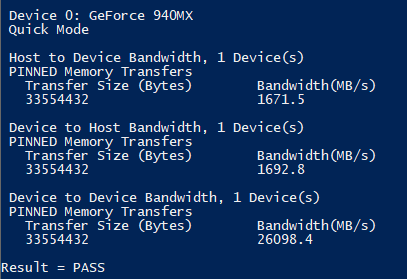
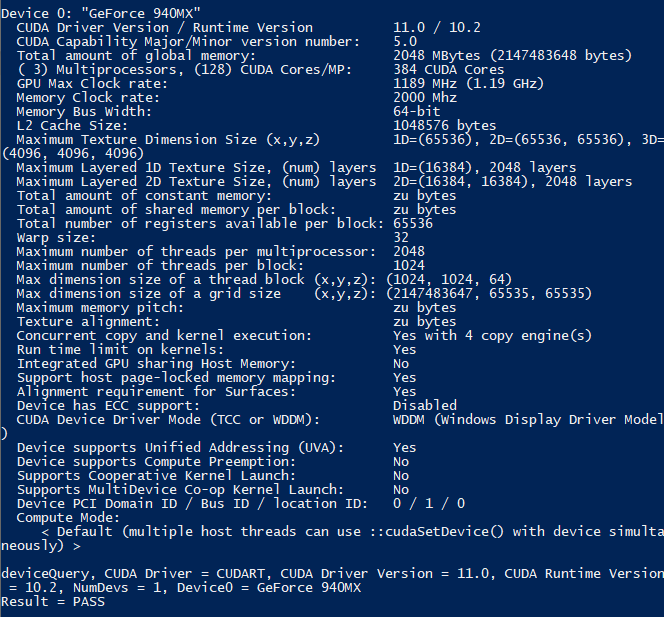
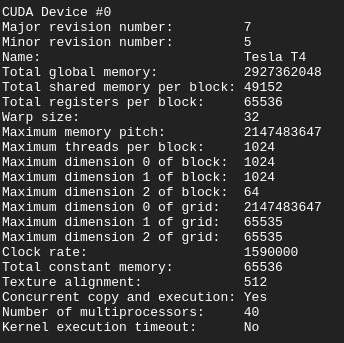
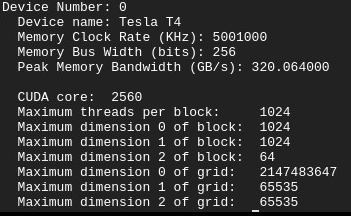
CUDA lab

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

ทดสอบ specifications ของ GPU ทั้งสองตัวได้ดังนี้

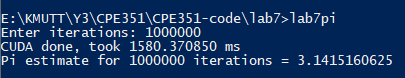
* NVIDIA Geforce 940MX (by NVIDIA DeviceQuery, BandwidthTest)

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

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

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 หากตรงตามเงื่อนไขจะนับค่าเพิ่มไป 1 เมื่อวนจนครบรอบแล้ว จะบันทึกค่าไปยัง array ที่ทำการ memcpy มาครั้งแรก เมื่อทำงานทั้งหมดเสร็จแล้วจะทำการ memcpy กลับไปยัง host เพื่อทำการรวมค่าจากทุก thread เข้าด้วยกันแล้วหารด้วยจำนวนจุดทั้งหมด แล้วนำไปคูณ 4 ก็จะได้ค่าประมาณของ Pi (เนื่องจาก หากวงกลมซ้อนอยู่ภายในสี่เหลี่ยม อัตราส่วนพื้นที่วงกลมต่อสี่เหลี่ยมจะเป็น Pi/4 จึงทำให้ Pi = 4 \* วงกลม / สี่เหลี่ยม โดยโปรแกรมนี้จะประมาณให้จุดทั้งหมดแทนพื้นที่สี่เหลี่ยม และจุดในวงกลมแทนพื้นที่วงกลม) ส่วนการจับเวลานั้นใช้การบันทึก CUDA Events ไว้ แล้วคำนวณเวลาด้วยฟังก์ชั่น cudaEventElapsedTime เมื่อทำงานทุกอย่างเสร็จสิ้นแล้วจึง free memory แล้วออกจากโปรแกรม