บทที่ 4

ผลการทดลอง

ผลการทดลองของงานวิจัยนี้ ได้ทำทดลองกับการประมวลผลการคูณเมตริกซ์ แบบ Single Precision Floating Point ขนาด $M * N$ โปรแกรมการคูณเมตริกซ์นี้ได้นำมาจาก CUDA SDK 3.2 โดยกำหนดขนาดของเมตริกซ์ และจำนวนบล็อกที่นำมาทดลองดังนี้

กำหนดขนาดของเมตริกซ์ $M * N$

เมื่อ $M = 5*(5*n)*16$ โดยที่ n มีค่าตั้งแต่ 2 ถึง 11

$N = 10*(5*n)*16$ โดยที่ n มีค่าตั้งแต่ 2 ถึง 11

กำหนดขนาดของจำนวนบล็อก $X * Y$

เมื่อ $X = 5*(5*k)$ โดยที่ k มีค่าตั้งแต่ 2 ถึง 11

$Y = 10*(5*k)$ โดยที่ k มีค่าตั้งแต่ 2 ถึง 11

ผลการคูณเมตริกซ์ดังกล่าวได้จากสมการ $C = A*B$ ซึ่งขนาดตัวแปร A B และ C นั้น ได้อธิบายขนาดไว้ตามตารางที่ 4.1 รวมไปถึงจำนวนบล็อกและเทรด / บล็อก สำหรับใช้ในการคูณเมตริกซ์แต่ละขนาด ดังตารางที่ 4.2 (ตัวอย่างบางส่วนของโปรแกรม sequential และ parallel อยู่ในภาคผนวก ก) และผลการทดลองที่ได้ในแต่ละการทดลองจะเป็นค่าเฉลี่ยจากการรัน 10 ครั้ง โดยแบ่งผลการทดลองออกเป็น 3 ลักษณะ ได้แก่ 1) ผลการทดลองประมวลผลผ่านซีพียูของเวอร์ชวลแมชชีนเปรียบเทียบกับการประมวลผลด้วยจีพียูผ่านเวอร์ชวลคูด้าของเวอร์ชวลแมชชีน 2) ผลการทดลองแสดงรายละเอียดระยะเวลาที่เรียกใช้งานจีพียูผ่านเวอร์ชวลคูด้า 3) ผลการทดลองการใช้งานจีพียูร่วมกันของเวอร์ชวลแมชชีนผ่านเวอร์ชวลคูด้า โดยแต่ละหัวข้อมีรายละเอียดดังต่อไปนี้

ตารางที่ 4.1

ขนาดของตัวแปรสำหรับใช้ในการประมวลผลการคูณเมตริกซ์

การคูณเมตริกซ์	ตัวแปร A		ตัวแปร B		ตัวแปร C	
	ขนาดเมตริกซ์	ขนาดข้อมูล (Byte)	ขนาดเมตริกซ์	ขนาดข้อมูล (Byte)	ขนาดเมตริกซ์	ขนาดข้อมูล (Byte)
800 x 1600	800 x 1600	5,120,000	800 x 800	2,560,000	800 x 1600	5,120,000
1200 x 2400	1200 x 2400	11,520,000	1200 x 1200	5,760,000	1200 x 2400	11,520,000
1600 x 3200	1600 x 3200	20,480,000	1600 x 1600	10,240,000	1600 x 3200	20,480,000
2000 x 4000	2000 x 4000	32,000,000	2000 x 2000	16,000,000	2000 x 4000	32,000,000
2400 x 4800	2400 x 4800	46,080,000	2400 x 2400	23,040,000	2400 x 4800	46,080,000
2800 x 5600	2800 x 5600	62,720,000	2800 x 2800	31,360,000	2800 x 5600	62,720,000
3200 x 6400	3200 x 6400	81,920,000	3200 x 3200	40,960,000	3200 x 6400	81,920,000
3600 x 7200	3600 x 7200	103,680,000	3600 x 3600	51,840,000	3600 x 7200	103,680,000
4000 x 8000	4000 x 8000	128,000,000	4000 x 4000	64,000,000	4000 x 8000	128,000,000
4400 x 8800	4400 x 8800	154,880,000	4400 x 4400	77,440,000	4400 x 8800	154,880,000

ตารางที่ 4.2

จำนวนบล็อกและเทรด / บล็อกสำหรับใช้ในการประมวลผลการคูณเมตริกซ์

การคูณเมตริกซ์	จำนวน เทรด / บล็อก	จำนวน บล็อก
800 x 1600	16 x 16	50 x 100
1200 x 2400	16 x 16	75 x 150
1600 x 3200	16 x 16	100 x 200
2000 x 4000	16 x 16	125 x 250
2400 x 4800	16 x 16	150 x 300
2800 x 5600	16 x 16	175 x 350
3200 x 6400	16 x 16	200 x 400
3600 x 7200	16 x 16	225 x 450
4000 x 8000	16 x 16	250 x 500
4400 x 8800	16 x 16	275 x 550

4.1 ผลการทดลอง

4.1.1 ผลการทดลองประมวลผลผ่านซีพียูของเวอร์ชวลแมชชีนเปรียบเทียบกับการประมวลผล ด้วยจีพียูผ่านเวอร์ชวลคูด้าของเวอร์ชวลแมชชีน

การทดลองในส่วนนี้จะวัดผลเพื่อเปรียบเทียบการทำงานของเวอร์ชวลแมชชีน กรณีใช้งานซีพียูประมวลผลกับสามารถใช้จีพียูประมวลผลได้โดยผ่านเวอร์ชวลคูด้า โดยวัดผลจากการจับเวลาการประมวลผลการคูณเมตริกซ์ขนาดต่าง ๆ ซึ่งการทดลองนี้จะแสดงผลการทดลองเป็นค่า Speedup เพื่อแสดงให้เห็นถึงประสิทธิภาพในการประมวลผลที่เพิ่มมากขึ้นเมื่อใช้จีพียูเข้ามาช่วยประมวลผลในการทำงาน โดยแสดงผลการทดลองดังตารางที่ 4.1

ผลการทดลองได้จาก $\text{Speedup} = \dfrac{\text{Execution Time}_{\text{VirtualMachine}}}{\text{Execution Time}_{\text{VirtualCUDA}}}$

เมื่อ $\text{Execution Time}_{\text{VirtualMachine}}$ คือ เวลาที่ใช้ในการประมวลผลการคูณเมตริกซ์ผ่านซีพียู ของเวอร์ชวลแมชชีน

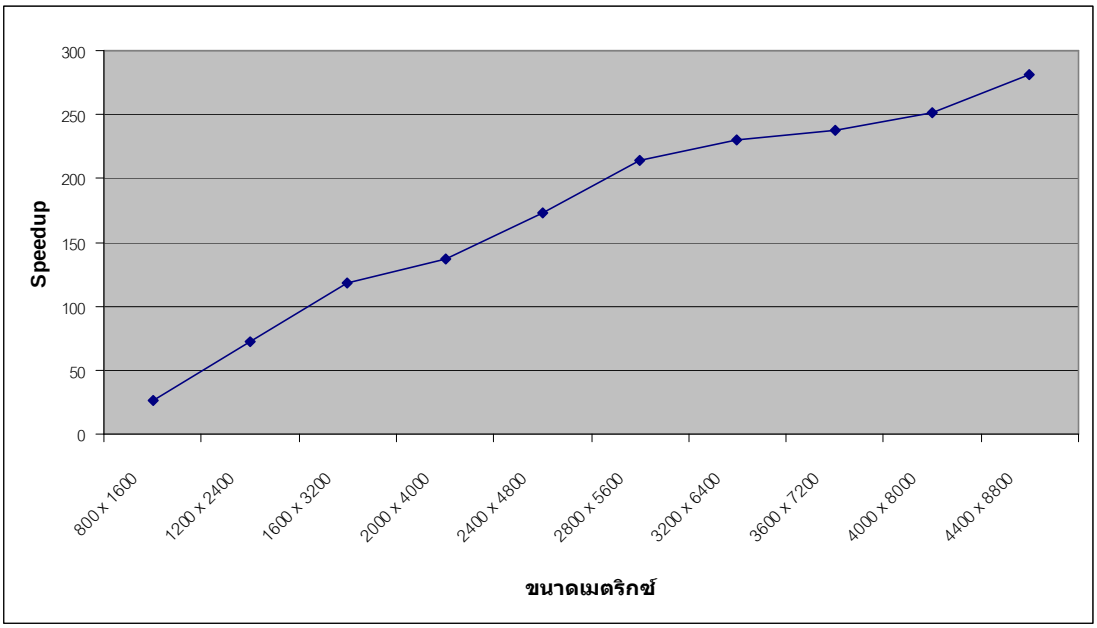
$\text{Execution Time}_{\text{VirtualCUDA}}$ คือ เวลาที่ใช้ในการประมวลผลการคูณเมตริกซ์ของเวอร์ชวลแมชชีนด้วยจีพียูโดยผ่านเวอร์ชวลคูด้า

ตารางที่ 4.3
ผลการทดลองเปรียบเทียบประสิทธิภาพการทำงานระหว่าง
การประมวลผลผ่านซีพียูของเวอร์ชวลแมชชีนกับการประมวลผลผ่านเวอร์ชวลคูด้า

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B+C (Byte)	VirtualMachine (วินาที)	VirtualCUDA (วินาที)	Speedup
800 x 1600	12,800,000	12.10	0.46	26.30
1200 x 2400	28,800,000	50.26	0.69	72.84
1600 x 3200	51,200,000	131.04	1.12	117.00

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B+C (Byte)	VirtualMachine (วินาที)	VirtualCUDA (วินาที)	Speedup
2000 x 4000	80,000,000	256.08	1.86	137.68
2400 x 4800	115,200,000	481.21	2.77	173.72
2800 x 5600	156,800,000	818.67	3.84	213.20
3200 x 6400	204,800,000	1185.20	5.15	230.14
3600 x 7200	259,200,000	1583.93	6.66	237.83
4000 x 8000	320,000,000	2166.46	8.62	251.33
4400 x 8800	387,200,000	3040.83	10.81	281.30

ภาพที่ 4.1
ผลการทดลองเปรียบเทียบประสิทธิภาพการทำงานระหว่าง
การประมวลผลผ่านซีพียูของเวอร์ชวลแมชชีนกับการประมวลผลผ่านเวอร์ชวลคูด้า



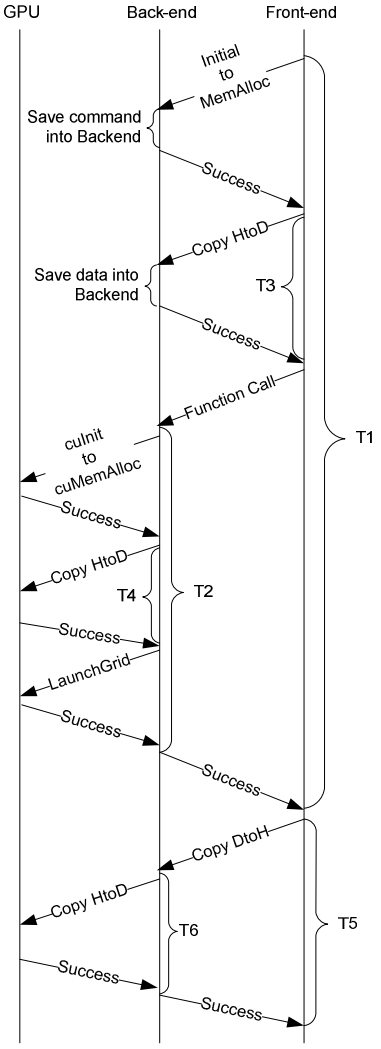
จากการทดลองพบว่าเวลาที่ใช้ในการประมวลผลการคูณเมตริกซ์ในแต่ละขนาดเมตริกซ์ของทั้งซีพียูบนเวอร์ชวลแมชชีน และบนจีพียูที่เรียกใช้งานผ่านเวอร์ชวลคูด้า เวลาที่ในการ

ทำงานของทั้งสองแบบจะใช้เวลาเพิ่มมากขึ้นตามลำดับของขนาดข้อมูลที่ใช้ในการประมวลผล และพบว่าเวลาที่ใช้ในการประมวลผลด้วยซีพียูบนเวอร์ชวลแมชชีนจะใช้เวลามากกว่าเวลาที่ใช้ประมวลผลบนจีพียูโดยผ่านเวอร์ชวลคูด้า ซึ่งแสดงให้เห็นค่า Speedup การทำงานของจีพียูที่เพิ่มมากขึ้นเมื่อเปรียบเทียบกับซีพียู กรณีที่ขนาดของเมตริกซ์มีขนาดเพิ่มมากขึ้น ตามตารางที่ 4.3

4.1.2 ผลการทดลองแสดงรายละเอียดระยะเวลาที่เรียกใช้งานจีพียูผ่านเวอร์ชวลคูด้า

ภาพที่ 4.2

การวัดผลเรียกใช้งานจีพียูของแบ็คเอ็นด์ และฟรอนท์เอนด์ผ่านเวอร์ชวลคูด้า



ผลการทดลองทดลองในส่วนนี้จะแสดงออกเป็น 3 ผลการทดลองคือ 1) ผลการเปรียบเทียบการทำงานระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ ตั้งแต่ Initialization (คำสั่ง cuInit) ถึง Function Call (คำสั่ง cuLaunchGrid) ด้วยการคูณเมตริกซ์ 2) ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Host to Device (จากซีพียูไปจีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ 3) ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Device to Host (จากจีพียูไปซีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ โดยมีรายละเอียดผลการทดลองดังนี้

1. ผลการเปรียบเทียบการทำงานระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ ตั้งแต่ Initialization (คำสั่ง cuInit) ถึง Function Call (คำสั่ง cuLaunchGrid) ด้วยการคูณเมตริกซ์ ซึ่งผลการทดลองที่ได้เกิดจากการจับเวลาการประมวลผลตั้งแต่เริ่ม Initialization การจองพื้นที่สำหรับค่าตัวแปรที่ใช้ในการประมวลผล การคัดลอกข้อมูลตัวแปร A และตัวแปร B ไปวางไว้ยังแบ็คเอ็นด์เพื่อรอทำการประมวลผล และการทำ Function Call ซึ่งผลที่ได้นี้จะไม่รวมการคัดลอกข้อมูลกลับมายัง ฟรอนท์เอนด์ และเวลาที่ใช้ในการยกเลิกการจองพื้นที่ในจีพียู ดังภาพที่ 4.2 โดยการทดลองนี้จะเปรียบเทียบระยะเวลาที่ $T1$ กับ $T2$ ผลการทดลองนี้ได้แสดงไว้ในตารางที่ 4.4

เมื่อ $T1$ คือ ระยะเวลาการทำงานของฟรอนท์เอนด์ ตั้งแต่ Initialization (คำสั่ง cuInit) ถึง Function Call (คำสั่ง cuLaunchGrid)

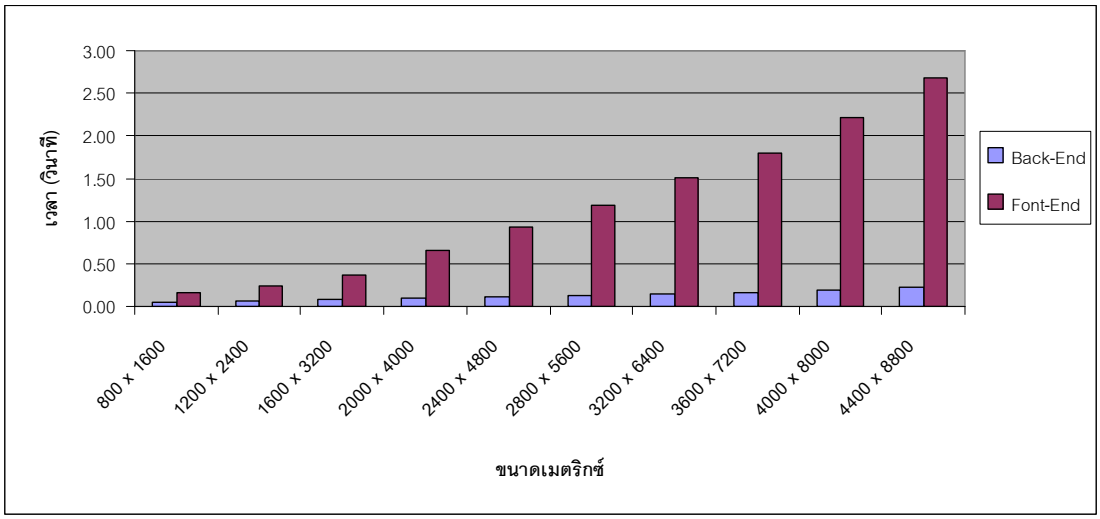
$T2$ คือ ระยะเวลาการทำงานของแบ็คเอ็นด์ ตั้งแต่ Initialization (คำสั่ง cuInit) ถึง Function Call (คำสั่ง cuLaunchGrid)

ตารางที่ 4.4
ผลการทดลองเปรียบเทียบการทำงานระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์
ตั้งแต่ Initialization ถึง Function Call ด้วยการคูณเมตริกซ์

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B (Byte)	เวลา (วินาที)	
		แบ็คเอ็นด์ ($T2$)	ฟรอนท์เอนด์ ($T1$)
800 x 1600	7,680,000	0.055	0.159
1200 x 2400	17,280,000	0.061	0.233
1600 x 3200	30,720,000	0.077	0.370

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B (Byte)	เวลา (วินาที)	
		แบ็คเอ็นด์ ($T2$)	ฟรอนท์เอนด์ ($T1$)
2000 x 4000	48,000,000	0.094	0.657
2400 x 4800	69,120,000	0.110	0.929
2800 x 5600	94,080,000	0.124	1.207
3200 x 6400	122,880,000	0.148	1.522
3600 x 7200	155,520,000	0.165	1.799
4000 x 8000	192,000,000	0.191	2.213
4400 x 8800	232,320,000	0.218	2.699

ภาพที่ 4.3
ผลการทดลองเปรียบเทียบการทำงานระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์
ตั้งแต่ Initialization ถึง Function Call ด้วยการคูณเมตริกซ์



ผลการทดลองจากตารางที่ 4.4 พบว่าเวลาที่ใช้ในการประมวลผลของฟรอนท์เอนด์จะใช้เวลามากกว่าการประมวลผลของแบ็คเอ็นด์ คือ 0.104, 0.172, 0.293, 0.563, 0.819, 1.083, 1.374, 1.634, 2.022 และ 2.481 วินาที ตามลำดับของขนาดข้อมูลเมตริกซ์ที่เพิ่มมากขึ้น ซึ่งโอเวอร์เฮดดังกล่าวเกิดจากเวลาในการคัดลอกข้อมูลของ ฟรอนท์เอนด์ผ่านเน็ตเวิร์คไปยัง

แบ็คเอ็นด์บวกกับเวลาที่เสียไปสำหรับการเข้าใช้งานจีพียูของแบ็คเอ็นด์ เพราะการทำงานของฟรอนท์เอนด์นั้นจำเป็นต้องส่งข้อมูลที่ต้องใช้ในการประมวลผลไปยังแบ็คเอ็นด์เพื่อใช้ในการประมวลผลและรอให้แบ็คเอ็นด์ประมวลผลการทำงานและเข้าถึงจีพียูทำให้เวลาในการประมวลผลของฟรอนท์เอนด์มากขึ้นตามลำดับ

2. ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Host to Device (จากซีพียูไปจีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ ซึ่งผลการทดลองที่ได้เกิดจากการวัดเวลาที่ใช้ไปสำหรับฟังก์ชั่นคัดลอกข้อมูลตัวแปร A และตัวแปร B ของทั้งแบ็คเอ็นด์ และฟรอนท์เอนด์ ดังภาพที่ 4.2 โดยการทดลองนี้จะเปรียบเทียบระยะเวลาที่ $T3$ กับ $T4$ โดยผลการทดลองนี้ได้แสดงไว้ในตารางที่ 4.5

เมื่อ $T3$ คือ ระยะเวลาที่ใช้ในการคัดลอกข้อมูลแบบ Host to Device ของฟรอนท์เอนด์

$T4$ คือ ระยะเวลาที่ใช้ในการคัดลอกข้อมูลแบบ Host to Device ของแบ็คเอ็นด์

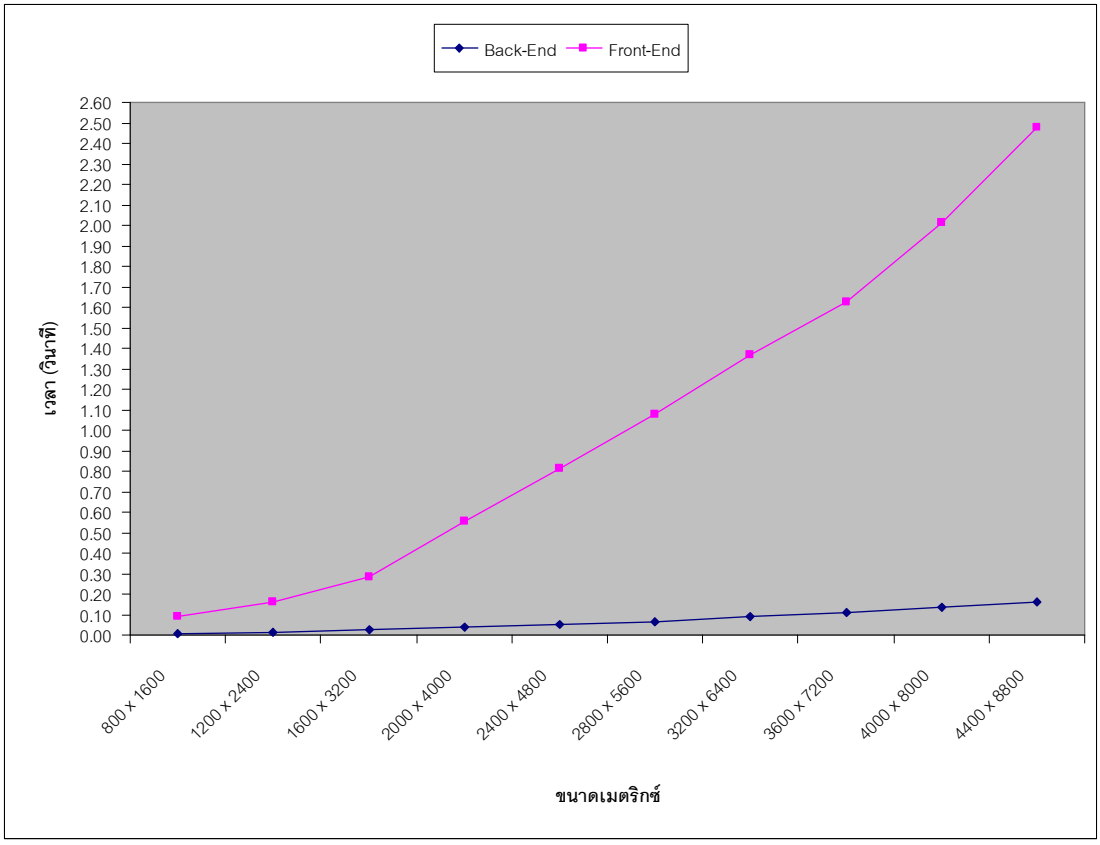
ตารางที่ 4.5

ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Host to Device (จากซีพียูไปจีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B (Byte)	เวลา (วินาที)	
		แบ็คเอ็นด์ ($T4$)	ฟรอนท์เอนด์ ($T3$)
800 x 1600	7,680,000	0.006	0.092
1200 x 2400	17,280,000	0.013	0.163
1600 x 3200	30,720,000	0.023	0.283
2000 x 4000	48,000,000	0.036	0.553
2400 x 4800	69,120,000	0.052	0.811
2800 x 5600	94,080,000	0.067	1.075
3200 x 6400	122,880,000	0.090	1.366
3600 x 7200	155,520,000	0.108	1.626
4000 x 8000	192,000,000	0.134	2.013

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B (Byte)	เวลา (วินาที)	
		แบ็คเอ็นด์ ($T4$)	ฟรอนท์เอนด์ ($T3$)
4400 x 8800	232,320,000	0.160	2.476

ภาพที่ 4.4
ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Host to Device (จากซีพียูไปจีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์



ผลการทดลองจากตารางที่ 4.5 พบว่าเวลาที่ใช้ในคัดลอกข้อมูลแบบ Host to Device ของฟรอนท์เอนด์จะใช้เวลามากกว่าคัดลอกข้อมูลแบบ Host to Device ของแบ็คเอ็นด์ คือ 0.086, 0.150, 0.260, 0.517, 0.759, 1.008, 1.276, 1.518, 1.879 และ 2.316 วินาที ตามลำดับของขนาดข้อมูลเมตริกซ์ที่เพิ่มมากขึ้น ซึ่งโอเวอร์เฮดดังกล่าวเกิดจากเวลาในการคัดลอกข้อมูลของ

ฟรอนท์เอนด์ผ่านเน็ตเวิร์คไปยังแบ็คเอ็นด์ แต่สำหรับการคัดลอกข้อมูลในส่วนของแบ็คเอ็นด์นั้นไม่จำเป็นต้องเสียเวลาในการคัดลอกข้อมูลผ่านเน็ตเวิร์ค เพราะข้อมูลได้ถูกนำมาวางไว้ที่แบ็คเอ็นด์เรียบร้อยแล้ว จึงทำให้เวลาที่เสียไปสำหรับการคัดลอกข้อมูลแบบ Host to Device ของแบ็คเอ็นด์ใช้เวลาน้อยกว่าฟรอนท์เอนด์

3. ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Device to Host (จากจีพียูไปซีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ ซึ่งผลการทดลองที่ได้เกิดจากการวัดเวลาที่ใช้ไปสำหรับฟังก์ชั่นคัดลอกข้อมูลตัวแปร C ของทั้งแบ็คเอ็นด์ และฟรอนท์เอนด์ ดังภาพที่ 4.2 โดยการทดลองนี้จะเปรียบเทียบระยะเวลาที่ $T5$ กับ $T6$ โดยผลการทดลองนี้ได้แสดงไว้ในตารางที่ 4.6

เมื่อ $T5$ คือ ระยะเวลาที่ใช้ในการคัดลอกข้อมูลแบบ Device to Host ของฟรอนท์เอนด์

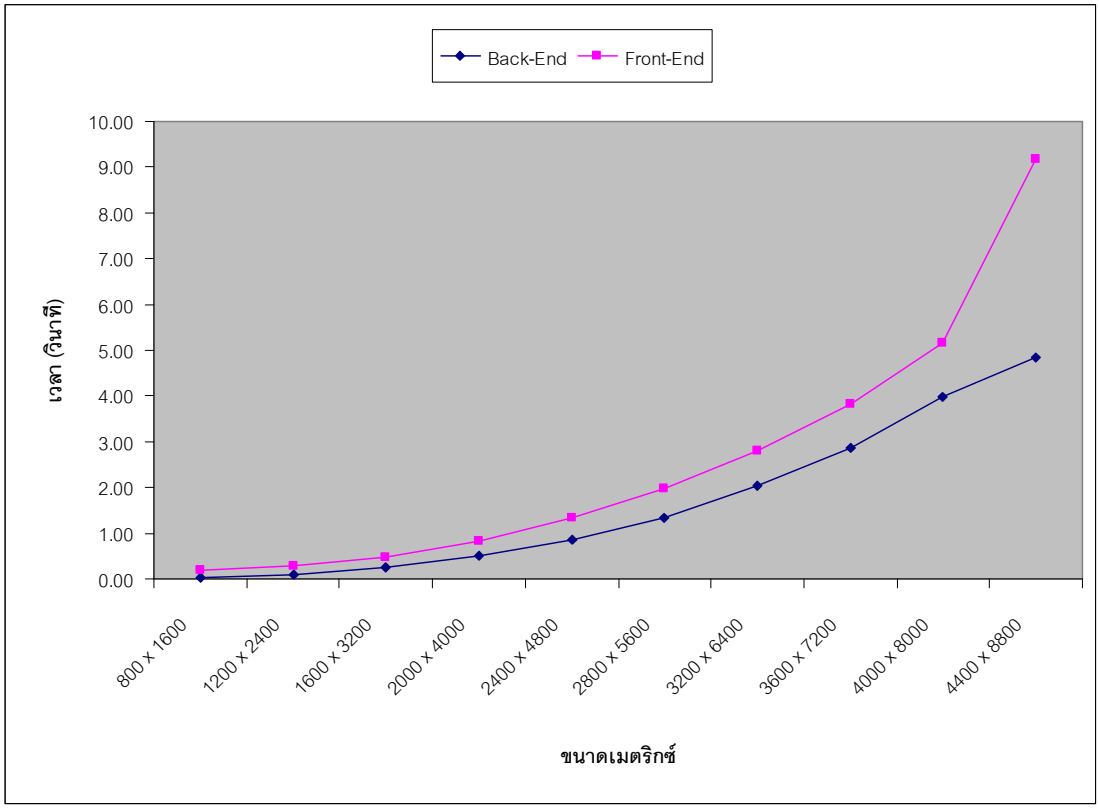
$T6$ คือ ระยะเวลาที่ใช้ในการคัดลอกข้อมูลแบบ Device to Host ของแบ็คเอ็นด์

ตารางที่ 4.6

ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Device to Host (จากจีพียูไปซีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์

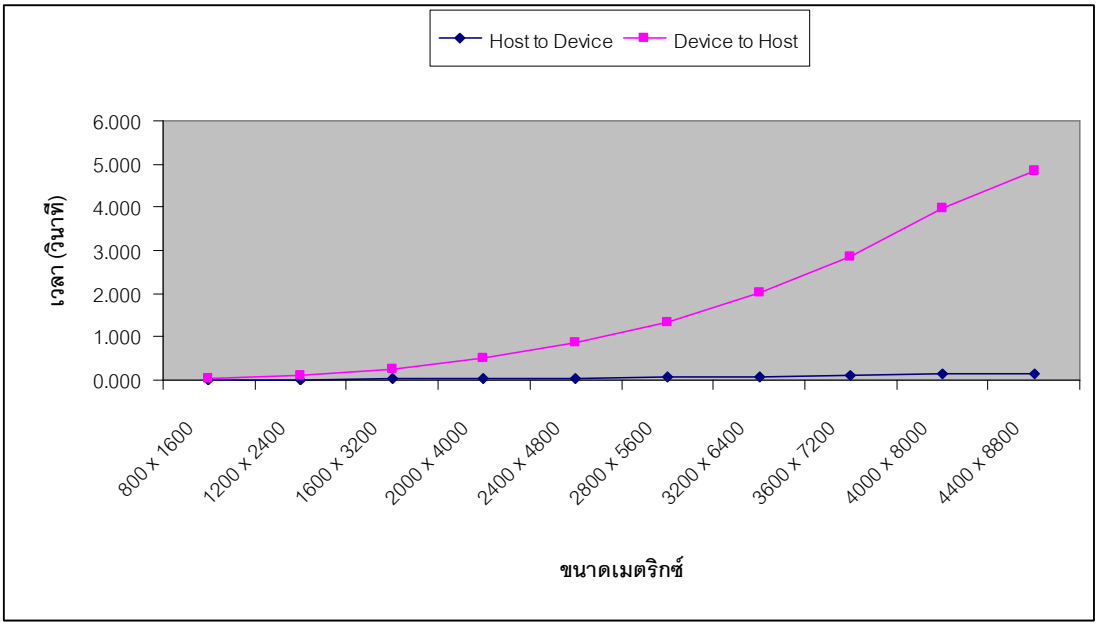
ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า C (Byte)	เวลา (วินาที)	
		แบ็คเอ็นด์ ($T6$)	ฟรอนท์เอนด์ ($T5$)
800 x 1600	5,120,000	0.034	0.184
1200 x 2400	11,520,000	0.111	0.276
1600 x 3200	20,480,000	0.255	0.469
2000 x 4000	32,000,000	0.495	0.822
2400 x 4800	46,080,000	0.863	1.344
2800 x 5600	62,720,000	1.346	1.966
3200 x 6400	81,920,000	2.030	2.813
3600 x 7200	103,680,000	2.850	3.833
4000 x 8000	128,000,000	3.974	5.175
4400 x 8800	154,880,000	4.842	9.180

ภาพที่ 4.5
ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Device to Host (จากจีพียูไปซีพียู)
ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์



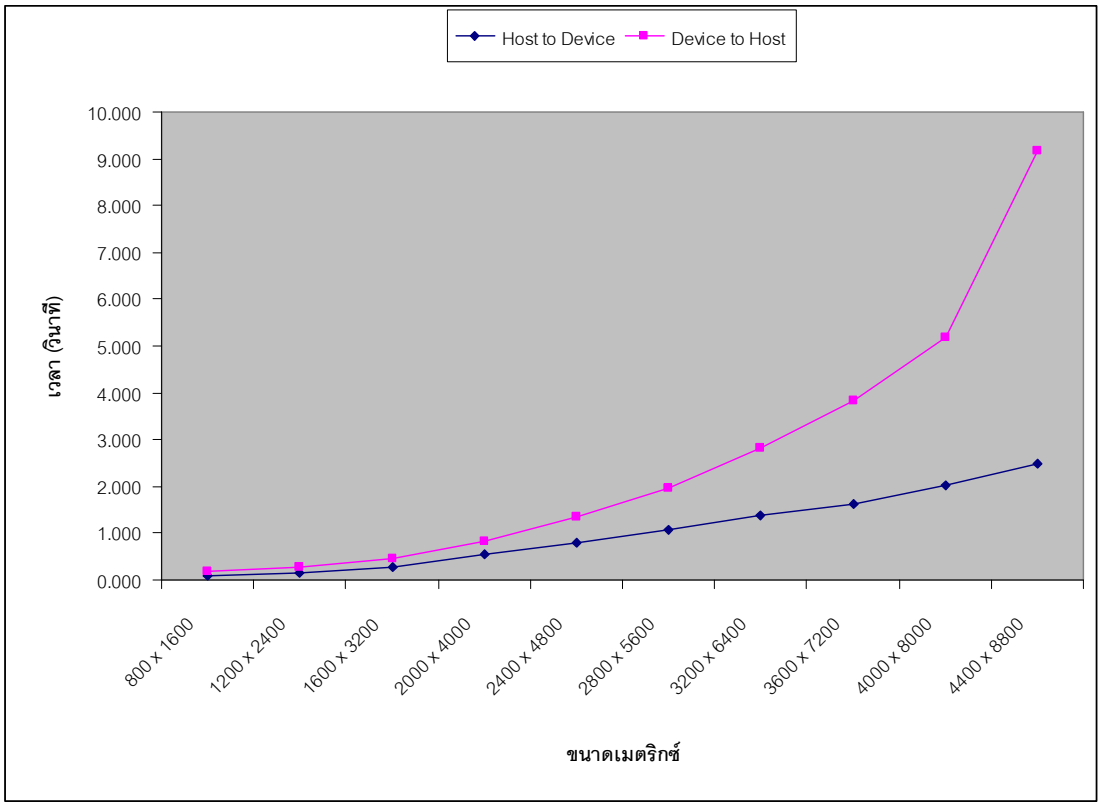
ผลการทดลองจากตารางที่ 4.6 พบว่าเวลาที่ใช้ในคัดลอกข้อมูลแบบ Device to Host ของฟรอนท์เอนด์จะใช้เวลามากกว่าคัดลอกข้อมูลแบบ Device to Host ของแบ็คเอ็นด์ คือ 0.150, 0.165, 0.214, 0.327, 0.481, 0.620, 0.783, 0.983, 1.201 และ 4.338 วินาที ตามลำดับของขนาดข้อมูลเมตริกซ์ที่เพิ่มมากขึ้น ซึ่งโอเวอร์เฮดดังกล่าวเกิดจากเวลาที่ใช้ในการคัดลอกข้อมูลจากยังแบ็คเอ็นด์ผ่านเน็ตเวิร์คกลับมาที่ฟรอนท์เอนด์บวกกับเวลาที่แบ็คเอ็นด์คัดลอกข้อมูลจากจีพียูมาว่างไว้ยังซีพียูเพื่อเตรียมที่จะส่งกลับไปยังฟรอนท์เอนด์ แต่สำหรับการคัดลอกข้อมูลในส่วนของแบ็คเอ็นด์นั้นไม่จำเป็นต้องเสียเวลาในการคัดลอกข้อมูลผ่านเน็ตเวิร์ค แต่ทำแค่เพียงคัดลอกมาวางไว้ยังซีพียูเท่านั้น จึงทำให้เวลาที่เสียไปสำหรับการคัดลอกข้อมูลแบบ Device to Host ของแบ็คเอ็นด์ใช้เวลาน้อยกว่าฟรอนท์เอนด์

ภาพที่ 4.6
ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลของแบ็คเอ็นด์
ระหว่าง Device to Host (จากจีพียูไปซีพียู) กับ Host to Device (จากซีพียูไปจีพียู)



จากผลการทดลองตามตารางที่ 4.5 และ ตารางที่ 4.6 พบว่าเวลาที่ใช้คัดลอกข้อมูลของแบ็คเอ็นด์แบบ Host to Device จะใช้เวลาในการคัดลอกข้อมูลน้อยกว่าเวลาที่ใช้สำหรับคัดลอกข้อมูลแบบ Device to Host เนื่องจากการคัดลอกแบบ Host to Device เป็นการทำ asynchronous function call แต่สำหรับการคัดลอกแบบ Device to Host เป็นการทำแบบ synchronous

ภาพที่ 4.7
ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลของฟรอนท์เอนด์
ระหว่าง Device to Host (จากจีพียูไปซีพียู) กับ Host to Device (จากซีพียูไปจีพียู)



จากผลการทดลองตามตารางที่ 4.5 และ ตารางที่ 4.6 พบว่าเวลาที่ใช้คัดลอกข้อมูลของฟรอนท์เอนด์แบบ Host to Device จะใช้เวลาในการคัดลอกข้อมูลน้อยกว่าเวลาที่ใช้สำหรับคัดลอกข้อมูลแบบ Device to Host เนื่องจากการคัดลอกแบบ Host to Device ของฟรอนท์เอนด์จะต้องเสียโอเวอร์เฮดในส่วนของเวลาที่ใช้ไปสำหรับการคัดลอกข้อมูลผ่านเน็ตเวิร์คจากฟรอนท์เอนด์มาเก็บไว้ยังแบ็คเอ็นด์เท่านั้น แต่สำหรับการคัดลอกข้อมูลแบบ Device to Host จะต้องเสียโอเวอร์เฮดในส่วนของเวลาที่ใช้ไปสำหรับการคัดลอกข้อมูลจากจีพียูมาที่แบ็คเอ็นด์บวกกับเวลาที่ต้องถ่ายโอนข้อมูลผ่านเน็ตเวิร์คจากแบ็คเอ็นด์ไปยังฟรอนท์เอนด์ทำให้เวลาในการคัดลอกข้อมูลแบบ Device to Host ใช้เวลามากกว่าการคัดลอกข้อมูลแบบ Host to Device

4.1.3 ผลการทดลองการใช้งานจีพียูร่วมกันของเวอร์ชวลแมชชีน

การทดลองในส่วนนี้จะวัดผลเพื่อแสดงถึงประโยชน์และความสามารถของเวอร์ชวลคูด้าในการจัดสรรทรัพยากรจีพียูสำหรับการใช้งานทรัพยากรจีพียูร่วมกันของเวอร์ชวลแมชชีน โดยแสดงผลการทดลองออกเป็น 2 แบบ คือ 1) ผลการทดลองเปรียบเทียบการทำงานระหว่างการใช้งานจีพียูที่แบ็คเอ็นด์อนุญาตให้เข้าถึงได้เพียงครั้งละ 1 เวอร์ชวลแมชชีนกับการใช้งานจีพียูพร้อมกันครั้งละมากกว่า 1 เวอร์ชวลแมชชีน 2) ผลการทดลองเปรียบเทียบการทำงานระหว่างการใช้งานจีพียูที่ 1 เวอร์ชวลแมชชีนกับการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีน โดยแสดงรายละเอียดผลการทดลองดังต่อไปนี้

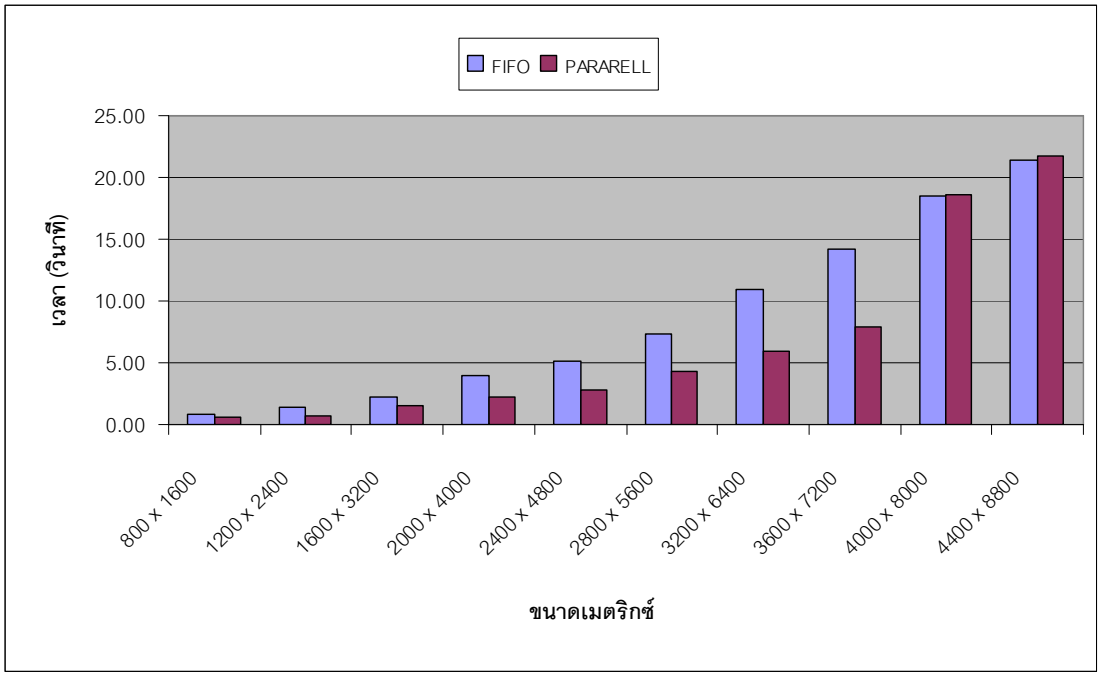
1. ผลการทดลองเปรียบเทียบการทำงานระหว่างการใช้งานจีพียูที่แบ็คเอ็นด์อนุญาตให้เข้าถึงได้เพียงครั้งละ 1 เวอร์ชวลแมชชีนกับการใช้งานจีพียูที่แบ็คเอ็นด์อนุญาตให้เข้าถึงได้พร้อมกันครั้งละมากกว่า 1 เวอร์ชวลแมชชีน ซึ่งวัดผลด้วยการจับเวลาจาก 2 เวอร์ชวลแมชชีนที่ต้องการเรียกใช้งานจีพียูผ่านเวอร์ชวลคูด้าพร้อมกันโดยประมวลผลการคูณเมตริกซ์ขนาดต่าง ๆ ผลที่ได้แสดงดังตารางที่ 4.7

ตารางที่ 4.7
ผลการทดลองเปรียบเทียบการทำงานเข้าถึงจีพียูสำหรับเวอร์ชวลแมชชีน
แบบครั้งละ 1 เวอร์ชวลแมชชีน และพร้อมกัน 2 เวอร์ชวลแมชชีน

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B+C (Byte)	เวลา (วินาที)	
		FIFO	PARARELL
800 x 1600	12,800,000	0.83	0.57
1200 x 2400	28,800,000	1.36	0.71
1600 x 3200	51,200,000	2.26	1.51
2000 x 4000	80,000,000	3.98	2.26
2400 x 4800	115,200,000	5.06	2.85
2800 x 5600	156,800,000	7.31	4.36
3200 x 6400	204,800,000	10.92	5.95

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B+C (Byte)	เวลา (วินาที)	
		FIFO	PARARELL
3600 x 7200	259,200,000	14.14	7.85
4000 x 8000	320,000,000	18.54	18.63
4400 x 8800	387,200,000	21.39	21.74

ภาพที่ 4.8
ผลการทดลองเปรียบเทียบการทำงานเข้าถึงจีพียูสำหรับเวอร์ชวลแมชชีน
แบบครั้งละ 1 เวอร์ชวลแมชชีน และพร้อมกัน 2 เวอร์ชวลแมชชีน



ผลการทดลองจากตารางที่ 4.7 พบว่าเวลาที่ใช้ในการประมวลผลการคูณเมตริกซ์โดยเรียกใช้งานจีพียูผ่านเวอร์ชวลคู่ด้ากรณีแบ็คเอ็นด์อนุญาตให้เข้าถึงจีพียูได้เพียงครั้งละ 1 เวอร์ชวลแมชชีนจะใช้เวลาในการประมวลผลมากกว่ากรณีแบ็คเอ็นด์อนุญาตให้ใช้งานจีพียูพร้อมกันครั้งละ2 เวอร์ชวลแมชชีน ที่เมตริกซ์มีขนาด 800 x 1600, 1200 x 2400, 1600 x 3200, 2000 x 4000, 2400 x 4800, 2800 x 5600, 3200 x 6400 และ 3600 x 7200 ด้วยผลต่างเวลา 0.26, 0.65, 1.72, 2.21, 2.95, 4.97 และ 6.29 วินาที ตามขนาดของเมตริกซ์ที่เพิ่มมากขึ้น แต่ขณะที่

เมตริกซ์มีขนาด 4000 x 8000 และ 4400 x 8800 นั้นการประมวลผลการคูณเมตริกซ์โดยเรียกใช้งานจีพียูผ่านเวอร์ชวลคูด้ากรณีแบ็คเอ็นด์อนุญาตให้เข้าถึงจีพียูได้เพียงครั้งละ 1 เวอร์ชวลแมชชีน จะใช้เวลาในการประมวลผลน้อยกว่ากรณีแบ็คเอ็นด์อนุญาตให้ใช้งานจีพียูพร้อมกันครั้งละ 2 เวอร์ชวลแมชชีน ที่ 0.09 และ 0.35 วินาที เนื่องจากเมตริกซ์ขนาด 4000 x 8000 และ 4400 x 8800 ต้องใช้พื้นที่สำหรับข้อมูลที่ต้องนำไปประมวลผลเกินกว่าพื้นที่ ที่จีพียูมีให้บริการ ดังนั้นแบ็คเอ็นด์ของเวอร์ชวลคูด้าจึงทำการจัดสรรทรัพยากรจีพียูโดยให้เวอร์ชวลแมชชีนที่เข้ามาขอใช้จีพียูรอต่อคิวเพื่อใช้งานจีพียูจนกว่าเวอร์ชวลแมชชีนเครื่องแรกจะประมวลผลเสร็จและมีพื้นที่เหลือเพียงพอที่จะให้เวอร์ชวลแมชชีนเครื่องต่อไปทำงาน จึงเป็นเหตุให้เวลาที่ได้จากการประมวลผลสำหรับ 2 เวอร์ชวลแมชชีนที่สามารถเข้าใช้งานจีพียูได้พร้อมกันมีเวลามากกว่าการประมวลผลสำหรับ 2 เวอร์ชวลแมชชีนที่เข้าใช้งานจีพียูได้เพียงครั้งละ 1 เวอร์ชวลแมชชีน

2. ผลการทดลองเปรียบเทียบการทำงานระหว่างการใช้งานจีพียูที่1เวอร์ชวลแมชชีน กับการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีน โดยทำการจับเวลาการประมวลผลการคูณเมตริกซ์ขนาดต่าง ๆ ซึ่งผลที่ได้แสดงดังตารางที่ 4.8 ดังนี้

ตารางที่ 4.8

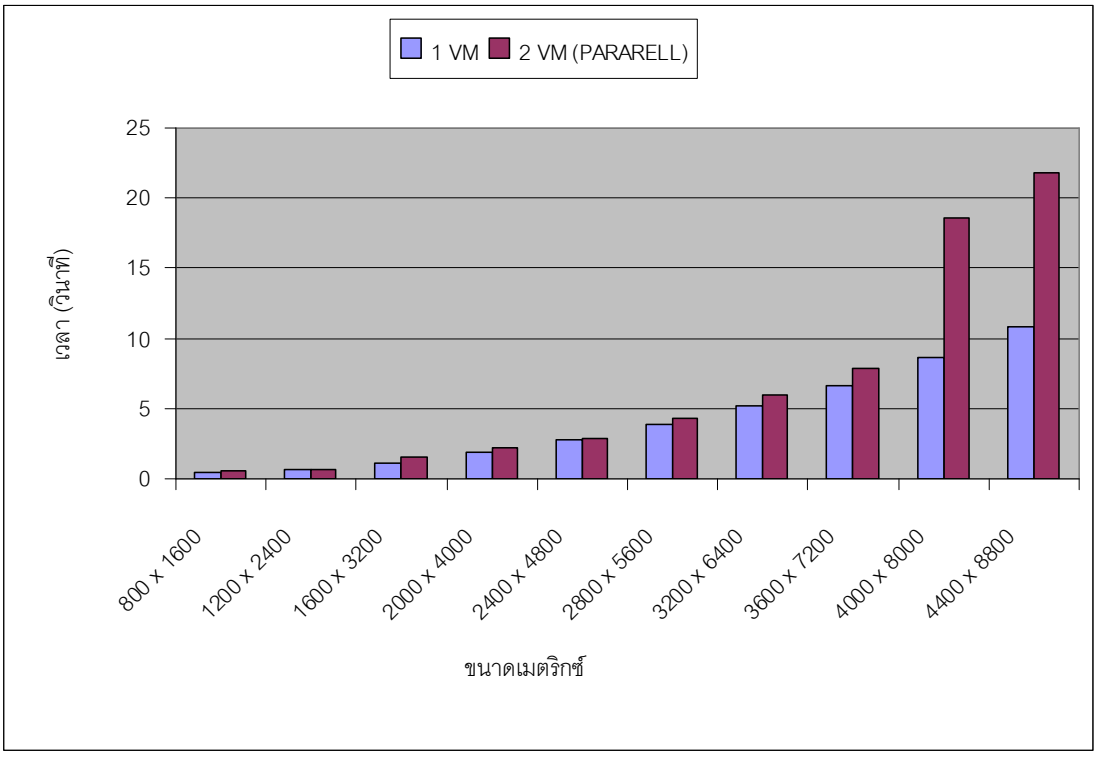
ผลการทดลองเปรียบเทียบการทำงาน

ระหว่างการใช้งานจีพียูที่ 1 เวอร์ชวลแมชชีน กับการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีน

ขนาดเมตริกซ์	ขนาดเมตริกซ์ ตามค่า A+B+C (Byte)	เวลา (วินาที)	
		1 VM	2 VM (PARARELL)
800 x 1600	12,800,000	0.46	0.57
1200 x 2400	28,800,000	0.69	0.71
1600 x 3200	51,200,000	1.12	1.51
2000 x 4000	80,000,000	1.86	2.26
2400 x 4800	115,200,000	2.77	2.85
2800 x 5600	156,800,000	3.84	4.36
3200 x 6400	204,800,000	5.15	5.95
3600 x 7200	259,200,000	6.66	7.85

ขนาดเมตริกซ์	ขนาดเมตริกซ์ตามค่า A+B+C (Byte)	เวลา (วินาที)	
		1 VM	2 VM (PARARELL)
4000 x 8000	320,000,000	8.62	18.63
4400 x 8800	387,200,000	10.81	21.74

ภาพที่ 4.9
ผลการทดลองเปรียบเทียบการทำงานระหว่าง
การใช้งานจีพียูที่ 1 เวอร์ชวลแมชชีน กับการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีน



ผลการทดลองจากตารางที่ 4.8 พบว่า การประมวลผลการคูณเมตริกซ์ สำหรับการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีน ที่เมตริกซ์มีขนาด 800 x 1600, 1200 x 2400, 1600 x 3200, 2000 x 4000, 2400 x 4800, 2800 x 5600, 3200 x 6400 และ 3600 x 7200 เวลาที่ใช้ในการประมวลผลไม่มากเท่ากับ 2 เท่าของการประมวลผลการคูณเมตริกซ์สำหรับการใช้งานจีพียูที่ 1 เวอร์ชวลแมชชีน แต่ใช้เวลามากกว่าเพียง 0.11, 0.02, 0.39, 0.40, 0.08, 0.52, 0.80

และ 1.19 วินาที ตามลำดับ แตกต่างกับที่เมตริกซ์มีขนาด 4000 x 8000 และ 4400 x 8800 คือ การประมวลผลการคูณเมตริกซ์ สำหรับการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีนจะใช้เวลามากกว่า การประมวลผลการคูณเมตริกซ์สำหรับการใช้งานจีพียูที่ 1 เวอร์ชวลแมชชีน 2 เท่า เนื่องจากเมตริกซ์ขนาด 4000 x 8000 และ 4400 x 8800 ต้องใช้พื้นที่สำหรับข้อมูลที่ต้องนำไปประมวลผลเกินกว่าพื้นที่ ที่จีพียูมีให้บริการ ดังนั้นแบ็คเอ็นด์ของเวอร์ชวลคูด้าจึงทำการจัดสรรทรัพยากรจีพียูโดยให้เวอร์ชวลแมชชีนที่เข้ามาขอใช้จีพียูรอต่อคิวเพื่อใช้งานจีพียูจนกว่าเวอร์ชวลแมชชีนเครื่องแรกจะประมวลผลเสร็จและมีพื้นที่เหลือเพียงพอที่จะให้เวอร์ชวลแมชชีนเครื่องต่อไปทำงาน จึงเป็นเหตุให้เวลาที่ได้จากการประมวลผลการใช้งานจีพียูพร้อมกัน 2 เวอร์ชวลแมชชีน มีเวลามากกว่าการใช้งานจีพียูที่ 1 เวอร์ชวลแมชชีนเป็น 2 เท่า

4.2 อภิปรายผลการวิจัย

4.2.1 อภิปรายผลการประมวลผลผ่านซีพียูและจีพียูของเวอร์ชวลแมชชีน

จากการผลทดลองแสดงให้เห็นได้ว่านอกจากเวอร์ชวลคูด้าจะทำให้เวอร์ชวลแมชชีนสามารถเรียกใช้งานจีพียูได้แล้ว เวอร์ชวลคูด้ายังเป็นผู้เพิ่มประสิทธิภาพในการประมวลผลของแอพพลิเคชันบางประเภท เช่น การคูณเมตริกซ์ ซึ่งสังเกตได้จากผลการทดลองในกรณีที่เวอร์ชวลแมชชีนไม่สามารถเรียกใช้งานจีพียูให้ช่วยประมวลผล เวอร์ชวลแมชชีนจำเป็นต้องประมวลผลบนซีพียูเท่านั้น ซึ่งเวลาที่ได้จากการใช้ซีพียูประมวลผลนั้นจะใช้เวลามากเมื่อเปรียบเทียบกับเมื่อใช้จีพียูเข้ามาช่วยประมวลผล

4.2.2 อภิปรายผลของระยะเวลาที่เรียกใช้งานจีพียูผ่านเวอร์ชวลคูด้า

จากการผลทดลองในส่วนนี้ทั้ง 3 ผลการทดลองคือ 1) ผลการเปรียบเทียบการทำงานระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ ตั้งแต่ Initialization ถึง Call Function ด้วยการคูณเมตริกซ์ 2) ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Host to Device (จากซีพียูไปจีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ 3) ผลการเปรียบเทียบฟังก์ชั่นการคัดลอกข้อมูลแบบ Device to Host (จากจีพียูไปซีพียู) ระหว่างแบ็คเอ็นด์ และฟรอนท์เอนด์ เห็นได้ชัดเจนว่าเวลาที่เสียไปในการเรียกใช้งานจีพียูของเวอร์ชวลแมชชีนส่วนมากจะเป็นเวลาที่ใช้ในการถ่ายโอนข้อมูลจากฟรอนท์เอนด์ไปยังแบ็คเอ็นด์ และการรอรับค่าจากแบ็คเอ็นด์มายังจากฟรอนท์เอนด์

4.2.3 อภิปรายผลการใช้งานจีพียูร่วมกันของเวอร์ชวลแมชชีน

จากการผลทดลองในส่วนนี้แสดงให้เห็นถึงประโยชน์และความสามารถของเวอร์ชวลคูด้าในการจัดสรรทรัพยากรจีพียูสำหรับการใช้งานทรัพยากรจีพียูร่วมกันของเวอร์ชวลแมชชีน ซึ่งทำให้เวอร์ชวลแมชชีนสามารถเข้าถึงจีพียูและประมวลผลคูด้าแอพพลิเคชันได้พร้อม ๆ กัน กรณีที่ทรัพยากรจีพียูมีพื้นที่เพียงพอต่อการเตรียมข้อมูลสำหรับนำไปประมวลผล อีกทั้งยังสามารถจัดคิวการทำงานให้กับเวอร์ชวลแมชชีนที่ต้องการเข้าถึงจีพียูแต่พื้นที่สำหรับให้บริการบนจีพียูไม่เพียงพอให้สามารถเข้าใช้งานได้สำเร็จในคิวถัดไป