Thread and Memory Model

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

Thread and Memory Model

Tran Giang Son, tran-giang.son@usth.edu.vn

Thread Model

Thread and Memory Model

Tran Giang Son, tran-giang.son@usth.edu.vn

Thread

- What? a single sequential of execution
- SIMT on GPU
 - Same instruction
 - Same time
 - Different data
 - Natural for graphics and scientific computing
- A way to simplify core

Thread





Thread and Memory Model

 $Tran\ Giang\ Son,\ tran-giang.son@usth.edu.vn$

- Thread: a single flow of kernel execution
- Block: a bunch of thread (1D, 2D, 3D)
 - blockDim.x, blockDim.y, blockDim.z
- Grid: a bunch of block (1D, 2D, 3D)
 - gridDim.x, gridDim.y, gridDim.z

Thread: Restrictions

- Dimensions is fixed after kernel launch
- All blocks in a grid have the same dimension
- Block size and grid size are upper bounded

Thread: Restrictions

- Dimensions is fixed after kernel launch
- All blocks in a grid have the same dimension
- Block size and grid size are upper bounded

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)

Global Thread ID



blockSize = 8

int globalThreadId = threadIdx.x + blockIdx.x * blockDim.x



- Where are we?
 - 1D: x = threadIdx.x + blockIdx.x * blockDim.x
 - 2D: y = threadIdx.y + blockIdx.y * blockDim.y
 - 3D: z = threadIdx.z + blockIdx.z * blockDim.z

- Where are we?
 - 1D: x = threadIdx.x + blockIdx.x * blockDim.x
 - 2D: y = threadIdx.y + blockIdx.y * blockDim.y
 - 3D: z = threadIdx.z + blockIdx.z * blockDim.z
- How about gridDim?
 - Number of blocks in each dimension in the grid
 - Use case: 1D grid for a 2D image
 - Length of a row: w = blockDim.x * gridDim.x
 - Next row: x += w

Thread: Hardware View

- Streaming Processor (CUDA cores)
- Streaming Multiprocessor : A bunch of Streaming Processors plus some extra Special Function Units (sine/cosine/...)
- Graphics Processing Cluster : A bunch of Streaming Multiprocessors
- Many simple cores \Rightarrow better performance

Thread: Hardware View

| | | | | PCI Express 3. | Host Interface | | | | |
|-------------------|-----------------|-------------------------|----------------|----------------|----------------|---------------|--|---|--------------------|
| | | | | GigaThre | id Engine | | | | |
| Memory Controller | | Rester Engine | | | | | Ander Englise | | Merrary Cardraller |
| Municy Consolar | | | | | | | | | Mennery Controller |
| Mercy Correlet | | | | | | | | | Memory Controller |
| Menory Controller | Province Logics | Periodical Ingeneration | N Contractions | | | Andread Legen | Prophysics Taylor Engine Taylor Engine | ere | Memory Controller |

Thread and Memory Model

Tran Giang Son, tran-giang.son@usth.edu.vn

Thread: Hardware View

| SM | | | | | | | | | | | | | | |
|--|--|--|--|--|--|-----------|--|--|---|---|---|---|--|--|
| | | | | | Instruct | tior | n Cache | | | | | | | |
| | l | nstructi | on Buffe | н | | 1 [| Instruction Buffer | | | | | | | |
| | | Warp Se | heduler | | | | Warp Scheduler | | | | | | | |
| • | ispatch Uni | 2 | | Dispatch Unit | | | Dispatch Unit Dispatch Unit | | | | | | | |
| | Regist | er File (' | 16,384 x | 32-bit) | | | Register File (16,384 x 32-bit) | | | | | | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| Core | Core | Core | Core | | SFU | | Core | Core | Core | Core | | SFU | | |
| | Texture / L1 Cache | | | | | | | | | | | | | |
| | | | | | Texture | ן ר יר | 1 Cache | | | | | | | |
| | Tex | | | Tex | Texture | | 1 Cache | Төх | | | Tex | | | |
| | Tex | nstructi | on Buffe | Tex | Texture | | 1 Cache | Tex | nstructi | on Buff | Tex | | | |
| | Tex | nstructi Warp S | on Buffe | Tex | Texture | | 1 Cache | Tex | nstructi Warp St | on Buffi | Tex | | | |
| | Tex I | nstructi Warp S | on Buffe | Tex er Dispatch U | Texture | | 1 Cache | Tex Ispatch Ur | nstructi Warp S4 | on Buffi | Tex M Nspatch U | nit | | |
| | Tex Ispatch Un Regist | nstructi Warp S R R | on Buffe cheduler 16,384 x | Tex er Dispatch U # 1 32-bit) | Texture | | 1 Cache | Tex Ispatch Ur Regist | nstructi Warp Si # # | on Buffi theduler | Tex Hr Nspatch U 32-bit) | nit | | |
| Core | Tex Ispatch Un Regist Core | nstructi Warp S R ter File (Core | on Buffe cheduler 16,384 x Core | Tex er Dispetch U * : 32-bit) LDIST | Texture | | 1 Cache | Tex Ispatch Ur Regist Core | nstructi Warp Sr it it it it it Core | on Buffi theduler 16,384 × Core | Tex M Nspatch U 32-bit) LDIST | nt | | |
| Core | Tex Ispatch Un Regist Core Core | nstructi Warp S R Ker File (Core Core | on Buffe cheduler 16,384 x Core Core | Tex er Dispetch U 2 : 32-bit) LDIST LDIST | Texture Nt SFU SFU | | 1 Cache D Core Core | Tex Ispatch Ur Regist Core Core | nstructi Warp Sr at er File (Core Core | on Buffr theduler 16,384 > Core Core | Tex Pr Sispatch U 32-bit) LDIST | nk SFU SFU | | |
| Core Core | Tex Ispatch Un Regist Core Core Core | nstructi Warp S it cer File (Core Core | on Buffe cheduler 16,384 × Core Core Core | Tex er Dispetch U 32-bit) LDIST LDIST | Texture Nt SFU SFU SFU | | Core Core | Tex Ispatch Ur Regist Core Core Core | nstructi Warp St at Core Core Core | on Buffi heduler 16,384 × Core Core | Tex Hr Dispatch U : 32-bit) LDIST LDIST | ak SFU SFU SFU | | |
| Core Core Core | Tex International Internationa | nstructi Warp S iz cer File (Core Core Core Core | on Buffe cheduler 16,384 × Core Core Core Core | Tex er c 32-bit) LDIST LDIST LDIST | Texture nit SFU SFU SFU SFU | | Core Core Core Core | Tex Regist Core Core Core Core | mstructi Warp St it Core Core Core Core | on Buffi cheduler 16,384 × Core Core Core | Tex Pr Nepatch U 232-bit) LDIST LDIST LDIST | sFU SFU SFU SFU | | |
| Core Core Core Core | Tex I Ispatch Un Regist Core Core Core Core Core | nstructi Warp S ker File (Core Core Core Core Core | on Buffe cheduler 16,384 x Core Core Core Core Core | Tex Dispetch U 2 32-bit) LOIST LOIST LOIST LOIST | Texture Nt SFU SFU SFU SFU SFU | | Core Core Core Core Core | Tex Regist Core Core Core Core Core | nstructi Warp So at Eer File (Core Core Core Core Core | Core Core Core Core Core | Tex H Supatch U 32-bit) LDIST LDIST LDIST LDIST | SFU SFU SFU SFU SFU | | |
| Core Core Core Core Core | Tex ispatch Un Regist Core Core Core Core Core Core | Marp S Warp S Core Core Core Core Core Core | on Buffe cheduler 16,384 x Core Core Core Core Core Core | Tex F Tex Tex Tex Tex Tex Tex Tex Tex | Texture SFU SFU SFU SFU SFU SFU SFU | | Core Core Core Core Core Core | Tex Regist Core Core Core Core Core Core | Natructi Warp Sr it Core Core Core Core Core Core | on Buffit theduler 16,384 > Core Core Core Core Core Core | Tex F Sepatch U LDIST LDIST LDIST LDIST LDIST LDIST | SFU SFU SFU SFU SFU SFU | | |
| Core Core Core Core Core Core | Tex Regist Core Core Core Core Core Core Core | Warp S Warp S Recer File (Core Core Core Core Core Core Core | Core Core Core Core Core Core Core Core | Tex er 2016petch U 2016petch | SFU SFU SFU SFU SFU SFU SFU | | Core Core Core Core Core Core Core | Tex Tex Regist Core Core Core Core Core Core | Warp Sr at Core Core Core Core Core Core Core Core | Core Core Core Core Core Core | Tex F Sheatch U LOST LOST LOST LOST LOST LOST | SFU SFU SFU SFU SFU SFU SFU | | |

Thread and Memory Model

• Each SM has "multiple of 32" cores

- Each SM has "multiple of 32" cores
- Threads in SM execute in group of 32 threads
 - A group of 32 thread inside a SM is called « Warp »
 - Warp is unit of thread scheduling in SMs

- Each SM has "multiple of 32" cores
- Threads in SM execute in group of 32 threads
 - A group of 32 thread inside a SM is called « Warp »
 - Warp is unit of thread scheduling in SMs
- Blocks are assigned to SMs into multiple of warps
 - Number of blocks per SM is constrained

- Each SM has "multiple of 32" cores
- Threads in SM execute in group of 32 threads
 - A group of 32 thread inside a SM is called « Warp »
 - Warp is unit of thread scheduling in SMs
- Blocks are assigned to SMs into multiple of warps
 - Number of blocks per SM is constrained
- No specific mapping between thread and core

- Each warp is executed in SIMD
 - All threads must execute same instruction at any time
- Fact
 - Not all warps are scheduled at anytime
 - Wait for data
 - Branch divergence

- CUDA virtualizes the physical hardware
 - Thread : virtualized scalar processor
 - registers
 - PC
 - state
 - Block is a virtualized multiprocessor
 - threads
 - shared memory

Thread: Branch divergence



Thread and Memory Model

Thread: Branch divergence

- When?
 - Condition
- Divergence

if threadIdx.x > 2:

• No divergence

if threadIdx.x / WARP_SIZE > 2:

Thread: Latency Tolerance

- When a warp does something with high latency
 - Pause it
 - Schedule next warp
- No context switch
 - Large register file
 - No need to "switch" register content to memory
 - Zero overhead

Thread: Latency Tolerance



Thread and Memory Model

Thread: Latency Tolerance

- Latency tolerance relies on many warps
- Branch divergence does not affect GPU high throughput like CPU
- CPU focuses on low latency
 - Branch is important
 - Branch prediction is even more important

Block size in CUDA

• Previously, in launching kernel

kernelName[gridSize, blockSize](args...)

• Example

pixelCount = imageWidth * imageHeight blockSize = 64 gridSize = pixelCount / blockSize grayscale[gridSize, blockSize](devInput, devOutput)

- This is 1D kernel launch
 - numBlock is essentially gridDim.x

Block size in CUDA

- For 2D kernel launches
 - Grid size and block size are 2-dimensional tuples
- Launch a kernel with of 8×8 blocks, each block has 32×32 threads

```
gridSize = (8, 8)
blockSize = (32, 32)
grayscale[gridSize, blockSize](devInput, devOutput)
```

Labwork & Exercises 4: Threads

- Copy labwork 3 code to labwork 4
- Improve labwork 4 code to use 2D blocks
- Use time.time() to measure speedup
- Write a report (in LAT_EX)
 - Name it « Report.4.threads.tex »
 - Explain how you improve the labwork
 - Try experimenting with different 2D block size values
 - Plot a graph of block size vs speedup
 - Compare speedup with previous 1D grid
 - Answer the questions in the upcoming slides, explain why
- Push the report and your code to your forked repository

Thread: Exercises 1

Consider a GPU having the following specs (maximum numbers):

- 512 threads/block
- 1024 threads/SM
- 8 blocks/SM
- 32 threads/warp

What is the best configuration for thread blocks to implement grayscaling?

- 8 × 8
- 16 × 16
- 32 × 32

Thread: Exercises 2

Consider a device SM that can take max

- 1,536 threads
- 4 blocks

Which of the following block configs would result in the most number of threads in the SM?

- 128 threads/blk
- 256 threads/blk
- 512 threads/blk
- 1,024 threads/blk

Thread: Exercises 3

Consider a vector addition problem

- Vector length is 2,000
- Each thread produces one output
- Block size 512 threads.

How many threads will be in the grid?