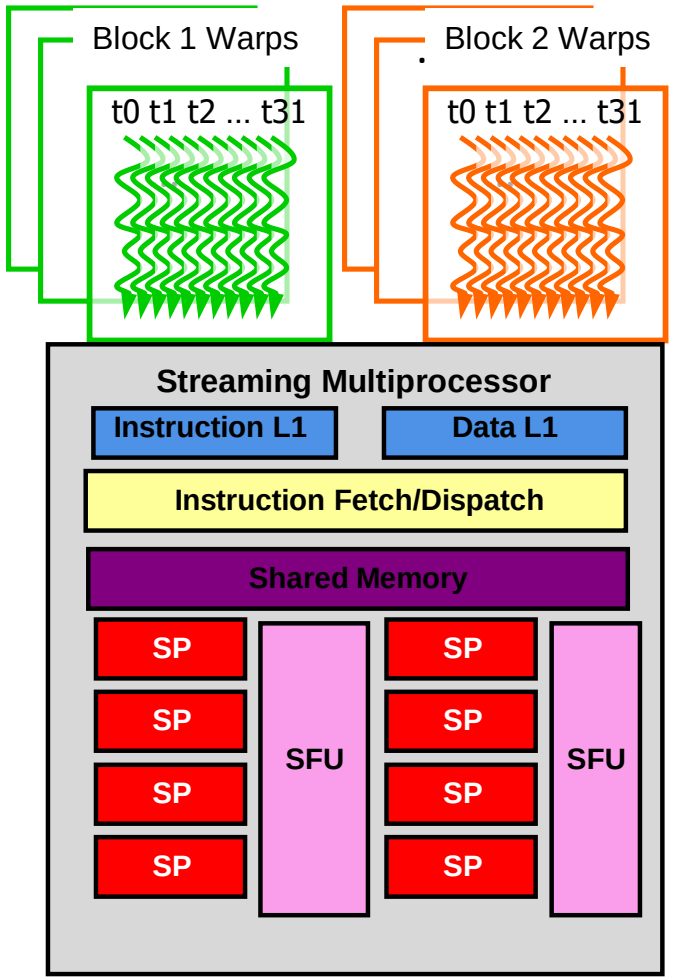
- Each Thread Blocks is divided in 32-thread Warps
  - This is an implementation decision, not part of the CUDA programming model
- Warps are scheduling units in SM

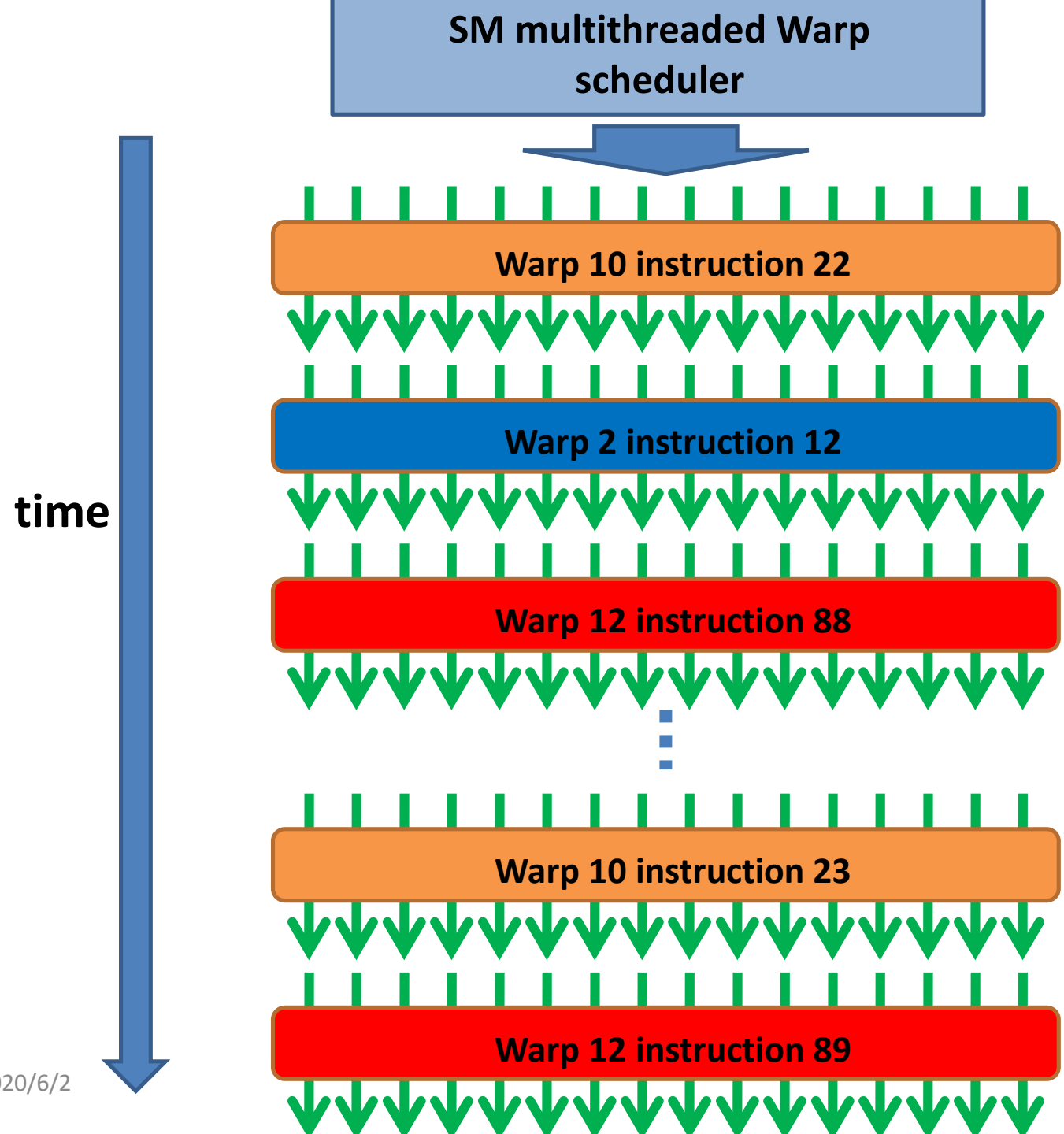


32個執行緒為一個執行單位，稱之為wrap

Thread Scheduling/Execution(2/3)

- If 2 thread blocks are assigned to an SM and each block has 512 threads, how many Warps are there in an SM?
  - Each Block is divided into 512/32 = 16 Warps
  - There are 16 * 2 = 32 Warps
  - At any point in time, only one of the 32 Warps will be selected for instruction fetch and execution.




SM multithreaded Warp scheduler
time
Warp 10 instruction 22
Warp 2 instruction 12
Warp 12 instruction 88
Warp 10 instruction 23
Warp 12 instruction 89

Warp(1/2)

- SM會以連續的方式對warp進行分組
  - 例如：一個thread block裡有128個threads，將會被分為四組warp (if 128 threads are assigned to a thread block)
    - Warp 1：0 – 31
    - Warp 2：32 – 63
    - Warp 3：64 – 95
    - Warp 4：96 – 127

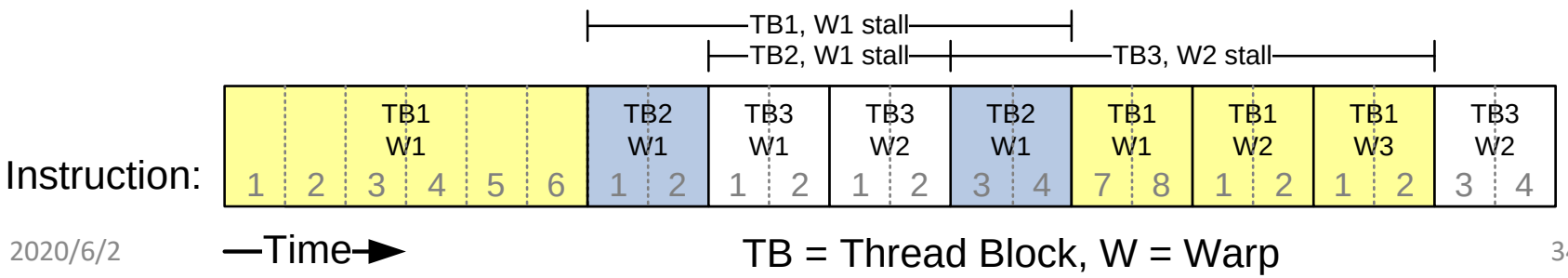
Warp(2/2)

- Thread block裡的threads數量不足32倍數，則將剩下的threads獨立成一個warp
  - 例如：thread block大小(一個block裡thread的數量)為98，就會有四個warps (if 98 threads are assigned to a thread block)：
    - Warp 1 ： 0 – 31
    - Warp 2 ：32 – 63
    - Warp 3 ：64 – 95
    - Warp 4 ： 96 – 97 (only two hreads)
  - Warp 4只有2個thread，所以浪費了30個thread的計算能力，設定thread block大小要注意

Warp Scheduling

- warp排程舉例
  - The SM implements a zero-overhead warp scheduling
  - Warps whose next instruction has its operands ready for consumption are eligible for execution
  - Eligible warps are selected for execution on a prioritized scheduling policy
  - All threads in a warp execute the same instruction
  - 先執行TB1的W1，當執行到第六道指令時需要等待，於是切換到其他thread block去執行其中的warp，等到TB1,W1 stall過後，就可以繼續執行TB1 W1的第七道指令

Coarse-grain multithreading



Reference

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