CUDA lab

1. เปรียบเทียบระหว่าง GPU ทั่วไป (Geforce 940MX) และ GPU Tesla T4

ทดสอบ specifications ของ GPU ทั้งสองตัวได้ดังนี้

- NVIDIA Geforce 940MX (by NVIDIA DeviceQuery, BandwidthTest)

```
Device 0: "GeForce 940MX"
  CUDA Driver Version / Runtime Version          11.0 / 10.2
  CUDA Capability Major/Minor version number:    5.0
  Total amount of global memory:                 2048 MBytes (2147483648 bytes)
  ( 3) Multiprocessors, (128) CUDA Cores/MP:     384 CUDA Cores
  GPU Max Clock rate:                            1189 MHz (1.19 GHz)
  Memory Clock rate:                             2000 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 1048576 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:               zu bytes
  Total amount of shared memory per block:       zu 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:                          zu bytes
  Texture alignment:                             zu bytes
  Concurrent copy and kernel execution:          Yes with 4 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 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 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simulta
neously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.0, CUDA Runtime Version
 = 10.2, NumDevs = 1, Device0 = GeForce 940MX
Result = PASS
```

```
Device 0: GeForce 940MX
Quick Mode

Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)        Bandwidth(MB/s)
  33554432                     1671.5

Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)        Bandwidth(MB/s)
  33554432                     1692.8

Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)        Bandwidth(MB/s)
  33554432                     26098.4

Result = PASS
```

- NVIDIA Tesla T4 (Custom script on Google Cloud Platform)

```
CUDA Device #0
Major revision number:         7
Minor revision number:         5
Name:                          Tesla T4
Total global memory:           2927362048
Total shared memory per block: 49152
Total registers per block:     65536
Warp size:                     32
Maximum memory pitch:          2147483647
Maximum threads per block:     1024
Maximum dimension 0 of block:  1024
Maximum dimension 1 of block:  1024
Maximum dimension 2 of block:  64
Maximum dimension 0 of grid:   2147483647
Maximum dimension 1 of grid:   65535
Maximum dimension 2 of grid:   65535
Clock rate:                    1590000
Total constant memory:         65536
Texture alignment:             512
Concurrent copy and execution: Yes
Number of multiprocessors:     40
Kernel execution timeout:      No
```

```
Device Number: 0
  Device name: Tesla T4
  Memory Clock Rate (KHz): 5001000
  Memory Bus Width (bits): 256
  Peak Memory Bandwidth (GB/s): 320.064000

  CUDA core:  2560
  Maximum threads per block:     1024
  Maximum dimension 0 of block:  1024
  Maximum dimension 1 of block:  1024
  Maximum dimension 2 of block:  64
  Maximum dimension 0 of grid:   2147483647
  Maximum dimension 1 of grid:   65535
  Maximum dimension 2 of grid:   65535
```

จะพบว่า มีส่วนที่แตกต่างกันที่สำคัญอยู่ 3 จุดคือ

- จำนวน multiprocessor

  พบว่า GPU ธรรมดามี multiprocessor (SMs) เพียง 3 ในขณะที่ Tesla T4 มีถึง 40 จำนวนของ SMs นั่นมีผลโดยตรงกับจำนวน threads ที่สามารถประมวลผลได้พร้อมกัน ซึ่งคำนวณได้จาก multiprocessor count * warp size เนื่องจาก warp size ของ GPU ทั้งสองเท่ากัน Tesla T4 จึงสามารถประมวลผลได้เร็วกว่ามาก เพราะมี multiprocessor count มากกว่า

  https://stackoverflow.com/questions/3606636/cuda-model-what-is-warp-size

- ขนาดของ memory bus

  memory bus ของ GPU ทั่วไปมีขนาด 64 bits ในขณะที่ Tesla T4 มีขนาด 256 bits ซึ่งจะช่วยให้การส่งข้อมูลไปยัง Warp มีประสิทธิภาพมากขึ้น ดังนั้นการที่มี memory bus ขนาดใหญ่ขึ้นจึงทำให้มีประสิทธิภาพมากขึ้น

  https://stackoverflow.com/questions/12589416/cuda-coalescing-memory-accesses-and-bus-width
- memory bandwidth

  GPU ทั่วไปมี memory bandwidth (device to device) ที่ 26 GBPS ในขณะที่ Tesla T4 มี memory bandwidth 320 GBPS ซึ่ง memory bandwidth มีมากกว่าจะทำให้การส่งถ่ายข้อมูลมีความรวดเร็วกว่า

2. Monte Carlo Pi Estimation on GPU

จากโปรแกรม ได้ทำการประมาณค่า Pi ผ่านการสุ่มจุดลงบนวงกลมหนึ่งหน่วย โดยให้แต่ละ thread ทำการจุดจำนวน 1 ล้านครั้ง โดยทำงานบน 1 grid 1024 blocks รวมจุดทั้งหมดประมาณ 1 พันล้านจุด พบว่ามีความแม่นยำจนถึงทศนิยมตำแหน่งที่ 4 และใช้เวลาทำงานบน CUDA (ไม่รวมเวลาการทำ memory copy) ประมาณ 1.5 วินาที

```
E:\KMUTT\Y3\CPE351\CPE351-code\lab7>lab7pi
Enter iterations: 1000000
CUDA done, took 1580.370850 ms
Pi estimate for 1000000 iterations = 3.1415160625
```

โปรแกรมทำงานโดยการกำหนดรอบการสุ่มตัวเลขเป็น 1 ล้านรอบ และเลือกให้ทำงานบน 1 grid 1024 blocks จากนั้นทำการจองพื้นที่เก็บของแต่ละ thread เป็น array แบบ long long int แล้วทำการ memcpy ไปยัง device แล้วเรียกใช้งานส่วนของโปรแกรมบน GPU โดยโปรแกรมส่วนนี้จะทำการวนลูปสุ่มตัวเลขตามจำนวนครั้งที่กำหนด โดยสุ่มสองค่า x และ y แทนพิกัดแกน x และ y โดยใช้ฟังก์ชั่น curand_normal_double ของ CUDA แต่ละ thread จะมี seed ต่างกันเพื่อให้ค่าออกมาต่างกันมากที่สุด จากนั้นจะตรวจสอบว่าค่าพิกัดที่สุ่มมานั้นอยู่ในวงกลมหนึ่งหน่วยหรือไม่ด้วยสูตร $x^2 + y^2 \leq 1$ หากตรงตามเงื่อนไขจะนับค่าเพิ่มไป 1

เมื่อวนจนครบรอบแล้ว จะบันทึกค่าไปยัง array ที่ทำการ memcpy มาครั้งแรก เมื่อทำงานทั้งหมดเสร็จแล้วจะทำการ memcpy กลับไปยัง host เพื่อทำการรวมค่าจากทุก thread เข้าด้วยกันแล้วหารด้วยจำนวนจุดทั้งหมด แล้วนำไปคูณ 4 ก็จะได้ค่าประมาณของ Pi (เนื่องจากหากวงกลมซ้อนอยู่ภายในสี่เหลี่ยม อัตราส่วนพื้นที่วงกลมต่อสี่เหลี่ยมจะเป็น Pi/4 จึงทำให้ Pi = 4 * วงกลม / สี่เหลี่ยม โดยโปรแกรมนี้จะประมาณให้จุดทั้งหมดแทนพื้นที่สี่เหลี่ยม และจุดในวงกลมแทนพื้นที่วงกลม) ส่วนการจับเวลานั้นใช้การบันทึก CUDA Events ไว้ แล้วคำนวณเวลาด้วยฟังก์ชั่น cudaEventElapsedTime เมื่อทำงานทุกอย่างเสร็จสิ้นแล้วจึง free memory แล้วออกจากโปรแกรม