ประสิทธิภาพของซอฟต์แวร์เชิงบริการสำหรับเครื่องแม่ข่ายเกมส์ออนไลน์

Efficiency of service-based software support for online game servers

ฤทธิชัย จิตภักดีบดินทร์

Rittichai Jitpukdeebodintra

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2553

ประสิทธิภาพของซอฟต์แวร์เชิงบริการสำหรับเครื่องแม่ข่ายเกมส์ออนไลน์

Efficiency of service-based software support for online game servers

ฤทธิชัย จิตภักดีบดินทร์

Rittichai Jitpukdeebodintra

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2553

ชื่อวิทยานิพนธ์ ประสิทธิภาพของซอฟต์แวร์เชิงบริการสำหรับเครื่องแม่ข่ายเกมส์ออนไลน์

ผู้เขียน นายฤทธิชัย จิตภักดีบดินทร์

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

……..

(ผู้ช่วยศาสตราจารย์ ดร.สุนทร วิทูสุรพจน์)

คณะกรรมการสอบ

……..…..... ประธานกรรมการ

(ดร.อนันท์ ชกสุริวงค์)

……..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุนทร วิทูสุรพจน์)

……..…..... กรรมการ

(ดร.ณรงค์ฤทธิ์ วราภรณ์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

……..

(รองศาสตราจารย์ ดร.เกริกชัย ทองหนู)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ ประสิทธิภาพของซอฟต์แวร์เชิงบริการสำหรับเครื่องแม่ข่ายเกมส์ออนไลน์

ผู้เขียน นาย ฤทธิชัย จิตภักดีบดินทร์

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2552

บทคัดย่อ

ในปัจจุบัน เครื่องแม่ข่ายเกมส์ออนไลน์มักจำเป็นต้องรองรับผู้เล่นจำนวนมากในระดับหลักพันคนพร้อมกัน ในลักษณะเช่นนี้ความรวดเร็วในการตอบกลับของโปรแกรมแม่ข่ายเป็นสิ่งที่สำคัญ โดยเฉพาะอย่างยิ่งในช่วงขณะที่มีจำนวนผู้ใช้เพิ่มขึ้นอย่างมาก วิธีการแก้ไขโดยทั่วไปนิยมใช้วิธีการกระจายภาระงานของเครื่องแม่ข่ายไปยังเครื่องคอมพิวเตอร์อื่น แม้ว่าวิธีการนี้จะสามารถแก้ไขปัญหาดังกล่าวได้ แต่ไม่ได้พิจารณานำทรัพยากรด้านการประมวลผลภายในของเครื่องแม่ข่ายมาใช้งานอย่างเต็มที่ โดยเฉพาะอย่างยิ่งหน่วยประมวลผล กราฟิก ซึ่งมีประสิทธิภาพสูงในด้านการช่วยการประมวลผลให้กับหน่วยประมวลผลกลาง ดังนั้น งานวิจัยเพื่อวิทยานิพนธ์นี้เสนอแนะให้นำหน่วยประมวลผลกราฟิกมาช่วยลดภาระงานในเบื้องต้น ก่อนที่จะกระจายภาระงานไปยังเครื่องคอมพิวเตอร์อื่นต่อไป ในที่นี้ได้นำเสนอกลไกประสานการทำงานและจัดลำดับเวลางานระหว่างหน่วยประมวลผลทั้งคู่ ผ่านทางเทคนิควิธี GPGPU (General-purpose computing on graphics processing units) ร่วมกับเทคนิคการพัฒนาโปรแกรมตามแนวทางสถาปัตยกรรมเชิงบริการ (Service-oriented architecture: SOA) นอกจากนั้น ยังได้นำเสนอผลการศึกษาประสิทธิภาพด้านเวลาในการตอบสนอง เมื่อนำหน่วยประมวลผลกราฟิกเข้าไปใช้ร่วมด้วยภายในโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายแบบต่างๆ กัน พร้อมได้นำเสนอผลการวิเคราะห์เพื่อจำแนกประเภทงานของโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่าย เพื่อการนำไปทำงานโดยหน่วยประมวลผลกราฟิกอย่างเหมาะสมอีกด้วย ดังนั้น ผลการวิจัยนี้จึงไม่เพียงจะมีประโยชน์ต่อนักพัฒนาเกมส์ออนไลน์ที่ต้องการเพิ่มประสิทธิภาพของเกมส์ตนเอง แต่ยังส่งผลต่อไปถึงประโยชน์ต่อวงการเกมส์ออนไลน์โดยรวมด้วย

คำสำคัญ: เกมส์ออนไลน์, สถาปัตยกรรมเชิงบริการ, การใช้ประโยชน์หน่วยประมวลผลกราฟิก

Thesis Title	Efficiency of service-based software support for online game server
Author	Mr. Rittichai Jitpukdeebodintra
Major Program	Computer Engineering
Academic Year	2009

ABSTRACT

At present, it is very common that an online game may be served for hundreds or even thousands of concurrent players at same time. In this regards, the response time of multiplayer online games is crucial, especially during the high load of massive players. Hence, in order to lighten the load, the popular approach is often done by upgrading the current game server machine with a higher performance one. While it is easy and straightforward, this approach demands a new machine to be invested without utilizing some local resources, especially Graphic Processor Unit (GPU), which are high performance processor. The work in this thesis argue to employ the GPU for reducing the Central Processor Unit (CPU) workload at initial on the online game server, before attempting to distribute the current workload to the other machine as usual. Here, we propose an extra software serving for cooperating and scheduling of tasks between these two processors via the General-purpose computing on graphics processing units (GPGPU) and the SOA (Service-oriented architecture) approaches. In addition, we show the performance study of various kinds of online games running on the game servers, and suggest the classification of online game servers that are suitable for utilizing the GPU. This will not only give benefits to online game developers, but also the game industry as a whole.

Key Words: Online Game, Service Oriented Architecture, Graphic processor utilization

กิตติกรรมประกาศ

ขอแสดงคำขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. สุนทร วิทูสุรพจน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก ที่ได้กรุณาอุทิศเวลาให้คำปรึกษา ให้การสนับสนุนการทำวิจัย กรุณาอุทิศเวลาให้คำปรึกษา แนะนำความรู้ในด้านการทำวิจัย เอกสาร ข้อมูลต่างๆเป็นอย่างดี รวมทั้งแนวความคิดและกำลังใจในการแก้ปัญหาตลอดจนตรวจทานแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบพระคุณ ดร. อนันท์ ชกสุริวงค์ ประธานกรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาให้คำปรึกษาคำแนะนำ และให้การช่วยเหลือแก้ไขปัญหาในงานวิจัยรวมถึงเวลาเป็นกรรมการสอบวิทยานิพนธ์และตรวจทานวิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบพระคุณ ดร. ณรงค์ฤทธิ์ วราภรณ์ กรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาให้คำปรึกษาคำแนะนำ และให้การช่วยเหลือแก้ไขปัญหาในงานวิจัย

ขอขอบคุณ พี่ๆ เพื่อน ๆ และน้องๆ ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ ทุกท่านที่ได้ให้คำแนะนำ คำปรึกษา และกำลังใจที่ดีมาโดยตลอดโดยเฉพาะกลุ่มงานวิจัยห้อง WIG ซึ่งมี นายกิติศักดิ์ วัฒนกุล นายกิตติ เชี่ยวชาญ นายจักรพันธ์ สัวบุตร และ นายพิทักษ์ เศวตสุนทร

ขอขอบพระคุณ คณาจารย์และเจ้าหน้าที่ในภาควิชาวิศวกรรมคอมพิวเตอร์ทุกๆท่าน ที่ให้ความช่วยเหลือในด้านต่างๆมาโดยตลอด จนกระทั่งงานสำเร็จลุล่วง

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ที่ให้ความช่วยเหลือด้านการประสานงานต่าง ๆ

สุดท้ายนี้ ข้าพเจ้าขอน้อมรำลึกถึงพระคุณของบิดา ดร. สุรชัย จิตภักดีบดินทร์ มารดา ผู้ช่วยศาสตราจารย ดร. สมฤทัย จิตภักดีบดินทร์ และน้องชาย นาย ชัยยฤทธิ์ จิตภักดับดินทร์ ที่ส่งเสริมสนับสนุน ให้คำแนะนำ ให้คำปรึกษา ให้กำลังใจ และทุนทรัพย์แก่ข้าพเจ้าตลอดมาจนกระทั่งทำให้ข้าพเจ้าประสบความสำเร็จ

ฤทธิชัย จิตภักดีบดินทร์

สารบัญ

หน้า

สารบัญ (6)

รายการภาพประกอบ (10)

รายการตาราง (11)

1. บทนำ 1

1.1 ความสำคัญและที่มาของวิทยานิพนธ์..... 1

1.2 วัตถุประสงค์ของวิทยานิพนธ์..... 2

1.3 ขอบเขตของวิทยานิพนธ์..... 3

1.4 ขั้นตอนการดำเนินงาน..... 3

1.5 ประโยชน์ที่คาดว่าจะได้รับ..... 4

2. ทฤษฎีและหลักการ 5

2.1 เกริ่นนำ..... 5

2.2 การกระจายภาระงานเกมส์ออนไลน์โดยใช้หน่วยประมวลผลกราฟิก..... 5

2.2.1 กลไกทำงานเพื่อการประมวลผลบนหน่วยประมวลผลกราฟิก..... 5

2.2.2 งานวิจัยที่สนับสนุนทฤษฎีข้างต้น..... 6

2.3 กลไกประสานการเชื่อมต่อระหว่างโปรแกรมที่พัฒนาจากภาษาต่างชนิดกัน..... 7

2.3.1 กลไกทำงานเว็บเซอร์วิส..... 8

2.3.2 งานวิจัยที่สอดคล้องแนวคิดข้างต้น..... 10

3. การออกแบบและพัฒนาระบบ 12

3.1 เกริ่นนำ..... 12

3.2 ประเด็นปัญหาในด้านการออกแบบที่เกี่ยวข้อง..... 12

3.2.1 ปัญหาการสร้างฟังก์ชั่นซอฟต์แวร์เว็บเซอร์วิสสำหรับหน่วยประมวลผลกราฟิก..... 12

3.2.2 ปัญหาด้านการจัดการภาระงานที่เกิดขึ้นพร้อมกันระหว่างหน่วยประมวลผลทั้งคู่..... 13

สารบัญ (ต่อ)

หน้า

3.3 การประมวลผลภาระงานบนหน่วยประมวลผลกราฟิก..... 15

3.4 กรณีศึกษาการพัฒนาโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายเพื่อการทดสอบ..... 22

3.4.1 การแบ่งชนิดภาระงานโปรแกรมแม่ข่ายเกมส์ออนไลน์..... 23

3.4.2 การออกแบบและพัฒนาตัวอย่างบริการเกมส์ที่ใช้ในการทดสอบ..... 26

3.4.2.1 เกมส์ตัวอย่างที่ใช้ในการทดสอบ..... 26

3.4.2.2 เกมส์ Tic Tac Toe..... 27

3.4.2.3 เกมส์ Virtual online stock exchange..... 34

4. ผลการทดสอบ 42

4.1 เกริ่นนำ..... 42

4.2 ฮารด์แวร์ที่ใช้ในการทดสอบ..... 42

4.3 การทดสอบเกมส์ออนไลน์แบบผู้เล่นคนเดียว..... 43

4.3.1 การทดลองด้านประสิทธิภาพในการประมวลผล 1 ภาระงาน..... 44

4.3.1.1 วัตถุประสงค์ในการทดสอบ..... 44

4.3.1.2 วิธีการทดสอบ..... 44

4.3.1.3 ผลการทดสอบ..... 48

4.3.1.4 สรุปผลการทดสอบ..... 48

4.3.2 การใช้ทรัพยากรหน่วยประมวลผลกลาง..... 49

4.3.2.1 วัตถุประสงค์ในการทดสอบ..... 49

4.3.2.2 วิธีการทดสอบ..... 49

4.3.2.3 ผลการทดสอบ..... 51

4.3.2.4 สรุปผลการทดสอบ..... 52

4.4 เกมส์ออนไลนในรูปแบบที่มีผู้เล่นพร้อมกันหลายคน..... 52

4.4.1 การทดสอบประสิทธิภาพของเวลาตอบสนองของโปรแกรม..... 52

4.4.1.1 วัตถุประสงค์ในการทดสอบ..... 52

สารบัญ (ต่อ)

หน้า

4.4.1.2 วิธีการทดสอบ..... 52
4.4.1.3 ผลการทดสอบ..... 53
4.4.1.4 สรุปผลการทดสอบ..... 56
5. บทสรุปและข้อเสนอแนะ 57
5.1 ผลสรุปจากวัตถุประสงค์งานวิจัย..... 57
5.2 ข้อเสนอแนะ..... 60
เอกสารอ้างอิง 61
ภาคผนวก 63
ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์..... 64
ประวัติผู้เขียน 76

รายการรูปภาพ

หน้า

รูปที่ 2.1 โครงสร้างของหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิก..... 6

รูปที่ 2.2 กลไกทำงานแบบเว็บเซอร์วิส ตามมาตรฐาน และ แบบไม่ต้องใช้ Service Broker... 9

รูปที่ 2.3 การเชื่อมต่อระหว่างกันของบริการด้วยเทคโนโลยีเว็บเซอร์วิส..... 10

รูปที่ 3.1 แนวคิดโดยรวมของการปรับปรุงประสิทธิภาพเครื่องแม่ข่ายเกมส์ออนไลน์..... 12

รูปที่ 3.2 สถาปัตยกรรมเว็บเซอร์วิสของโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายที่ออกแบบ..... 13

รูปที่ 3.3 แผนภาพขั้นตอนการทำงานของหน่วยบริการกลาง..... 14

รูปที่ 3.4 ขั้นตอนการทำงานของสถาปัตยกรรมที่ออกแบบเมื่อมีการใช้งานหน่วยประมวลผลกราฟิก..... 15

รูปที่ 3.5 แผนภาพในการประมวลผลโปรแกรม โดยใช้หน่วยประมวลผลกราฟฟิก และโดยใช้หน่วยประมวลผลกลาง..... 16

รูปที่ 3.6 การแบ่งจำนวนงานบนหน่วยประมวลผลกราฟิก..... 20

รูปที่ 3.7 Memory Hierarchy..... 21

รูปที่ 3.8 ลำดับงานในการดำเนินงานเพื่อกรณีศึกษาโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่าย..... 23

รูปที่ 3.9 การจำแนกชนิดของภาระงานของโปรแกรมแม่ข่ายเกมส์ออนไลน์..... 24

รูปที่ 3.10 แผนภาพการทำงานของโปรแกรมแม่ข่ายเกมส์ Tic Tac Toe ออนไลน์..... 28

รูปที่ 3.11 หมายเลขตำแหน่งบนตารางของเกมส์ Tic Tac Toe..... 29

รูปที่ 3.12 แผนภาพการทำงานของบริการที่ 2 บนหน่วยประมวลผลกลาง..... 31

รูปที่ 3.13 แผนภาพการทำงานของบริการที่ 2 บนหน่วยประมวลผลกราฟิก..... 32

รูปที่ 3.14 ตำแหน่งที่ NPC จะเดินเมื่อ $pnt=1..... 33

รูปที่ 3.15 แผนภาพการทำงานของโปรแกรมแม่ข่ายเกมสจำลองสถานะการซื้อขายหุ้นในตลาดหลักทรัพย์..... 34

รูปที่ 3.16 การทำงานของบริการค้นหารายชื่อหุ้นบนหน่วยประมวลผลกลาง..... 37

รูปที่ 3.17 การทำงานของบริการค้นหารายชื่อหุ้นบนหน่วยประมวลผลกราฟิก..... 38

รูปที่ 3.18 การทำงานของบริการเพิ่มคะแนนบนหน่วยประมวลผลกลาง..... 39

รายการรูปภาพ (ต่อ)

หน้า

รูปที่ 3.19 การทำงานของบริการเพิ่มคะแนนบนหน่วยประมวลผลกราฟิก..... 40

รูปที่ 4.1 รายละเอียดของอุปกรณ์ที่ใช้ในการทดสอบผ่านโปรแกรม CPU-Z และ GPU-Z..... 41

รูปที่ 4.2 ช่วงของการจับเวลาในการทดสอบที่ 4.2.1..... 44

รูปที่ 4.3 โครงสร้างของโปรแกรมแม่ข่ายแบบทุกบริการทำงานบนหน่วยประมวลผลกลาง.. 45

รูปที่ 4.4 โครงสร้างของโปรแกรมแม่ข่ายรูปแบบที่ 2..... 46

รูปที่ 4.5 โครงสร้างของโปรแกรมแม่ข่ายรูปแบบที่ 3..... 46

รูปที่ 4.6 โครงสร้างของโปรแกรมแม่ข่ายรูปแบบที่ 4..... 47

รูปที่ 4.7 การบันทึกการใช้ทรัพยากรหน่วยประมวลผลกลาง ในการทดสอบที่ 4.3.2..... 50

รูปที่ 4.8 โครงสร้างการทำงานของโปรแกรมจำลองคำสั่งสำหรับการทดสอบที่ 4.3.2..... 50

รูปที่ 4.9 การใช้ทรัพยากรหน่วยประมวลผลกลาง เมื่อรันโปรแกรม 100 และ 1,000 คำสั่ง.... 51

รูปที่ 4.10 การใช้ทรัพยากรหน่วยประมวลผลกลาง ในการทดสอบที่ 4.4.1..... 53

รูปที่ 4.11 โครงสร้างการทำงานของโปรแกรมจำลองคำสั่งสำหรับการทดสอบที่ 4.4.1..... 53

รูปที่ 4.12 ผลการทดสอบประสิทธิภาพเวลาตอบสนองบริการจัดเรียงคะแนน..... 54

รูปที่ 4.13 ผลการทดสอบประสิทธิภาพเวลาตอบสนองบริการค้นหารายชื่อหุ้น..... 55

รูปที่ 4.14 ผลการทดสอบประสิทธิภาพเวลาตอบสนองบริการปรับปรุงสถานะของผู้เล่น..... 55

รายการตาราง

หน้า

ตารางที่ 1 ผลจากการทดลองที่ 4.3.1.3 ด้านประสิทธิภาพในการประมวลผลภาระงาน..... 48

ตารางที่ 2 ข้อเสนอแนะจากผลการทดสอบ..... 59

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของวิทยานิพนธ์

ปัจจุบันอุตสาหกรรมเกมส์ออนไลน์ได้รับความนิยมเพิ่มมากขึ้นอย่างรวดเร็วจากเดิมเกมส์ออนไลน์มีผู้เล่นพร้อมกันจำนวนไม่มากนัก จนปัจจุบันกลายเป็นเรื่องปกติที่เกมส์ออนไลน์หนึ่งๆ จะต้องรองรับผู้เล่นในหลักพันหรือหลักหมื่นคนพร้อมๆกัน ซึ่งทำให้เกิดปัญหาในการให้บริการเกมส์ออนไลน์โดยเฉพาะอย่างยิ่งปัญหาในด้านประสิทธิภาพการตอบสนองของเกมส์ อันเนื่องมาจากความไม่เพียงพอของทรัพยากรหน่วยประมวลผล [1] ซึ่งในปัจจุบันมักจะใช้วิธีการกระจายภาระงานของเครื่องแม่ข่ายให้บริการเกมส์ออนไลน์ไปยังเครื่องแม่ข่ายอื่นๆ เช่น ในงานวิจัย [4] ได้กล่าวถึงการนำเทคโนโลยีกริด (Grid technology) เข้ามาใช้งานเพื่อช่วยการกระจายภาระงานไปยังคอมพิวเตอร์จำนวนหลายตัว สำหรับเกมส์ออนไลน์ Final Fantasy XI ของบริษัท SquareEnix เป็นต้น วิธีการนี้สามารถแก้ไขปัญหาดังกล่าวได้อย่างมีประสิทธิภาพ แต่ทว่าวิธีการนี้จำเป็นต้องมีการลงทุนในส่วนเครื่องแม่ข่ายเพิ่มเติมโดยที่ยังไม่ได้ใช้ทรัพยากรของเครื่องแม่ข่ายอย่างเต็มที่ โดยเฉพาะอย่างยิ่งหน่วยประมวลผลกราฟิก (Graphic Processing Unit : GPU) และหน่วยความจำของหน่วยประมวลผลกราฟิก วิทยานิพนธ์นี้จึงมุ่งที่จะนำเอาหน่วยทำงานทั้งคู่มาใช้ในการจัดการปัญหาข้างต้นที่พบในเครื่องให้บริการเกมส์ออนไลน์ โดยใช้เทคนิคขั้นสูงทางวิศวกรรมซอฟต์แวร์ 2 ประการ คือ

1) เทคนิค GPGPU (General-purpose computing on graphics processing units) [2] ซึ่งเป็นการพัฒนาโปรแกรมที่นำหน่วยประมวลผลกราฟิกมาใช้ในการประมวลผล คู่ขนานไปการประมวลผลโปรแกรมในหน่วยประมวลผลกลาง (Central Processing Unit : CPU) เช่นที่พบในการประมวลผลแบบพื้นฐานทั่วไป ซึ่งมีข้อดีหลายประการ เช่น สามารถคำนวนแบบขนานพร้อมๆกันเป็นจำนวนมากได้ อันเนื่องมาจากโครงสร้างของหน่วยประมวลผลกราฟิกที่มีจำนวน Streaming Processor จำนวนมาก (ยกตัวอย่างเช่นในหน่วยประมวลผลกราฟิก รุ่น GeForce GTX 480 นั้นจะมี Stream Processor ถึง 480 ชุด) ซึ่งจะส่งผลทำให้ประสิทธิภาพของโปรแกรม

ที่ใช้เทคนิค GPGPU นั้นดีขึ้นกว่าโปรแกรมที่ประมวลผลโดยใช้หน่วยประมวลผลกลางแต่เพียงอย่างเดียวมาก

2) เทคนิคการโปรแกรมตามแนวทางสถาปัตยกรรมเชิงบริการ (Service-oriented architecture หรือ SOA) [3] ซึ่งออกแบบให้โปรแกรมสามารถถูกเรียกใช้งานได้ในลักษณะของ “การให้บริการ” เพื่อประโยชน์ทั้งในด้านของการเรียกใช้งานจากโปรแกรมอื่นๆ ได้อย่างอิสระ โดยที่ไม่จำเป็นต้องเป็นภาษาในการโปรแกรมที่เหมือนกันได้ เพียงแต่ทราบตำแหน่งที่อยู่ของโปรแกรม และตัวแปรที่ต้องการเท่านั้น ก็สามารถเรียกใช้งานได้ (ซึ่งใช้ศัพท์เทคนิคว่า “การเชื่อมต่อแบบหลวมๆ หรือ Loosely couple interface”) ดังนั้น แนวทางของสถาปัตยกรรม SOA นี้ จึงน่าจะเหมาะสมที่จะนำไปใช้ในสนับสนุนการสั่งหรือควบคุมการประมวลระหว่างโปรแกรมที่ทำงานอยู่บนหน่วยประมวลผลกลางและที่อยู่บนหน่วยประมวลผลกราฟิก ซึ่งอาจพัฒนาขึ้นด้วยภาษาในการโปรแกรมที่แตกต่างกัน

โดยความสำคัญแล้ว วิทยานิพนธ์นี้จะมุ่งศึกษาการเพิ่มประสิทธิภาพในการให้บริการของเครื่องแม่ข่าย เมื่อมีการแบ่งส่วนประกอบภายในโปรแกรมเกมส์ทางส่วนของแม่ข่ายออกเป็นให้มีลักษณะเป็น “บริการงานสนับสนุน” ต่างๆ และย้ายไปประมวลผลยังหน่วยประมวลผลกราฟิกแทน เพื่อเป็นการบรรเทาภาระงานในการส่วนของโปรแกรมด้านอื่นๆ ที่ยังประมวลผลอยู่ ณ หน่วยประมวลผลกลางเช่นเดิม ซึ่งคาดว่าน่าจะมีศักยภาพในการช่วยให้เครื่องแม่ข่ายสามารถรองรับจำนวนผู้ใช้งานได้เพิ่มมากขึ้น นอกจากนั้น ยังส่งผลทำให้ส่วนของโปรแกรมเกมส์ทางด้านแม่ข่ายสามารถจัดการแก้ไข/บำรุงรักษาได้ง่ายขึ้น จากประโยชน์ของแนวทางการโปรแกรมตามแนวทางสถาปัตยกรรมเชิงบริการอีกด้วย จึงคาดว่าน่าจะมีประโยชน์อย่างยิ่งต่ออุตสาหกรรมเกมส์ออนไลน์

1.2 วัตถุประสงค์ของวิทยานิพนธ์

1. เพื่อศึกษาประสิทธิภาพการจัดการปัญหาความคับคั่งที่เครื่องแม่ข่ายให้บริการเกมส์ออนไลน์โดยใช้แนวทางในการพัฒนาโปรแกรมด้วยสถาปัตยกรรมเชิงบริการและเทคนิค GPGPU
2. เพื่อเปรียบเทียบลักษณะของงานประเภทต่างๆ ของเกมส์ออนไลน์ที่เหมาะสม สำหรับการนำไปประมวลผลบนหน่วยประมวลกราฟิก

1.3 ขอบเขตของวิทยานิพนธ์

1. ศึกษาแนวทางการโปรแกรมด้วยสถาปัตยกรรมเชิงบริการ และเทคนิควิธี GPGPU เพื่อการแยกส่วนบริการโปรแกรมแม่ข่ายสำหรับเกมส์ออนไลน์แบบต่างๆ ที่นำมาใช้การศึกษาวิจัย
2. ศึกษาเปรียบเทียบประสิทธิภาพด้านความรวดเร็วในการตอบสนองผู้ใช้งานโปรแกรมเกมส์ออนไลน์ทางด้านแม่ข่ายให้บริการ ระหว่างแบบพื้นฐานซึ่งประมวลผลด้วยหน่วยประมวลผลกลางโดยลำพัง กับแบบที่ใช้หน่วยประมวลกราฟิกร่วมด้วย ในสภาวะแวดล้อมที่มีภาระงานจากจำนวนของผู้เล่นที่เพิ่มมากขึ้นเป็นลำดับ จากเกมส์ออนไลน์ 2 ประเภท คือ เกมส์ออนไลน์แบบผู้เล่นคนเดียว และ แบบผู้เล่นพร้อมกันหลายคน
3. ศึกษาความเหมาะสมในการใช้เทคนิค GPGPU กับลักษณะงานโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายชนิดต่างๆ โดยใช้ผลทดสอบความเร็วในการตอบสนองและข้อจำกัดของเทคนิค GPGPU เป็นเกณฑ์ตัดสิน

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษางานวิจัยที่ใช้ประโยชน์จากสถาปัตยกรรมเชิงบริการ และเทคนิค GPGPU ในบริบทของเกมส์ออนไลน์ทางด้านแม่ข่าย
2. ศึกษาสถาปัตยกรรมของโปรแกรมเกมส์ออนไลน์ทางด้านแม่ข่ายที่จะนำมาใช้ในการวิจัย เพื่อจำแนกองค์ประกอบของฟังก์ชั่นงานบางส่วนให้มีลักษณะเป็นบริการงาน
3. พัฒนา และทดสอบประสิทธิภาพในการใช้ประโยชน์จากเทคนิค GPGPU ของลักษณะงานภายในเกมส์ออนไลน์แบบต่างๆ กัน พร้อมศึกษาประสิทธิภาพด้านความรวดเร็วในการตอบสนองผู้ใช้งาน ระหว่างบริการโปรแกรมแบบที่ใช้ CPU กับแบบที่ใช้ GPU และ CPU (ผ่านทางเทคนิคการโปรแกรมแบบ GPGPU)
4. รวบรวมผลการทดสอบ สรุปผล และจัดทำวิทยานิพนธ์ฉบับสมบูรณ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เทคนิคการใช้ฟังก์ชั่นไลบารี่พื้นฐานสำหรับหน่วยประมวลผลกราฟิก ในการพัฒนาเป็นกลไกเชิงบริการตามมาตรฐานเว็บเซอร์วิส
2. แนวทางที่เหมาะสมในการกระจายภาระงานเกมส์ออนไลน์ส่วนแม่ข่ายจากหน่วยประมวลผลกลางไปยังหน่วยประมวลผลกราฟิก โดยใช้ประโยชน์จากกลไกทำงานเชิงบริการและเทคนิคการโปรแกรมแบบ GPGPU
3. ลักษณะการจำแนกประเภทงานภายในเกมส์ออนไลน์ที่ทำงานภายในส่วนของเครื่องแม่ข่าย ซึ่งเป็นประโยชน์ต่อผู้พัฒนาโปรแกรมเกมส์ออนไลน์ ซึ่งประสงค์จะใช้ประโยชน์จากหน่วยประมวลผลกราฟิก

บทที่ 2
ทฤษฎีและหลักการ

2.1 เกริ่นนำ

จากปัญหาความไม่เพียงพอของทรัพยากรคอมพิวเตอร์แม่ข่ายในการให้บริการเกมส์ออนไลน์ภายใต้ช่วงเวลาที่มีผู้ใช้งานพร้อมกันจำนวนมาก ที่ส่งผลกระทบต่อเวลาในการตอบสนองของการให้บริการเกมส์ออนไลน์ดังที่กล่าวในบทที่ผ่านมาแล้วนั้น จะได้นำเสนอแนวทางการแก้ไขปัญหาที่ได้ใช้ในงานวิจัยเพื่อวิทยานิพนธ์นี้เป็นลำดับๆ ไป โดยเริ่มต้นจะเป็นการกล่าวถึงแนวทางที่จะนำมาใช้เพื่อกระจายภาระงานของหน่วยประมวลผลกลาง (CPU) ไปประมวลผลยังหน่วยประมวลผลกราฟิก (GPGPU) พร้อมกับทบทวนวรรณกรรมที่เกี่ยวข้อง หลังจากนั้นจึงเป็นการกล่าวถึงแนวทางของการพัฒนาโปรแกรมด้วยแนวทางสถาปัตยกรรมเชิงบริการ (SOA) ซึ่งจะช่วยให้โปรแกรมทั้งส่วนที่ทำงานอยู่บนหน่วยประมวลผลทั้งสองแบบสามารถจะติดต่อกันได้โดยอิสระมากขึ้น แม้ว่าจะพัฒนาขึ้นจากภาษาการโปรแกรมที่แตกต่างประเภทกันก็ตาม พร้อมกับทบทวนวรรณกรรมที่เกี่ยวข้องเช่นกัน

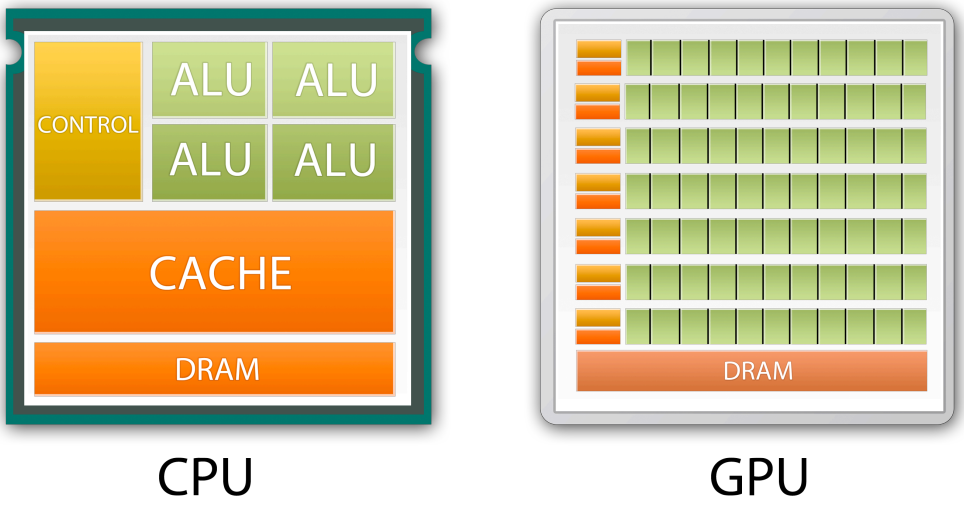
2.2 การกระจายภาระงานเกมส์ออนไลน์โดยใช้หน่วยประมวลผลกราฟิก

การเพิ่มจำนวนการรองรับผู้ใช้งานของเครื่องแม่ข่ายเกมส์ออนไลน์นั้น ในปัจจุบัน มักจะใช้วิธีกระจายภาระงานไปยังเครื่องแม่ข่ายอื่นๆ เช่น ในงานวิจัย [11] ได้นำเทคโนโลยีกริดเข้าใช้เพื่อช่วยการกระจายภาระงานให้กับคอมพิวเตอร์จำนวนหลายตัว อย่างไรก็ตามวิธีการนี้มีการลงทุนเพิ่มเติมเพื่อนำคอมพิวเตอร์อื่นๆ เข้ามาช่วยงาน แต่ก็ยังไม่ได้พิจารณานำทรัพยากรที่มีอยู่เดิมในเครื่องคอมพิวเตอร์แล้ว เช่น หน่วยประมวลผลกราฟิก (GPU) พร้อมหน่วยความจำ เป็นต้น มาใช้งานเพื่อการเพิ่มประสิทธิภาพเป็นลำดับต้นๆ

2.2.1 กลไกทำงานเพื่อการประมวลผลบนหน่วยประมวลผลกราฟิก

การนำหน่วยประมวลผลทางกราฟิกเข้ามาช่วยในบรรเทาภาระงานให้กับหน่วยประมวลผลกลางของเครื่องคอมพิวเตอร์นั้น แต่เดิมนิยมใช้กันมากเฉพาะการช่วยงานด้านแสดงผลกราฟิก

อย่างไรก็ตาม เมื่อมีความแพร่หลายของกราฟิกความละเอียดสูงในโปรแกรมเกมส์แบบสามมิติ จึงได้รับการปรับเปลี่ยนเชิงสถาปัตยกรรมให้กลายเป็นหน่วยประมวลผลอเนกประสงค์แบบมีสมรรถนะสูง และมีคุณลักษณะที่สามารถประมวลผลแบบขนานพร้อมๆ กันเป็นจำนวนมากได้ [2] เนื่องจากมีของหน่วยประมวลผลทางคณิตศาสตร์ (Arithmetic Logic Unit หรือ ALU) จำนวนมากอยู่ภายในดังได้แสดงการเปรียบเทียบกับหน่วยประมวลผลกลางแบบหลายแกนประมวลผล (Multi-core CPU) ไว้ในรูปที่ 2.1 โดยจะสังเกตุได้ว่าองค์ประกอบพื้นฐาน (ได้แก่ หน่วยความจำแบบ DRAM, หน่วยความจำแคช (Cache), หน่วยทำงาน ALU และส่วนควบคุม (Control)) เหมือนกัน เพียงแต่จำนวนของ ALU ของหน่วยประมวลผลกราฟิกแตกต่างกันอย่างเห็นได้ชัดเจน ดังนั้น แนวความคิดในการย้ายภาระงานบางส่วนของโปรแกรมเกมส์มาประมวลผลยังหน่วยประมวลกราฟิก ซึ่งเรียกว่า GPGPU (General-Purpose computing on Graphics Processing Units) จึงน่าจะเพิ่มประสิทธิภาพด้านความเร็วในการประมวลผลของเครื่องแม่ข่ายได้ดียิ่งขึ้น จากผลของ ALU จำนวนมหาศาลที่มีอยู่ภายในหน่วยประมวลผลกราฟิกนั่นเอง



รูปที่ 2.1 โครงสร้างของหน่วยประมวลผลกลาง (CPU) และหน่วยประมวลผลกราฟิก (GPU)

2.2.2 งานวิจัยที่เกี่ยวข้อง

ตัวอย่างของงานวิจัยบางส่วนที่รายงานประสิทธิภาพด้านความเร็วในการประมวลผลที่เพิ่มสูงขึ้น เมื่อนำหน่วยประมวลผลกราฟิก (GPU) เข้ามาร่วมทำงานด้วยกับหน่วยประมวลผลกลาง เช่น

2.2.2.1 งานวิจัยเรื่อง “GPU-based Desktop Supercomputing” โดย James Glenn-Anderson และคณะ [2]

งานวิจัยชิ้นนี้ได้ศึกษาและทดสอบประสิทธิภาพของการนำเอา Graphics Processor Unit (GPU) มาใช้ในการช่วยประมวลผลงานต่างๆ โดยที่ศึกษาบน API ที่มีการนำมาใช้งานจริงแล้ว เช่น Compute Unified Device Architecture (CUDA) ของบริษัท NVIDIA และ Data Parallel Virtual Machine (DPVM) ของบริษัท ATI โดยผลที่ได้มีความน่าสนใจอย่างยิ่งคือมีการเพิ่มประสิทธิภาพของโปรแกรมเมื่อเทียบกับการรันโปรแกรมบนหน่วยประมวลผลกลางอย่างเดียวเป็นอย่างมาก โดยผลจากการทดลอง ได้แสดงให้เห็นว่าสามารถเพิ่มความเร็วจากการรันแบบปกติได้สูงมากขึ้นในช่วงตั้งแต่ 17 ถึง 1040 เท่า งานวิจัยนี้จึงสนับสนุนแนวความคิดของงานในวิทยานิพนธ์นี้ได้เป็นอย่างดี

2.2.2.2 งานวิจัยเรื่อง “In-Memory Grid Files on Graphics Processors” โดย Ke Yang และคณะ [5]

งานวิจัยชิ้นนี้ได้ศึกแนวคิดในการนำเอาหน่วยประมวลผลทางกราฟิกมาใช้ในการประมวลผลข้อมูลจำนวนมาก เช่น การ Query ข้อมูล การเปรียบเทียบข้อมูล เป็นต้น โดยจากการทดสอบประสิทธิภาพด้านความเร็วในการประมวลผลที่ได้นั้น เพิ่มขึ้นอย่างมาก ซึ่งสนับสนุนแนวความคิดของงานในวิทยานิพนธ์นี้ได้เป็นอย่างด

2.3 กลไกประสานการเชื่อมต่อระหว่างโปรแกรมที่พัฒนาจากภาษาต่างชนิดกัน

เนื่องจากเทคนิควิธีที่กระจายโปรแกรมบางส่วนของหน่วยประมวลผลกลางให้ไปประมวลผลยังหน่วยประมวลผลกราฟิกนั้น จำเป็นต้องมีส่วนของโปรแกรมที่ทำหน้าที่ควบคุมการประสานงานระหว่างหน่วยประมวลผลทั้งคู่ ดังนั้น จึงอาจเกิดปัญหาหรือความยุ่งยากในการประสานงานขึ้นได้ ในกรณีที่โปรแกรมสำหรับหน่วยประมวลผลทั้งคู่นั้นได้รับพัฒนาด้วยภาษาที่ต่างชนิดกัน จากข้อจำกัดของตัวแปลภาษาชั้นสูงที่มีใช้งานบนหน่วยประมวลผลกราฟิก [6]

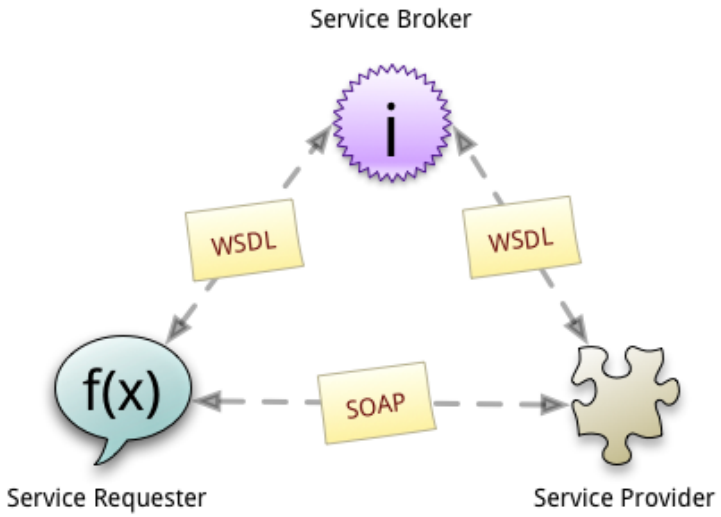
แนวคิดของการออกแบบโปรแกรมตามสถาปัตยกรรมเชิงบริการ (SOA) สามารถนำมาใช้ในการแก้ไขปัญหานี้ได้ เนื่องจากมุ่งเน้นใช้กลไกที่มีพื้นฐานจากภาษา XML ในการแลกเปลี่ยนข้อมูลระหว่างคู่โปรแกรมที่ต้องการสื่อสารกันได้อย่างอิสระ ในลักษณะของการประสานงานกัน

แบบหลวมๆ (Loosely coupled interface) แม้ว่าโปรแกรมเหล่านั้นจะพัฒนาขึ้นมาจากภาษาคอมพิวเตอร์ที่แตกต่างกันก็ตาม ตัวอย่างของเทคโนโลยีที่ใช้แนวคิดสถาปัตยกรรมเชิงบริการ เช่น COBRA, XML-RPC, และ Web Service เป็นต้น อย่างไรก็ตาม ในวิทยานิพนธ์นี้จะเน้นเฉพาะเทคโนโลยีเว็บเซอร์วิส (Web Service) เนื่องจากความแพร่หลายของการใช้ในงานวิจัยด้านการพัฒนาเกมส์ออนไลน์แบบเครือข่ายที่มีการเผยแพร่ (เช่น [7, 8]) อีกทั้งยังมีความสะดวกต่อการศึกษาวิจัยในวิทยานิพนธ์จากฟังก์ชั่นทำงานสนับสนุนการประมวลผลข้อมูลแบบ XML โดยตรง

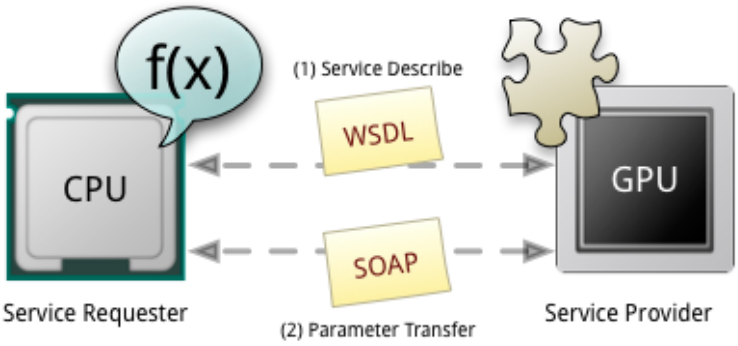
2.3.1 กลไกทำงานเว็บเซอร์วิส

เว็บเซอร์วิสเป็นกลไกทำงานมาตรฐานของ W3C (World Wide Web Consortium) [9] ตามแนวคิดของสถาปัตยกรรมเชิงบริการเพื่อทำให้โปรแกรมที่อาจต่างประเภทกันสามารถติดต่อแลกเปลี่ยนข้อมูลระหว่างกันได้ ดังนั้นในบริบทของวิทยานิพนธ์นี้ จึงทำให้ส่วนของโปรแกรมเกมส์ที่ดำเนินการภายในหน่วยประมวลผลกลางซึ่งพัฒนาขึ้นด้วยภาษา PHP สามารถสื่อสารกับส่วนของโปรแกรมซึ่งพัฒนาขึ้นด้วยภาษา CUDA [6] สำหรับดำเนินการหน่วยประมวลผลกราฟิกของผู้ผลิต NVIDIA ได้ อย่างไรก็ตาม เมื่ออ้างอิงกับโมเดลทำงานตามมาตรฐานแบบเว็บเซอร์วิส (ดังแสดงในรูปที่ 2.2(a)) แล้วจะไม่มีการใช้หน่วยทำงาน Service Broker ในการศึกษาวิจัยนี้ เนื่องจากหน่วยประมวลผลทั้งสองนั้นได้ทราบรายละเอียดของบริการที่ต้องการซึ่งได้กำหนดไว้ล่วงหน้าภายในโปรแกรมแล้ว (ดูรูปที่ 2.2(b))

การสื่อสารตามมาตรฐานเว็บเซอร์วิสนั้น ผู้เรียกใช้บริการ (Service Requester) จะต้องทราบรายละเอียดเกี่ยวกับบริการจากผู้ให้บริการ (Service Provider) เช่น จำนวนพารามิเตอร์ที่ต้องการให้ระบุไว้ก่อนการเรียกใช้บริการ หรือพารามิเตอร์ที่จะได้รับหลังจากเสร็จสิ้นการประมวลผลบริการ เป็นต้น ข้อมูลเหล่านี้จะกำหนดไว้ในลักษณะของเอกสาร XML ตามข้อกำหนดของภาษา WSDL (Web Services Description Language) หลังจากนี้แล้ว ผู้เรียกใช้บริการก็สามารถจะจัดการข้อมูลเพื่อการเรียกสั่งงานรวมถึงการจำแนกข้อมูลเพื่อรับข้อมูลที่ตอบกลับมาได้อย่างถูกต้อง อย่างไรก็ตาม ข้อมูลที่สื่อสารระหว่างกันนี้จะเป็นลักษณะของเอกสาร XML เช่นกัน แต่กำหนดตามภาษา SOAP (Simple Object Access Protocol)



(a)

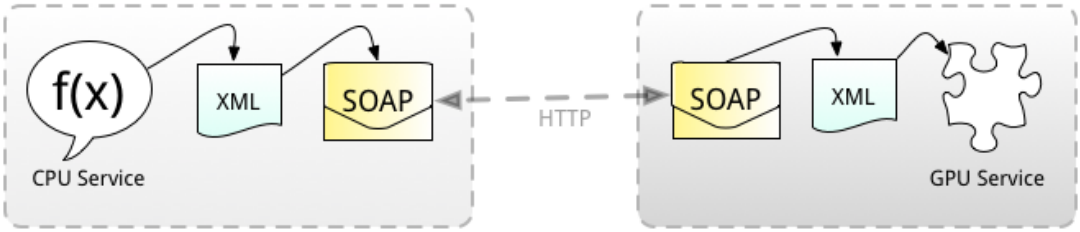


(b)

รูปที่ 2.2 กลไกทำงานแบบเว็บเซอร์วิส (a) ตามมาตรฐาน และ (b) แบบไม่ต้องใช้ Service Broker

ในรูปที่ 2.3 แสดงให้เห็นถึงการเชื่อมต่อเพื่อแลกเปลี่ยนข้อมูลตามมาตรฐานเว็บเซอร์วิส ซึ่งได้ดำเนินการในวิทยานิพนธ์นี้ กล่าวคือ ฟังก์ชั่นที่เกี่ยวข้องกับกลไกทำงานเว็บเซอร์วิสจะมีลักษณะเป็นฟังก์ชั่นที่ทำหน้าที่ช่วยการประสานงาน (Interface Function) เพื่อรับ/ส่งข้อมูลกับฟังก์ชั่นทำงานทั่วไปอีกทอดหนึ่ง ดังนั้น จากรูปที่ 3 นี้ได้ออกแบบให้ฟังก์ชั่นทำงานทั่วไปที่ต้องการใช้กลไกเว็บเซอร์วิสจะต้องจัดรูปแบบข้อมูลล่วงหน้าให้เป็นลักษณะเอกสารแบบ XML ก่อนที่จะส่งให้ฟังก์ชั่นช่วยประสานงาน ซึ่งจะหุ้มห่อข้อมูลนี้ให้เป็นไปตามรูปแบบภาษา SOAP (จึงเปรียบ

เสมือนนำข้อมูล XML มาใส่ซองจดหมายแบบ SOAP) ก่อนที่จะส่งออกไปยังบริการที่อยู่ในหน่วยประมวลผลอื่นต่อไป ในทางตรงกันข้าม ก็จะใช้ลักษณะทำงานเช่นเดียวกันนี้ในการส่งกลับข้อมูลบริการ



รูปที่ 2.3 การเชื่อมต่อระหว่างกันของบริการด้วยเทคโนโลยีเว็บเซอร์วิส

2.3.2 งานวิจัยที่สอดคล้องแนวคิดข้างต้น

ตัวอย่างของงานวิจัยบางส่วนที่นำแนวทางสถาปัตยกรรมเชิงบริการไปใช้เพื่อช่วยดำเนินการของโปรแกรมเกมส์ส่วนแม่ข่ายที่อยู่ข้ามเครื่องกัน ซึ่งแตกต่างจากงานวิจัยในวิทยานิพนธ์นี้ซึ่งใช้แนวทางของสถาปัตยกรรมนี้เช่นกัน แต่ดำเนินการด้วยกลไกทำงานเว็บเซอร์วิส และประสานงานระหว่างกันกับหน่วยบริการที่ดำเนินการโดยหน่วยประมวลผลกลางและกราฟิกที่อยู่ภายในเครื่องคอมพิวเตอร์ตัวเดียวกัน

2.3.2.1 งานวิจัยเรื่อง “Modeling System Performance in MMORPG” โดย Gao Huang และคณะ [10]

งานวิจัยชิ้นนี้นำเสนอระบบตัวอย่างของการสร้างบริการแบบ On Demand สำหรับการให้บริการเกมส์ออนไลน์ โดยออกแบบฟังกชั่นที่จะสามารถใช้งานร่วมกันได้ให้ออกมาในรูปแบบบริการ เช่น บริการการจัดการการเข้าใช้ (login management) , บริการสำหรับการพบปะกันระหว่างผู้เล่น (lobby services) หรือ บริการสำหรับการจับคู่เล่นเกมส์ (matchmaking services)จากผลการทดลองได้แสดงให้เห็นถึงการนำแนวคิดสถาปัตยกรรมเชิงบริการมาใช้ในการออกแบบสถาปัตย์กรรมเกมส์ออนไลน์ส่วนแม่ข่าย ซึ่งนอกจากจะได้แสดงให้เห็นถึงความสามารถในการเชื่อมต่อ

ระหว่างโปรแกรมที่พัฒนาจากภาษาต่างชนิดกัน ยังได้แสดงให้เห็นว่าการใช้แนวคิดแบบ Service-Oriented นี้นั้นไม่ได้ทำให้ประสิทธิภาพของเกมส์ออนไลน์นั้นลดลงไปแต่อย่างไรอีกด้วย

2.3.2.2 งานวิจัยเรื่อง "Evaluating a Middleware for Crossmedia Games" โดย FERNANDO TRINTA และคณะ [11]

งานวิจัยดังกล่าวได้แสดงให้เห็นถึงการนำเอาแนวคิดสถาปัตยกรรมเชิงบริการ มาใช้ในการพัฒนาเกมส์ออนไลน์โดยมีการแบ่งเป็นบริการ เพื่อที่จะให้สามารถสร้างเกมส์สำหรับสื่อในหลายๆ รูปแบบ จากการทดลองได้แสดงให้เห็นถึงประโยชน์ที่จะได้รับหลังจากการแบ่งหน่วยบริการของเครื่องแม่ข่ายเกมส์ออนไลน์ ในเชิงการพัฒนาเกมส์ออนไลน์บนสื่อหลายๆชนิด

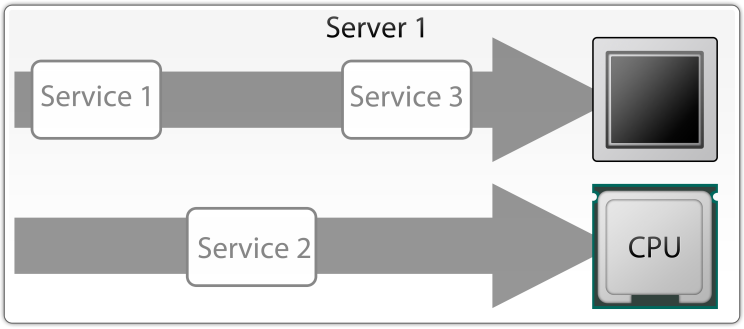
2.3.2.3 งานวิจัยเรื่อง "Building a simple yet powerful MMO game architecture" โดย Hyun Sung Chu [7]

งานวิจัยชิ้นนี้ได้แสดงให้เห็นการนำเอาเว็บเซอร์วิสมาใช้ในการออกแบบโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายเพื่อ จากการทดสอบนั้นได้แสดงให้เห็นถึงประโยชน์ในการนำเอาเว็บเซอร์วิสมาใช้ในการพัฒนานั้นทำให้สามารถกระจายภาระงานเกมส์ออนไลน์ไปยังเครื่องแม่ข่ายอื่นๆได้โดยง่าย อีกทั้งยังง่ายในการดูแลรักษาระบบอีกด้วย

บทที่ 3
การออกแบบและพัฒนาระบบ

3.1 เกริ่นนำ

จากแนวทางการกระจายภาระงานของเกมส์ออนไลน์ส่วนแม่ข่ายที่ใช้ในการดำเนินงานในวิทยานิพนธ์นี้ตามที่อธิบายในบทที่ผ่านมา (ดูแผนภาพในรูปที่ 3) ได้ให้ข้อมูลเบื้องต้น พร้อมแนวคิดที่จะนำแนวการจัดการโปรแกรมแบบเชิงบริการ (SOA) และเทคนิค GPGPU เพื่อนำบริการบางส่วนไปประมวลผลยังหน่วยประมวลกราฟิก (GPU) สำหรับในบทนี้จะเป็นการอธิบายรายละเอียดของระบบที่ได้ออกแบบขึ้นตามแนวคิดข้างต้น พร้อมตัวอย่างบริการที่จะนำไปใช้ในการทดสอบประสิทธิภาพของระบบต่อไป



รูปที่ 3.1 แนวคิดโดยรวมของการปรับปรุงประสิทธิภาพเครื่องแม่ข่ายเกมส์ออนไลน์

3.2 ประเด็นปัญหาในด้านการออกแบบที่เกี่ยวข้อง

3.2.1 ปัญหาการสร้างฟังก์ชั่นซอฟต์แวร์เว็บเซอร์วิสสำหรับหน่วยประมวลผลกราฟิก

3.2.1.1 ที่มาของปัญหา

เนื่องจากซอฟต์แวร์ NVIDIA CUDA (Compute Unified Device Architecture) [6] ซึ่งเป็นซอฟต์แวร์ที่ใช้ในการพัฒนาโปรแกรมเพื่อทำงานบนหน่วยประมวลผลกราฟิก เวอร์ชั่นที่นำมาใช้ในวิทยานิพนธ์นี้ (3.0) ยังไม่มีฟังก์ชั่นรองรับกลไกการทำงานแบบเว็บเซอร์วิสดังนั้นจึงต้องหาแนวทางในการพัฒนาฟังก์ชั่นดังกล่าวนี้

3.2.1.2 แนวคิดในการแก้ปัญหา

จากการศึกษาฟังก์ชั่นทำงานของซอฟต์แวร์ NVIDIA CUDA พบว่ามีฟังก์ชั่นสนับสนุนการทำงานประมวลผลเอกสาร XML แต่ไม่มีฟังกชั่นในการรับ/ส่งข้อมูลมาตรฐาน SOAP และ

WSDL จึงจำเป็นต้องใช้หน่วยประมวลผลกลางในการถอดความจากมาตรฐานข้างต้น มากลายเป็นข้อมูลในรูปแบบตัวแปรที่หน่วยประมวลผลกราฟิกสามารถใช้งานได้ ซึ่งจากรูปที่ 3.2 ได้พัฒนาให้อยู่ในส่วน Service Wrapper

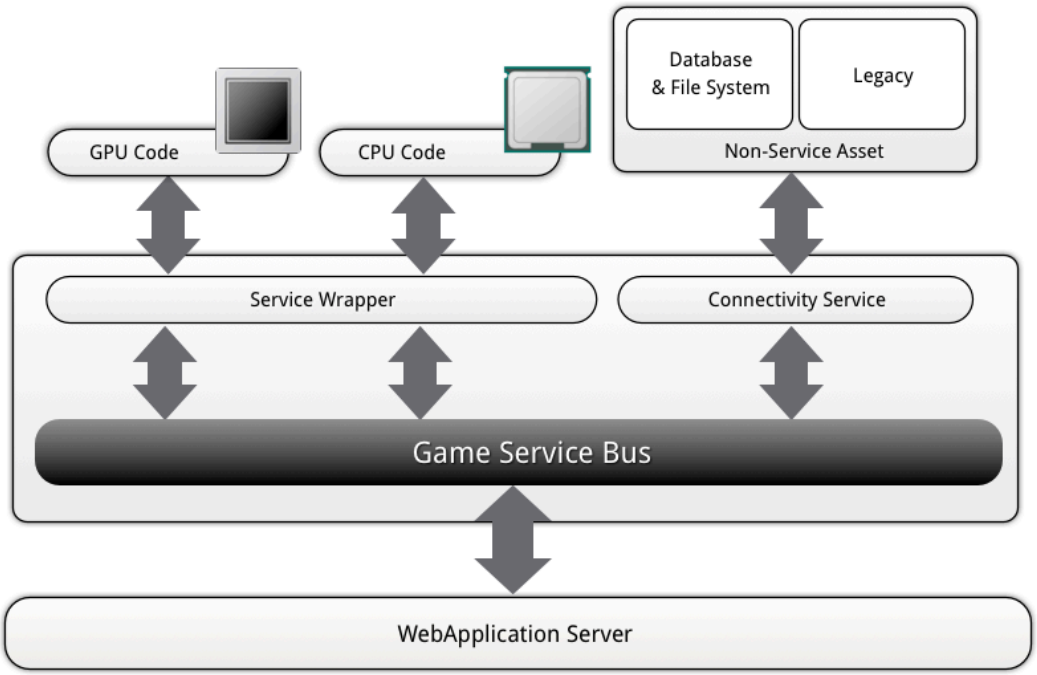
3.2.2 ปัญหาด้านการจัดการภาระงานที่เกิดขึ้นพร้อมกันระหว่างหน่วยประมวลผลทั้งคู่

3.2.2.1 ที่มาของปัญหา

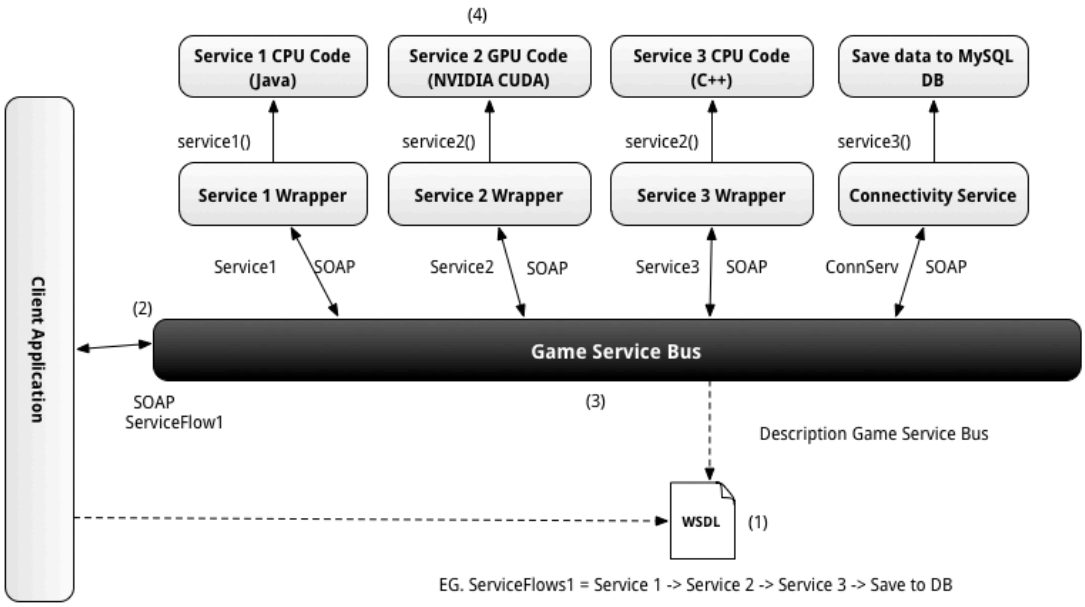
เนื่องจากอัตราเร็วในการทำงาน (Execution speed) ของหน่วยประมวลผลกลางและกราฟิกไม่เท่ากัน อาจส่งผลต่อการประสานงานระหว่างหน่วยประมวลผลทั้งคู่ ดังนั้น จึงจำเป็นต้องมีการจัดการทำงานให้สอดประสานกันระหว่างหน่วยประมวลผลทั้งคู่

3.2.2.2 แนวคิดในการแก้ปัญหา

แม้ว่าแนวทางการจัดการให้มีการประสานจังหวะการทำงานกันระหว่างหน่วยประมวลผลหลายตัวจะทำได้หลายลักษณะ แต่ในที่นี้เลือกใช้แบบพื้นฐานผ่านกลไกจัดลำดับเวลางาน (Queue scheduling) เนื่องจากง่ายต่อการพัฒนาสำหรับการทดสอบแนวความคิดเบื้องต้นในวิทยานิพนธ์นี้ ซึ่งจากรูปที่ 3.2 ได้พัฒนาให้อยู่ในส่วน Game Service Bus



รูปที่ 3.2 สถาปัตยกรรมเว็บเซอร์วิสของโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายที่ออกแบบ



รูปที่ 3.3 แผนภาพขั้นตอนการทำงานของหน่วยบริการกลาง

รูปที่ 3.3 ได้แสดงให้เห็นถึงขั้นตอนในการทำงานของหน่วยบริการกลาง ดังต่อไปนี้

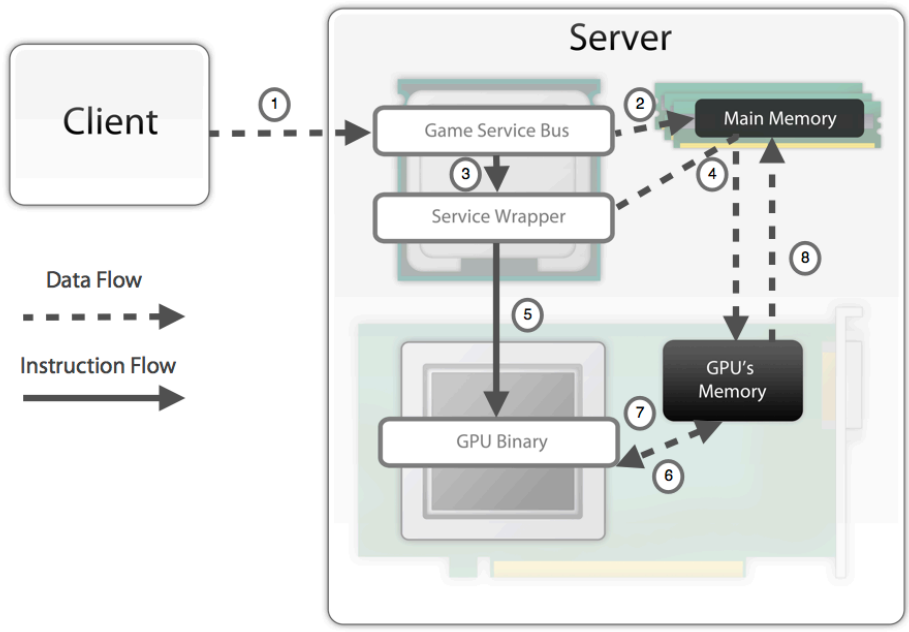
1. เครื่องลูกข่ายจะรับรู้ว่าการทำงานของบริการต่างๆจะต้องการตัวแปรใดผ่านทางไฟล์เอกสาร WSDL และส่ง input ผ่านโปรโตคอล SOAP
2. เมื่อเครื่องแม่ข่ายรับตัวแปรแล้ว GSB จะทำการเรียกใช้บริการโดยใช้วิธีการ Queue scheduling ในการจัดลำดับในการทำงาน คือ บริการจะทำงานตามลำดับขั้นตอนที่ได้ทำการตั้งค่าไว้แล้ว เช่น ในตัวอย่างจากรูปที่ 3.3 นี้ (ServiceFlow1 = service1->service2->Service3->save to DB) หมายความว่าจะประมวลผล Service1 เสร็จก่อนแล้วจึงประมวลผล Service2 เมื่อประมวลผล Service 2 เสร็จแล้วจึงประมวลผล Service3 เมื่อประมวลผล Service 3 เสร็จแล้วจึงประมวลผลบริการ save to DB เป็นต้น โดยมีการทำงานดังโค้ดต่อไปนี้

```
1.  while(servicesig==0){
2.    servicesig=0;
3.    while(servicesig1==0){
4.       servicesig1 = service1_wrapper(processor_param);
5.    }
6.    servicesig=1;
7.  }
```

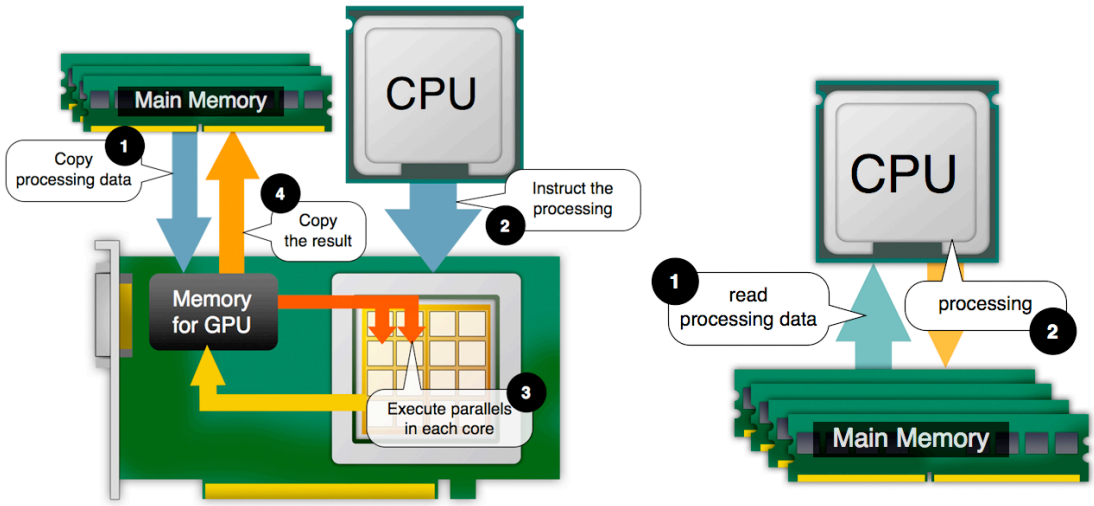
จากโค้ดการทำงานด้านบนนั้นได้แสดงให้เห็นในบรรทัดที่ 2 และ 4 ว่ามีตัวแปร servicesig และ servicesig1 ซึ่งเป็นตัวแปรสัญญาณการทำงานของโปรแกรมเมื่อมีค่า = 0 จะหมายความว่าภาระงานนั้นยังไม่ได้ประมวลผลจนแล้วเสร็จ แต่ถ้าภาระงานทำงานจนแล้วเสร็จจะต้องคืนค่า 1 กลับมา (เช่นในโค้ดข้างต้นนั้น ถ้าหากฟังกชั่น service1_wrapper() ทำงานแล้วเสร็จจะต้องส่งค่า 1 (return 1;) กลับมา) ซึ่งจะส่งผลให้ตัวแปร servicesig นี้เปลี่ยนค่าเป็น 1 แล้วโปรแกรมจะสามารถทำงานต่อจาก While loop นี้ได้

3. จากนั้นจึงส่งตัวแปรที่บริการนั้นๆต้องการผ่านทางมาตรฐาน SOAP ไปยัง Service Wrapper เพื่อจะทำการแปลงข้อมูลถอดความจากมาตรฐาน SOAP มากลายเป็นข้อมูลในรูปแบบตัวแปรที่หน่วยประมวลผลกราฟิกและหน่วยประมวลผลกลางสามารถใช้งานร่วมกันได้
4. ด้วยลักษณะเฉพาะของสถาปัตยกรรมบริการทำให้ไม่ว่าบริการนั้นๆจะถูกพัฒนาด้วยภาษาใดๆ ก็สามารถทำงานร่วมกันได้ ซึ่งในส่วนนี้กรณีที่ภาระงานทั้งหมดถูกประมวลผลบนหน่วยประมวลผลกลางการทำงานจะจบอยู่ที่การใช้หน่วยประมวลผลกลางประมวลผลบริการนั้นๆเท่านั้น ซึ่งจะแตกต่างกันกับการประมวลผลบริการบนหน่วยประมวลผลทางกราฟิก ตามที่จะได้อธิบายต่อไปในหัวข้อ 3.3

3.3 การประมวลผลภาระงานบนหน่วยประมวลผลกราฟิก



รูปที่ 3.4 ขั้นตอนการทำงานของสถาปัตยกรรมที่ออกแบบเมื่อมีการใช้งานหน่วยประมวลผลกราฟิก



รูปที่ 3.5 แผนภาพในการประมวลผลโปรแกรม (a) โดยใช้หน่วยประมวลผลกราฟฟิก (b) โดยใช้หน่วยประมวลผลกลาง

รูปที่ 3.4 และ 3.5 ได้แสดงให้เห็นถึงขั้นตอนในการทำงานของระบบเมื่อมีการประมวลบนหน่วยประมวลผลทางกราฟิกดังนี้

ขั้นตอนที่ 1 (รูปที่ 3.4 (1)) เครื่องลูกข่ายส่งคำสั่งไปยังเครื่องแม่ข่าย เครื่องแม่ข่ายจะเรียก GSB ขึ้นมาประมวลผลก่อนเป็นบริการแรก

ขั้นตอนที่ 2 (รูปที่ 3.4 (2)) GSB จะทำการเก็บค่าตัวแปรของลูกข่ายไปไว้ในหน่วยประมวลผลหลัก

ขั้นตอนที่ 3 (รูปที่ 3.4 (3)) GSB ทำการสั่งงาน Service Wrapper ให้เริ่มทำงาน

ขั้นตอนที่ 4 (รูปที่ 3.4 (4)) Service wrapper ทำการคัดลอกข้อมูลของลูกข่ายที่จำเป็นต้องใช้งานไปยังหน่วยความจำของหน่วยประมวลผลกราฟิกโดยจะใช้คำสั่งดังนี้ (จากขั้นตอนที่ 1 ในรูป 3.5 (a))

```
1.  // Initialize host array and copy it to CUDA device
2.  for (int i=0; i<N; i++) a_h[i] = (float)i;
3.  cudaMemcpy(a_d, a_h, size, cudaMemcpyHostToDevice);
4.  answer_h = (int *) malloc(sizeof(int));
5.  cudaMalloc((void **) &answer_d, sizeof(int));
```

เหตุผลที่จำเป็นต้องมีการคัดลอกข้อมูลเนื่องจากหน่วยประมวลผลกราฟิกไม่สามารถอ้างอิงหน่วยความจำหลักโดยตรงได้ ขึงต้องมีการสำเนาข้อมูลไปยังหน่วยความจำของหน่วยประมวลผลกราฟฟิกก่อน ในส่วนนี้จะแตกต่างจากการประมวลผลบนหน่วยประมวลผลกลางอยู่มากเนื่องจากหน่วยประมวลผลกลางนั้นจะสามารถอ้างอิงหน่วยความจำหลักได้โดยตรงเลยดังนั้นจึงไม่จำเป็นต้องมีการทำขั้นตอนดังกล่าวหากแต่มีแค่กาประกาศค่าตัวแปรต่างๆเท่านั้น (จากขั้นตอนที่ 1 ในรูป 3.5 (b)) เช่น

```
1.  float *a_h, *a_d;
2.  int *answer_h, *answer_d;
3.  const int N = 999999;
```

ขั้นตอนที่ 5 (รูปที่ 3.4 (5)) Service wrapper ทำการสั่งการหน่วยประมวลผลกราฟิก โดยใช้คำสั่งดังนี้ (จากขั้นตอนที่ 2 ในรูป 3.5 (a))

```
1.  // Do calculation on device:
2.  int block_size = 128;
3.  int n_blocks = N/block_size + (N%block_size == 0 ? 0:1);
4.  square_array <<< n_blocks, block_size >>> (a_d, N);
5.  insert_array <<< n_blocks, block_size >>> (a_d, 395741);
6.  search_array <<< n_blocks, block_size >>> (a_d,search,answer_d);
```

จะเห็นได้ว่าการสั่งงานของหน่วยประมวลผลกลางที่ส่งไปยังหน่วยประมวลผลกราฟิกนั้นจะมีรูปแบบที่ต่างจากการสั่งงานของหน่วยประมวลผลกลางอยู่ กล่าวคือมีการประกาศค่าจำนวน block และ ขนาดของ block (ในบรรทัดที่ 3) จะแตกต่างกับการสั่งงานหน่วยประมวลผลกลางซึ่งไม่จำเป็นต้องมีการประกาศในส่วนนี้ (จากขั้นตอนที่ 2 ในรูป 3.5 (b)) เช่น

```
1.  square_array(a_d, N);
```

การประกาศค่าจำนวน block และ ขนาดของ blockในการสั่งการหน่วยประมวลผลทางกราฟิกนั้นมีที่มีจากหลักการที่สำคัญในการพัฒนาโปรแกรมบนหน่วยประมวลผลทางกราฟิก 3 ข้อดังต่อไปนี้

1) ในการประมวลผลบนหน่วยประมวลผลทางกราฟิกผู้ใช้งานสามารถประกาศ Function ที่เรียกว่า Kernels ขึ้นเมื่อเรียกใช้ หน่วยประมวลผลทางกราฟิกจะทำการ Execute N ครั้งในรูปแบบขนาน และจำนวน N Threads ซึ่งจะต่างกับการประมวลผลครั้งเดียวของ Function ของภาษา C ทั่วๆไป Kernel จะถูกประกาศด้วยการใช้ __global__ นำหน้าชื่อ Function และจะถูกเรียกใช้โดยการใช้ <<<...>>> ดังนี้

```
1.  __global__ void VecAdd(float* A, float* B, float* C){}
2.  int main()
3.  {
4.          // Kernel invocation
5.          VecAdd<<<1, N>>>(A, B, C);
6.  }
```

แต่ล่ะ Threads จะมี Threads ID ที่แตกต่างกัน ซึ่งจะสามารถเข้าถึงได้ด้วยตัวแปร threadIdx โดยในโค้ดตัวอย่างได้แสดงให้เห็นถึงการบวกค่า vector A และ B และเก็บไว้ใน Array C ดังต่อไปนี้

```
1.  __global__ void VecAdd(float* A, float* B, float* C){
2.          int i = threadIdx.x;
3.          C[i] = A[i] + B[i];
4.  }
5.  int main(){
6.          // Kernel invocation
7.          VecAdd<<<1, N>>>(A, B, C);
8.  }
```

2) ในการประมวลผลบนหน่วยประมวลผลทางกราฟิกผู้ใช้งานสามารถอ้างอิงลำดับของ Threads ได้โดยผ่านการประกาศตัวแปร threadIdx ซึ่งเป็น 3-component vector นั้นหมายความว่า จะสามารถอ้างอิง Threads ได้ในรูปแบบ 1 มิติ , 2 มิติ และ 3 มิติ ดั่งเช่นในตัวอย่างซึ่งเป็นโปรแกรมในการคำนวน Matrix 2 มิติ

```
1.  // Kernel definition
2.  __global__ void MatAdd(float A[N][N], float B[N][N],float C[N][N]){
3.          int i = threadIdx.x;
4.          int j = threadIdx.y;
5.          C[i][j] = A[i][j] + B[i][j];
6.  }
7.  int main(){
8.  // Kernel invocation
9.          dim3 dimBlock(N, N);
10.         MatAdd<<<1, dimBlock>>>(A, B, C);
11. }
```

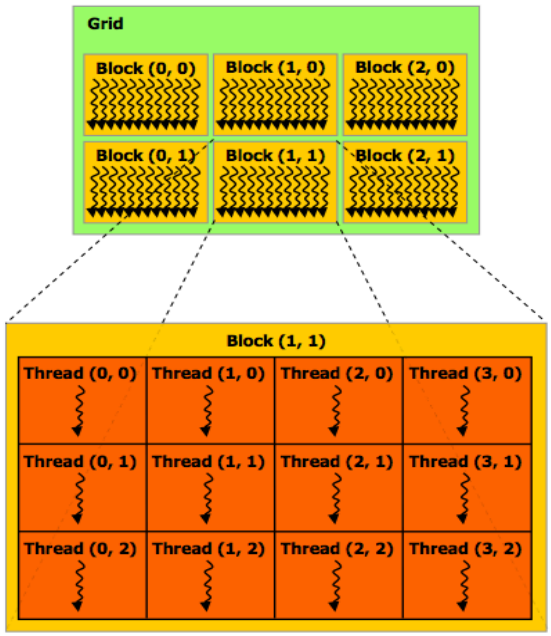
ดัชนีที่ใช้สำหรับอ้างอิง Threads (Thread index) และ Threads ID จะมีความสัมพันธ์แบบตรงไปตรงมา กล่าวคือ สำหรับ block 1 มิตทั้ง 2 ค่าจะมีค่าเท่ากัน สำหรับ block 2 มิติ ที่มีค่าเป็น (Dx, Dy) thread ID ของ ดัชนีที่ใช้สำหรับอ้างอิง Threads (x, y) จะเป็น (x + y Dx) และสำหรับ block ขนาด 3 มิติ (Dx, Dy, Dz) thread ID ของ ดัชนีที่ใช้สำหรับอ้างอิง Threads (x, y, z) จะเป็น (x +yDx +zDx Dy)

Threads ที่อยู่ใน block เดียวกันจะสามารถ sharing ข้อมูลผ่าน shared memory และสามารถประสานข้อมูลสำหรับการประมวลผลได้ โดยใช้คำสั่ง __syncthreads()

Thread ที่อยู่ใน Block เดียวกันนั้นจะถูกทำการประมวลบน Processor Core (ของหน่วยประมวลผลทางกราฟิก) เดียวกันโดยที่จำนวนของ Threads ต่อ Block นั้นจะขึ้นอยู่กับจำนวนทรัพยากรของหน่วยความจำของแต่ล่ะ Processor Core (เพื่อประสิทธิภาพในการทำงานจะใช้หน่วยความจำที่เป็นแบบ low-latency เช่นเดียวกับ L1 Cache บนหน่วยประมวลผลกลาง) ในหน่วยประมวลผลในปัจจุบันจำนวน Thread ต่อ Block อาจจะมีจำนวนถึง 512 threads
การสั่งจำนวน Threads และ Block จะใช้คำสั่งดังตัวอย่าง

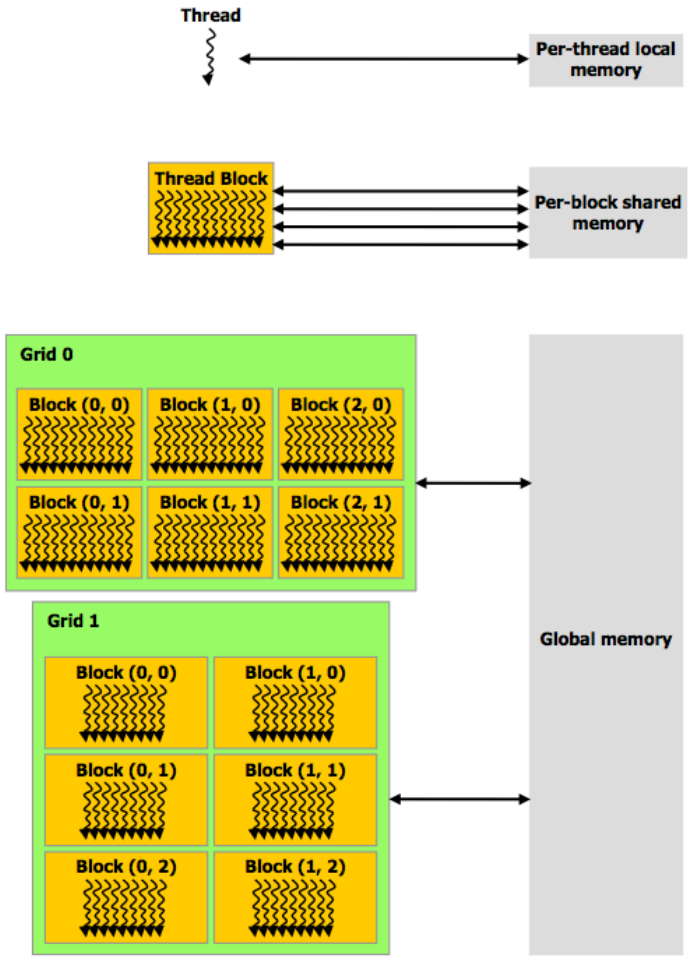
```
1.  // Kernel definition
2.  __global__ void MatAdd(float A[N][N], float B[N][N],
3.  float C[N][N]){}
4.  int main(){
5.          // Kernel invocation
6.          dim3 dimBlock(16, 16);
7.          dim3 dimGrid((N + dimBlock.x – 1) / dimBlock.x,
8.                       (N + dimBlock.y – 1) / dimBlock.y);
9.          MatAdd<<<dimGrid, dimBlock>>>(A, B, C);
10. }
```

จะเห็นได้ว่าในตัวอย่างข้างต้นจำนวน Block ทั้งหมดจะมีค่าคือ 16x16 = 256 ส่วนจำนวนของ Threads นั้นจะถูกคำนวนให้เพียงพ่อต่อการประมวลผลข้างต้น



รูปที่ 3.6 การแบ่งจำนวนงานบนหน่วยประมวลผลกราฟิก [5]

3) ในการประมวลผลบนหน่วยประมวลผลทางกราฟิกผู้ใช้งานสามารถสั่งการให้ Threads ต่างๆ สามารถที่จะเข้าถึงข้อมูลในหน่วยความจำเพื่อใช้สำหรับประมวลผลข้อมูลดังรูปที่ ซึ่งแต่ล่ะ Thread จะมี private local memory ของแต่ล่ะ Threads เอง และแต่ล่ะ Thread block จะมี shared memory ที่ทุก Thread ใน block นั้นๆสามารถเข้าถึงได้ และท้ายที่สุด จะมี global memory ที่ทุก Thread จะสามารถเข้าถึงได้



รูปที่ 3.7 Memory Hierarchy [5]

ขั้นตอนที่ 6 (รูปที่ 3.4 (6)) หน่วยประมวลผลกราฟิกทำการประมวลผลโดยใช้ข้อมูลในหน่วยประมวลผลกราฟิก โดยในขั้นนี้จะทำการคำนวนโดยใช้ Streaming Processor ที่ปกติแล้วหน่วยประมวลผลกราฟิกใช้สำหรับการประมวลผลกราฟฟิกแต่ละ Texture เพื่อที่จะใช้ในการ Render กราฟฟิกนั้นมาประมวลผลในข้อมูลที่ไม่ใช่กราฟฟิกแทน ซึ่งในปัจจุบันนั้นหน่วยประมวลผลกราฟิกนั้นจะมี Streaming Processor จำนวนมาก จึงทำให้การประมวลผลบน GPU นั้นมีความเร็วที่สูงอันเนื่องมาจากมีการแบ่งส่วนการประมวลผลไปร่วมประมวลผลในแบบ Parallels บน Streaming Processor ที่มีจำนวนมากบนหน่วยประมวลผลกราฟิกนั้นเอง ในด้านการพัฒนาโปรแกรมสำหรับประมวลผลค่าบนหน่วยประมวลผลกราฟิก นั้นจะใช้ NVIDIA CUDA C Compiler ซึ่งเป็นคอมไพลเลอร์ที่สามารถประมวลผลคำสั่งภาษา C บนหน่วยประมวลผลกราฟิกได้ ดังตัวอย่างเป็นตัวอย่างการประมวลผลบนหน่วยประมวลผลกลาง (จากขั้นตอนที่ 3 ในรูป 3.5 (a)) ซึ่งในการประมวลผลส่วนนี้บนหน่วยประมวลผลกลางจะใช้คำสั่ง

```
1.   __global__ void insert_array(float *a, int index)
2.   {
3.           a[index]=1;
4.           .....
5.   }
```

ในส่วนของคำสั่งในการประมวลผลบนหน่วยประมวลผลทางกราฟิกนั้นจะอธิบายในหัวข้อ 3.4 และบนหน่วยประมวลผลทางกราฟิกจะใช้คำสั่ง

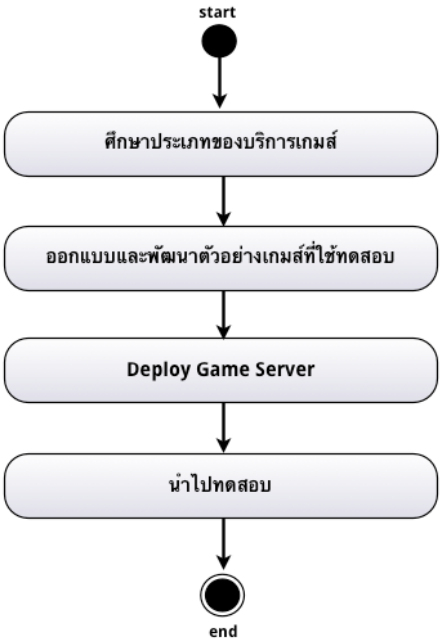
```
1.   void insert_array(float *a, int index)
2.   {
3.           a[index]=1;
4.           .....
5.   }
```

ขั้นตอนที่ 7 (รูปที่ 3.4 (7)) หน่วยประมวลผลกราฟิกจะเก็บผลลัพธ์ที่ได้ไว้ในหน่วยความจำของมัน **ขั้นตอนที่ 8** (รูปที่ 3.4 (8)) หลังจากที่ประมวลผลภาระงานทั้งหมดเสร็จสิ้น Service Wrapper จะทำการสั่งการให้หน่วยประมวลผลกราฟิกทำการคัดลอกข้อมูลในหน่วยความจำของมันกลับมายังหน่วยความจำหลัก (รูปที่ 3.4 (8) และ รูป 3.5 (a) (4))

```
1.   // Retrieve result from device and store it in host array
2.   cudaMemcpy(a_h, a_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
3.   cudaMemcpy(answer_h,answer_d,sizeof(int),
4.                                   cudaMemcpyDeviceToHost);
```

3.4 กรณีศึกษาการพัฒนาโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายเพื่อการทดสอบ

การออกแบบและพัฒนาโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายเพื่อใช้ทดสอบแนวคิดในงานวิจัยนี้จะกำหนดขั้นตอนดำเนินการ ดังต่อไปนี้



รูปที่ 3.8 ลำดับงานในการดำเนินงานเพื่อกรณีศึกษาโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่าย

3.4.1 การแบ่งชนิดภาระงานโปรแกรมแม่ข่ายเกมส์ออนไลน์

ในการพัฒนาเกมส์ออนไลน์นั้นสามารถแบ่งการพัฒนาเกมส์ได้เป็นส่วนใหญ่ๆ 2 ส่วนคือ ส่วนของโปรแกรมแม่ข่าย และ ส่วนของโปรแกรมลูกข่าย ซึ่งในวิทยาพิพนธ์นี้นใจที่จะปรับปรุงประสิทธิภาพของโปรแกรมในส่วนแม่ข่ายเกมส์ออนไลน์ ซึ่งเป็นส่วนที่จะให้บริการแก่โปรแกรมลูกข่ายในหลายๆด้าน เช่น บริการฐานข้อมูลของเกมส์ บริการคำนวนคะแนนแก่ผู้เล่นทุกคน เป็นต้น

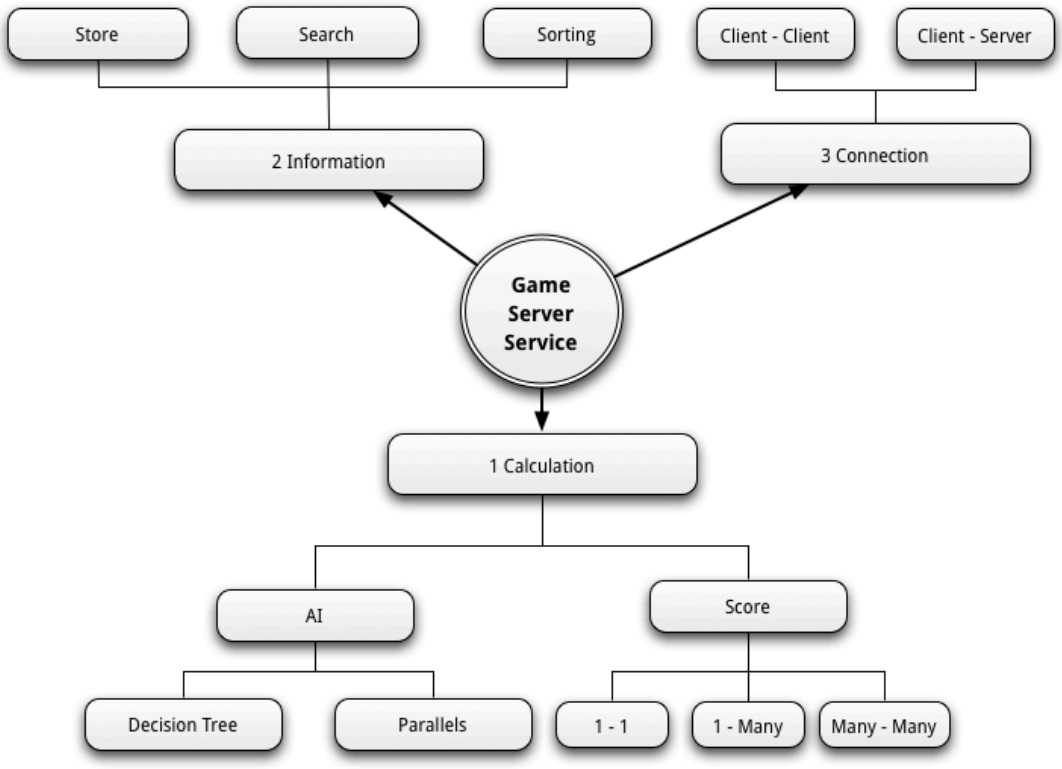
ในส่วนโปรแกรมแม่ข่ายเกมส์ออนไลน์นั้นส่วนใหญ่ๆจะสามารถแบ่งภาระงานได้เป็น 3 ชนิดคือ

1. ภาระในการคำนวนผล (Logic Server) จะเป็นส่วนที่รวมเอาการคำนวนด้านต่างๆของเกมส์เอาไว้ ในส่วนนี้อาจจะแบ่งได้เป็นหลายๆรูปแบบ ตัวอย่างเช่น ในเกมส์ Ragnarok Online นั้นในส่วนของโปรแกรมแม่ข่ายที่มีชื่อว่า Aegis นั้นจะแบ่งส่วนของ Logic Server เป็น 4 ส่วนหลักๆกล่าวคือ Inter Server ในส่วนนี้ใช้รองรับ Guild Siege, Guild Chat และ Private Message , Zone Server ใช้ในการรองรับ Game Play Dynamics, Character Server ใช้ในการรองรับ ข้อมูลต่างๆ

ของตัวละครในเกมส์ และ Account Server ใช้ในการตรวจเช็คข้อมูลในฐานข้อมูลเกี่ยวกับด้าน accountเช่นในส่วนการตรวจสอบการชำระเงินในการเล่น. [7]

2. ภาระในด้านการประมวลผลข้อมูล (Database Server) เป็นส่วนที่จะใช้ในการเก็บข้อมูลต่างๆของเกมส์ ในส่วนนี้อาจจะใช้ DBMS (Database Management System) ที่มีอยู่ก่อนหน้านี้ในการจัดเก็บข้อมูลก็เป็นได้ เช่น ตัวอย่างเช่น ในเกมส์ Ragnarok Online นั้นในส่วนของโปรแกรมแม่ข่ายที่มีชื่อว่า Aegis นั้นจะใช้ Microsoft SQL Server เป็น DBMS [7]
3. ภาระในด้านการเชื่อมต่อ (Connection Server) เป็นส่วนที่จะใช้ในการเชื่อต่อ ส่งข้อมูลกันระหว่างโปรแกรมลูกข่าย และ โปรแกรมแม่ข่าย

จาก 3 ส่วนหลักๆที่กล่าวมานั้น เราสามารถจำแนกชนิดของภาระงานโปรแกรมแม่ข่ายเกมส์ออนไลน์โดยอาศัยหลักเกณฑ์ชนิดอัลกอริทึ่มที่ใช้ในการพัฒนาโปรแกรม ได้ดังรูปต่อไปนี้



รูปที่ 3.9 การจำแนกชนิดของภาระงานของโปรแกรมแม่ข่ายเกมส์ออนไลน์

จากรูปที่ 3.9 เราสามารถจำแนกชนิดของภาระงานโปรแกรมแม่ข่ายเกมส์ออนไลน์ได้ ดังต่อไปนี้

3.4.1.1 ภาระงานด้านที่เกี่ยวกับการคำนวน (Calculation-Based Service)

สามารถแบ่งได้เป็น 2 ลักษณะคือการคำนวน AI และการคำนวนคะแนน

1.1 การคำนวน AI มี 2 รูปแบบคือ

1.1.1 การคำนวน AI ในรูปแบบ Decision Tree คือการประมวลผลแบบประมวลตามเงื่อนไขไปเรื่อยๆเช่น เมื่อผ่านเงื่อนไข A จึงทำงานเงื่อนไข B ต่อ เช่น

```
1.  if(a=1){
2.     a=win
3.  }
4.  else{
5.     if(b=2){
6.        b=win
7.     }
8.  }
```

1.1.2 การคำนวน AI ในรูปแบบ Parallels คือการประมวลผลแบบพร้อมๆกัน เช่น

```
1.  #pragma omp parallel for
2.     for (i = 0; i < N; i++){
3.        a[i] = 2 * i;
4.      }
5.  }
```

1.2 การคำนวนคะแนน มี 3 รูปแบบคือ

1.2.1 รูปแบบการคำนวนคะแนนแบบ 1-1 คือการประมวลผลคะแนนของผู้เล่นแต่ล่ะคนไป

```
1.  if(id=2){
2.     point++
3.  }
```

1.2.1 รูปแบบการคำนวนคะแนนแบบ 1-Many คือการประมวลผลคะแนนของผู้เล่นค่าเดียวพร้อมๆกันหลายๆคน

1.2.2 รูปแบบการคำนวนคะแนนแบบ Many-Many คือการประมวลผลคะแนนของผู้เล่นหลายๆค่า พร้อมๆกันหลายๆคน

```
1.  while(i<100){
2.    point1++
3.    point2++
4.  }
```

3.4.1.2 ภาระงานด้านที่เกี่ยวกับข้อมูล (Information-Based Service)

2.1 การเก็บข้อมูลลงหน่วยความจำหลัก หรือ ลงอุปกรณ์เก็บข้อมูล (Store) คือ ภาระงานที่โปรแกรมแม่ข่ายจะต้องทำการจัดเก็บข้อมูลลงไปยังอุปกรณ์เก็บข้อมูล

2.2 การค้นหาข้อมูล (Search) ภาระงานที่โปรแกรมแม่ข่ายจะต้องทำการค้นหาข้อมูลจากหน่วยความจำหลัก หรือ อุปกรณ์เก็บข้อมูล

2.3 การจัดเรียงข้อมูล (Sorting) ภาระงานที่โปรแกรมแม่ข่ายจะต้องทำการจัดเรียงข้อมูลจากหน่วยความจำหลัก หรือ อุปกรณ์เก็บข้อมูล

3.4.1.3 ภาระงานด้านที่เกี่ยวกับการเชื่อมต่อ (Connection-Based Service)

มีด้วยกัน 2 แบบคือ ภาระงานในการจัดการการเชื่อมต่อระหว่างเครื่องลูกข่ายด้วยกัน และ ภาระงานในการจัดการการเชื่อมต่อระหว่าเครื่องลูกข่ายและแม่ข่าย

3.4.2 การออกแบบและพัฒนาตัวอย่างบริการเกมส์ที่ใช้ในการทดสอบ

ในหัวข้อที่ 3.4.2 นี้จะทำการสร้างตัวอย่างบริการเกมส์เพื่อสำหรับการทดสอบโดยบริการที่นำมาใช้ในการทดสอบนั้นจะสร้างโดยอิงจากผลการศึกษาชนิดของภาระงานเกมส์ออนไลน์ในหัวข้อ 3.4.2.1

3.4.2.1 เกมส์ตัวอย่างที่ใช้ในการทดสอบ

วิทยานิพนธ์นี้ได้พัฒนาเกมส์ที่ใช้ในการทดสอบประสิทธิภาพของแนวคิดที่ได้นำเสนอไว้ โดยมีการทดสอบกับเกมส์ 2 ประเภทคือ เกมส์ออนไลน์ที่มีผู้เล่นเพียง 1 คน และ เกมส์ออนไลน์ที่มีผู้เล่นพร้อมๆกันหลายๆคน ในการพัฒนาโปรแกรมตัวอย่างที่จะใช้ในการทดสอบนั้นจะมีการพัฒนา

บริการที่เกมส์ประเภทนั้นๆใช้งาน ทั้งหน่วยประมวลผลกลางและหน่วยประมวลผลทางกราฟิก แล้วนำมาทำงานอยู่บนสถาปัตยกรรมที่ได้ออกแบบ

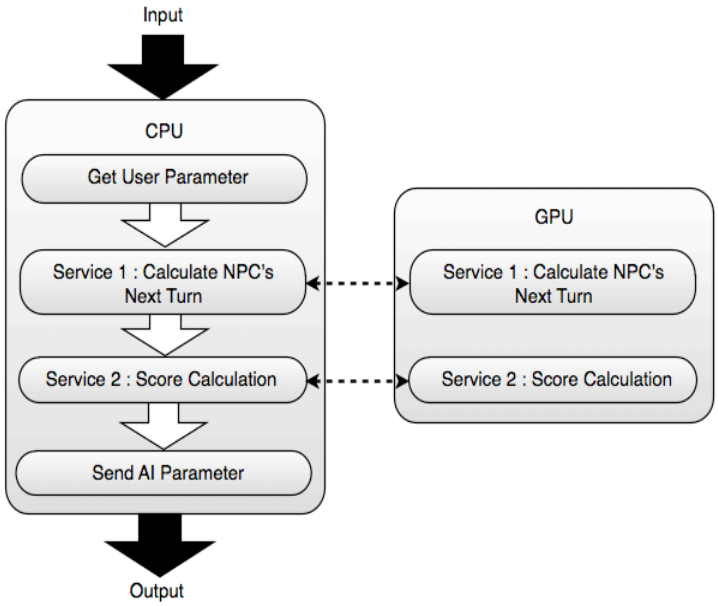
โดยเกมส์ที่จะใช้ในการทดสอบนั้นคือ เกมส์ Tic-Tac-Toe ในการทดสอบเกมส์ออนไลน์ที่มีผู้เล่นเพียง 1 คน และ เกมส์จำลองสถานะการซื้อขายหุ้นในตลาดหลักทรัพย์ เพื่อใช้ในการทดสอบเกมส์ออนไลน์ที่มีผู้เล่นพร้อมๆกันหลายๆคน เนื่องจากเกมส์ออนไลน์ 2 เกมส์นี้มีบริการที่ตรงกับกลุ่มบริการที่ได้แบ่งไว้โดยใช้เกณฑ์จากหัวข้อ 3.4.1 จำนวนมากดั่งที่จะได้อธิบายดังต่อไปนี้

3.4.2.2 เกมส์ Tic Tac Toe

เกมส์แรกที่ใช้ในการทดสอบคือเกมส์ Tic-Tac-Toe แบบที่เล่นผ่านเครือข่ายอินเตอร์เน็ทซึ่งในส่วนของ โปรแกรมเครื่องลูกข่ายนั้นจะเป็นส่วนที่ใช้ในการติดต่อกับตัวผู้เล่น และจะทำการตรวจสอบว่าเกมส์จบหรือยังถ้ายังจะทำการส่งค่า parameter ของตัวผู้เล่นไปยังเครื่องแม่ข่าย เมื่อเครื่องแม่ข่ายรับค่าดังกล่าวแล้วจะทำการคำนวนคะแนนของผู้แล่นและตำแหน่งที่ NPC จะเดินในรอบต่อไป จากนั้นจึงส่งค่าดังกล่าวกลับไปยังโปรแกรมเครื่องลูกข่ายเพื่อแสดงผลลัพธ์วนไปเรื่อยๆจนกระทั่งจบเกมส์และส่วนของโปรแกรมเครื่องแม่ข่ายเกมส์ Tic-Tac-Toe ออนไลน์ ซึ่งจะแบ่งออกเป็น 4 บริการดังต่อไปนี้

1. บริการรับค่า Input จากเครื่องลูกข่าย
2. บริการส่ง Output คืนเครื่องลูกข่าย
3. บริการคำนวนคะแนนที่มีหน้าที่ในการคำนวนคะแนนจาก Parameter ที่ผู้เล่นส่งมาให้โดยทำการ + คะแนนใหม่เข้าไปในคะแนนเก่า บริการนี้จะแสดงให้เห็นถึง บริการนี้อยู่ในเกณฑ์หัวข้อ ที่ 3.4.1 : รูปแบบการคำนวนคะแนนแบบ 1-1
4. บริการคำนวนตำแหน่งที่ NPC จะเดินในรอบถัดไปที่จะทำการค้นหาตำแหน่งต่อไปที่ NPC จะเดินโดยการ Query ออกมาจากตารางขนาด 362,880 ค่า (มาจากวิธีการเล่นเกมส์ Tic Tac Toe ทั้งหมด 362,880 รูปแบบ เลือกค่าที่ดีที่สุด (ใกล้เคียงเกมส์ในปัจจุบันและผลคือ NPC เป็นฝ่ายชนะ หรือ อย่างแย่ที่สุดคือเสมอ) ออกมาเพื่อเป็นตำแหน่งการเดินตำแหน่งต่อไปของ NPC บริการนี้อยู่ในเกณฑ์จากหัวข้อที่ 3.3.1 : การคำนวน AI ในรูปแบบ Parallels

โครงสร้างของโปรแกรมแม่ข่ายเกมส์ออนไลน์ตัวอย่างนี้จะแสดงในรูปที่ 3.9



รูปที่ 3.10 แผนภาพการทำงานของโปรแกรมแม่ข่ายเกมส์ Tic Tac Toe ออนไลน์

จากโปรแกรมแม่ข่ายเกมส์ออนไลน์ตัวอย่าง (เกมส์ Tic Tac Toe) นี้นั้นจะมีอยู่เพียงแค่ 2 บริการเท่านั้นที่สามารถแบ่งภาระงานไปประมวลผลบนหน่วยประมวลผลทางกราฟิกได้ คือ บริการคำนวนตำแหน่งที่ NPC จะเดินในรอบถัดไป และ บริการคำนวนคะแนน ส่วนอีก 2 บริการไม่สามารถแบ่งภาระงานไปประมวลผลบนหน่วยประมวลผลทางกราฟิกได้เนื่องจากการพัฒนาโปรแกรมบนหน่วยประมวลผลกราฟิกในปัจจุบันนั้นยังไม่สามารถทำได้ บริการทั้ง 4 ของโปรแกรมแม่ข่ายเกมส์ Tic-Tac-Toe นั้นสามารถอธิบายได้ดังนี้

1 บริการรับค่า Input จากเครื่องลูกข่าย

บริการรับค่า Input จากเครื่องลูกข่ายนั้นใช้ภาษา PHP ในการเรียกใช้งาน โดยที่จะใช้คำสั่ง GET ในการรับค่าที่ส่งมาจาก โปรแกรมส่วนลูกข่าย จากนั้นจึงมีหน้าที่ในการเรียกบริการที่ 2 ขึ้นมาทำงาน ในบริการส่วนที่ 1 ไม่สามารถประยุกต์ใช้งานบนหน่วยประมวลผลกราฟิกได้จึงมีแค่ในส่วนการประมวลผลบนหน่วยประมวลผลกลางเท่านั้นใน บริการรับค่า Input จากเครื่องลูกข่าย จะมีการทำงานอยู่ 2 ขั้นตอนคือ

1) รับค่าจากเครื่องลูกข่าย

ในการรับค่าจะเครื่องลูกข่ายนั้นจะคำสั่งดังต่อไปนี้

```
1.  $_GET['userpoint'];
2.  $_GET['gamearry'];
```

โดยที่ userpoint คือคะแนนล่าสุดของผู้เล่น

และ gamearry คือสถานะของเกมส์ล่าสุดจะเก็บค่าตำแหน่งที่ เดินทั้งหมดโดยที่แต่ล่ะตำแหน่งบนตารางของเกมส์ TIC TAC TOE จะมีหมายเลขกำกับดังรูปที่ 3.10

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

รูปที่ 3.11 หมายเลขตำแหน่งบนตารางของเกมส์ Tic Tac Toe

โดยค่า gamearry ที่ถูกส่งมาจะอยู่ในรูปแบบ ตำแหน่งการเดินรอบที่ 1-9 เรียงกัน เช่น

```
1.  $gamearry= 32518;
```

คือ การเดินในรอบแรกจะเดินตำแหน่ง 3

การเดินในรอบที่ 2 จะเดินตำแหน่ง 2

การเดินในรอบที่ 3 จะเดินตำแหน่ง 5

การเดินในรอบที่ 4 จะเดินตำแหน่ง 1

การเดินในรอบที่ 5 จะเดินตำแหน่ง 8

2) ทำการเรียกใช้งานบริการต่อไป

หลังจากที่ได้รับค่าทั้ง 2 แล้วโปรแกรม 1 ก็จะทำการเรียกใช้งานบริการที่ 2 โดยคำสั่งที่ใช้ในการ Execute บริการที่ 2 มีดังต่อไปนี้

```
1.  $ret=exec($cmd);
```

โดยที่ $path คือคำสั่งและ parameter (สถานะล่าสุดของเกมส์จากค่า gamearry ในขั้นตอนที่ 1) ของบริการที่ 2 และ $output คือผลลัพทธ์จากบริการที่ 2

2 บริการคำนวนตำแหน่งที่ NPC จะเดินในรอบถัดไป

บริการที่ 2 จะทำการคำนวนตำแหน่งที่ NPC จะเดินในรอบถัดไป โดยจะมีการทำงาน 2 ขั้นตอนคือ

1) อ่านกรณีที่จะสามารถเล่นเกมส์ Tic Tac Toe ใส่ตัวแปรแบบ Array

ในขั้นตอนการทำงานแรกนี้โปรแกรมจะทำการอ่านกรณีที่จะสามารถเล่นเกมส์ TIC TAC TOE และผลจากการเล่นในรูปแบบนั้นๆทั้งหมดจากฐานข้อมูลชนิด Text File แล้วจึงเก็บไว้ในรูปแบบตัวแปร Array โดยที่ Array Index จะมีค่าเป็นรูปแบบเกมส์ล่าสุด และ ค่าของ Array ตำแหน่งนั้นๆคือค่าผลจากการเล่นเกมส์ตามลำนั้น (1 คือผู้เล่นที่เริ่มต้นก่อนชน่ะ , 2 คือผู้เล่นที่เริ่มต้นหลังชน่ะ และ 3 คือเสมอกัน) เช่น

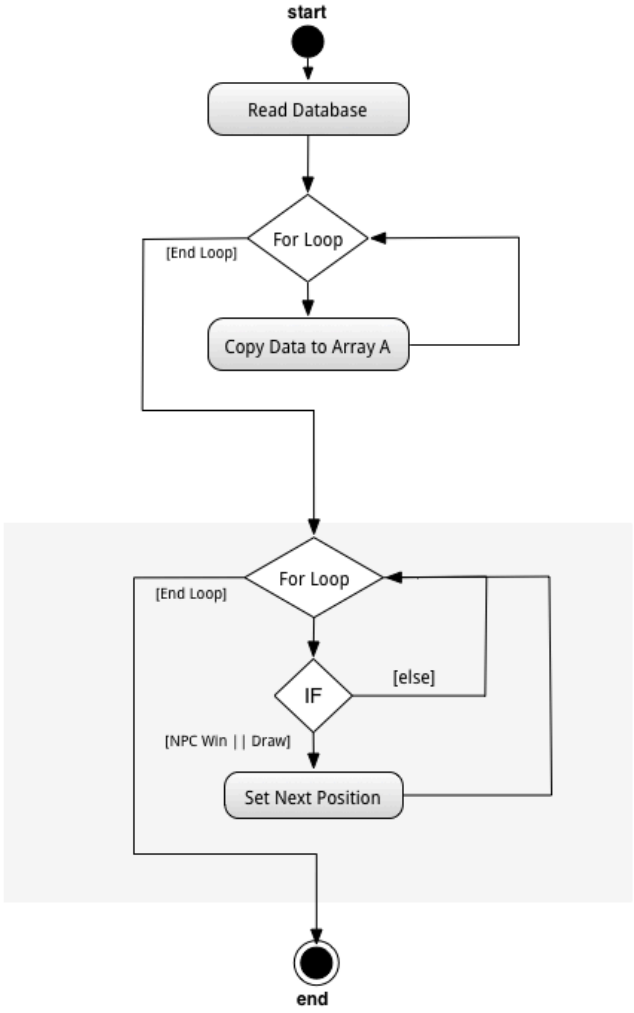
```
1.  a[15328] = 2;
```

มีความหมายว่า
การเล่นที่มีรูปแบบการเดินรอบแรกตำแหน่งที่ 1
การเดินในรอบที่ 2 เดินตำแหน่งที่ 5
การเดินในรอบที่ 3 เดินตำแหน่งที่ 3
การเดินในรอบที่ 4 เดินตำแหน่งที่ 2
การเดินในรอบที่ 5 เดินตำแหน่งที่ 8
จะมีผลคือ ผู้เล่นที่เริ่มต้นหลังชน่ะ

ในโปรแกรมขั้นตอนที่ 1 นี้ทั้งการพัฒนาโปรแกรมบนหน่วยประมวลผลทางกราฟิกและหน่วยประมวลผลกลางจะมีรูปแบบในการทำงานที่เหมือนกันคือทำการวน for loop เพื่อที่จะเก็บค่าผลของแต่ล่ะค่าลงไปยัง Array A

2) นำค่าที่ได้จากบริการที่ 1 มาค้นหาตำแหน่งเดินของ NPC ในรอบถัดไป

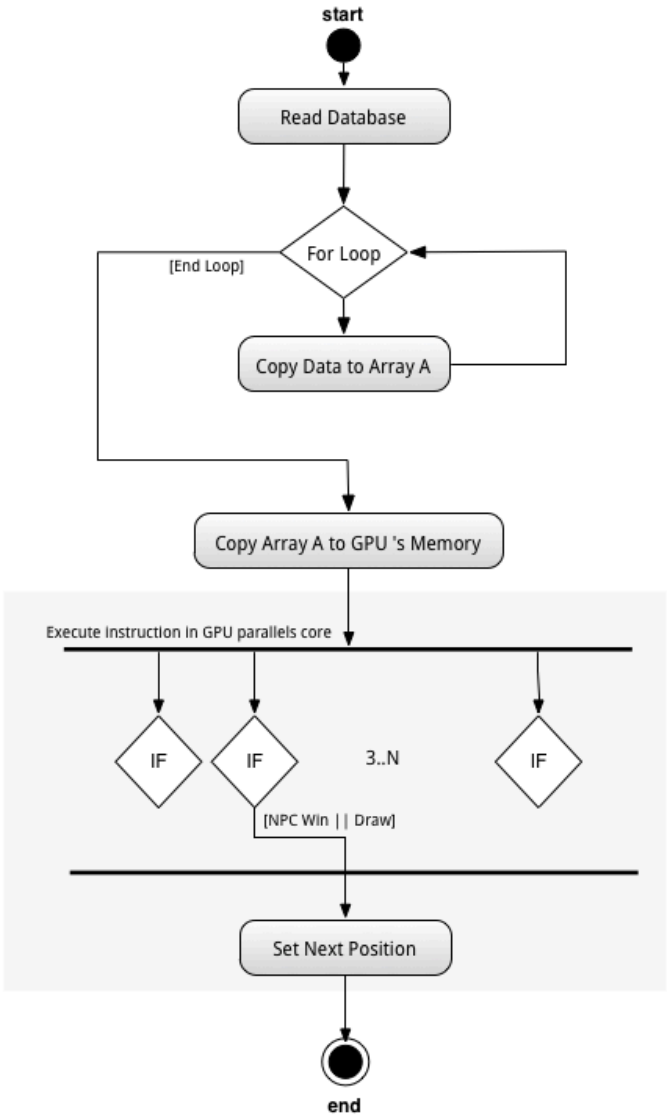
ในขั้นตอนที่ 2 นี้โปรแกรมจะทำการค้นหาตำแหน่งเดินของ NPC ในรอบถัดไป โดยที่ตำแหน่งการเดินของ NPC นั้นจะต้องมีค่าผลลัพท์คือ NPC ชน่ะหรือเสมอเท่านั้น ในการทำงานขั้นตอนที่ 2 นี้โปรแกรมที่ประมวลผลบนหน่วยประมวลผลกลางจะทำงานในรูปแบบที่แตกต่างกันกับโปรแกรมที่ประมวลผลบนหน่วยประมวลผลทางกราฟิก
โดยที่โปรแกรมบนหน่วยประมวลผลกลางจะทำงานดังต่อไปนี้



รูปที่ 3.12 แผนภาพการทำงานของบริการที่ 2 บนหน่วยประมวลผลกลาง

จะเห็นได้ว่าการทำงานขั้นตอนที่ 2 นี้ (ในพื้นที่สีเทา) โปรแกรมแม่ข่ายที่ประมวลผลบนหน่วยประมวลผลกลางจะใช้วิธีการวน for loop ทั้งหมดเพื่อที่จะค้นหาค่าใน Array A

โปรแกรมบนหน่วยประมวลผลกราฟิกจะทำงานดังต่อไปนี้



รูปที่ 3.13 แผนภาพการทำงานของบริการที่ 2 บนหน่วยประมวลผลกราฟิก

จะเห็นได้ว่าการทำงานขั้นตอนที่ 2 นี้ (ในพื้นที่สีเทา) โปรแกรมแม่ข่ายที่ประมวลผลบนหน่วยประมวลผลกลางจะใช้วิธีการสั่งให้มีการรัน Kernel เพื่อรันคำสั่ง IF ทั้งหมดทุกตัวโดยรันในแบบขนานพร้อมๆกันทีล่ะ 248 คำสั่งหลังจากนั้นจึงเก็บค่าที่ได้จากการค้นหาไปใช้ยังบริการถัดไป

3 บริการคำนวนคะแนน

บริการคำนวนคะแนนที่มีหน้าที่ในการคำนวนคะแนนจาก Parameter ที่ผู้เล่นส่งมาให้โดยทำการ + คะแนนใหม่เข้าไปในคะแนนเก่า โดยหลังจากรับค่าจากบริการที่ 2 แล้วโปรแกรมจะทำการ

นำคะแนนเดิมของผู้เล่นมาบวกค่าคะแนนใหม่เพิ่มขึ้น (ในโปรแกรมตัวอย่างนี้จะบวกเพิ่ม 100 คะแนนต่อรอบ) ในบริการที่ 3 นี้ทั้งบน หน่วยประมวลผลกลาง และ หน่วยประมวลผลกราฟิก จะทำงานเหมือนกันโดยใช้คำสั่งดังต่อไปนี้

```
1.  *answer=a+100;
```

4 บริการส่ง Output คืนเครื่องลูกข่าย

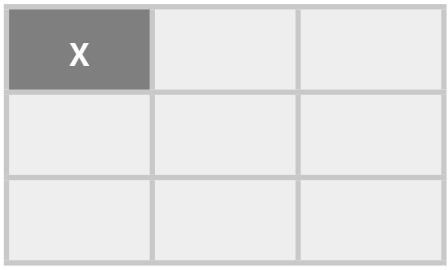
บริการที่ 4 จะทำการคืนค่า Output ที่ได้รับจาก บริการที่ 2 และ 3 ไปยังเครื่องลูกข่ายนั้นโดยใช้ภาษา PHP ในการเรียกใช้งานโดยใช้คำสั่งดังต่อไปนี้

```
1.  $answer = '<walk>$pnt</walk>
2.                  <score>$scr</score>'
3.  return $answer;
```

โดยที่ $pnt คือค่าตำแหน่งการเดินที่ถูกส่งมาจากบริการที่ 2 จะอยู่ในรูปแบบตัวเลขเดี่ยว บอกตำแหน่งที่ NPC จะเดินในรอบต่อไป เช่น

```
1.   $pnt = 1;
```

คือตำแหน่งที่ NPC จะเดินในรอบต่อไปคือตำแหน่ง



รูปที่ 3.14 ตำแหน่งที่ NPC จะเดินเมื่อ $pnt=1

และ $scr เป็นคะแนนของผู้เล่นที่ถูกส่งมาจากบริการที่ 3

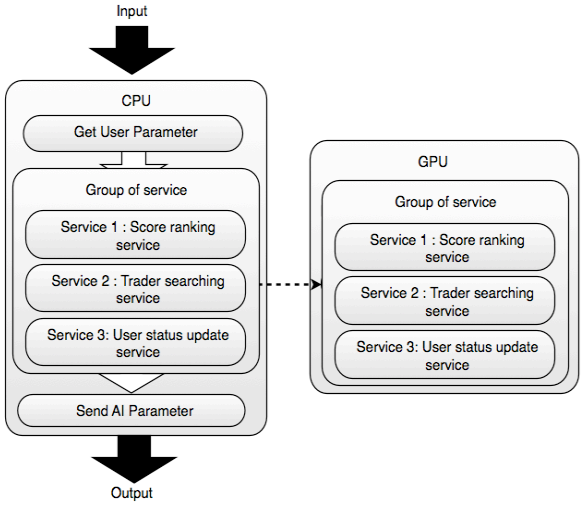
ในบริการส่วนที่ 4 นี้ไม่สามารถประยุกต์ใช้งานบนหน่วยประมวลผลทางกราฟิกได้จึงมีแค่ในส่วนการประมวลผลบนหน่วยประมวลผลกลางเท่านั้น

3.4.2.3 Virtual online stock exchange games

โปรแกรมแม่ข่ายเกมส์ออนไลน์ตัวอย่างเกมส์ที่สองที่ได้นำมาใช้ในการทดสอบนี้คือ เกมส์จำลองสถานะการซื้อขายหุ้นในตลาดหลักทรัพย์ (Virtual online stock exchange games) โดยโปรแกรมนี้จะประกอบไปด้วย 5 บริการดังต่อไปนี้ :

1. บริการรับค่า Input จากเครื่องลูกข่าย
2. บริการในการจัดเรียงคะแนนของผู้เล่นทั้งหมดในเกมส์เพื่อที่จะจัดอันดับคะแนน บริการนี้อยู่ในเกณฑ์จากหัวข้อ 3.4.1 : การค้นหาข้อมูล
3. บริการในการค้นหารายชื่อหุ้นที่มีขายอยู่ในเกมส์ บริการนี้อยู่ในเกณฑ์จากหัวข้อ 3.4.1 : การการจัดเรียงข้อมูล
4. บริการเพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเกมส์ บริการนี้อยู่ในเกณฑ์จากหัวข้อ 3.4.1 : รูปแบบการคำนวนคะแนนแบบ 1-Many และ รูปแบบการคำนวนคะแนนแบบ Many-Many
5. บริการส่ง Output คืนเครื่องลูกข่าย

โครงสร้างของโปรแกรมแม่ข่ายเกมส์ออนไลน์ตัวอย่างนี้จะแสดงในรูปที่ 3.15



รูปที่ 3.15 แผนภาพการทำงานของโปรแกรมแม่ข่ายเกมสจำลองสถานะการซื้อขายหุ้นในตลาดหลักทรัพย์

โปรแกรมจะเริ่มต้นรับข้อมูลจากเครื่องลูกข่ายผ่านทาง “บริการรับค่า Input จากเครื่องลูกข่าย” จากนั้นจึงเลือกใช้บริการใดบริการหนึ่งจาก 3 บริการนี้คือ “บริการในการจัดเรียงคะแนนของผู้เล่นทั้งหมดในเกมส์เพื่อที่จะจัดอันดับคะแนน” , “บริการในการค้นหารายชื่อหุ้นที่มีขายอยู่ในเกมส์” และ “บริการเพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเกมส์” จากนั้นจึงทำการส่งค่ากลับไปยังเครื่องลูกข่ายผ่าน “บริการส่ง Output คืนเครื่องลูกข่าย” แต่ละบริการสามารถอธิบายอย่างละเอียดได้ดังนี้

1 บริการรับค่า Input จากเครื่องลูกข่าย

บริการรับค่า Input จากเครื่องลูกข่ายนั้นใช้ภาษา PHP ในการเรียกใช้งาน โดยที่จะใช้คำสั่ง GET ในการรับค่าที่ส่งมาจาก โปรแกรมส่วนลูกข่าย จากนั้นจึงมีหน้าที่ในการเรียกบริการที่ 2 ขึ้นมาทำงาน ในบริการส่วนที่ 1 ไม่สามารถประยุกต์ใช้งานบนหน่วยประมวลผลกราฟิกได้จึงมีแค่ในส่วนการประมวลผลบนหน่วยประมวลผลกลางเท่านั้นใน บริการรับค่า Input จากเครื่องลูกข่าย จะมีการทำงานอยู่ 2 ขั้นตอนคือ

1) รับค่าจากเครื่องลูกข่าย
ในการรับค่าจะเครื่องลูกข่ายนั้นจะคำสั่งดังต่อไปนี้

```
1.  $_GET['param'];
2.  $_GET['service'];
```

โดยที่ param คือ ค่าตัวแปรที่ผู้เล่นส่งมา และ service คือ บริการที่เลือกใช้โดยที่
1 คือ บริการในการจัดเรียงคะแนนของผู้เล่น
2 คือ บริการในการค้นหารายชื่อหุ้นที่มีขายอยู่ในเกมส์
3 คือ บริการเพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเกมส์
เช่น

```
1.  $service=3;
```

คือให้ทำการเลือกใช้บริการเพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเกมส์

2) ทำการเรียกใช้งานบริการต่อไป

หลังจากที่ได้รับค่าทั้ง 2 แล้วโปรแกรมก็จะทำการเรียกใช้งานบริการที่ผู้ใช้เลือกโดยคำสั่งที่ใช้ในการ Execute บริการมีดังต่อไปนี้

```
1.  $ret=exec($cmd);
```

โดยที่ $cmd คือคำสั่งที่ใช้ในการสั่งงานบริการถัดไป เช่น

```
1.  $cmd = ranking_serv;
```

เป็นการส่งให้บริการจัดเรียงคะแนนทำงานเป็นต้น

2 บริการในการจัดเรียงคะแนนของผู้เล่นทั้งหมดในเกมส์เพื่อที่จะจัดอันดับคะแนน

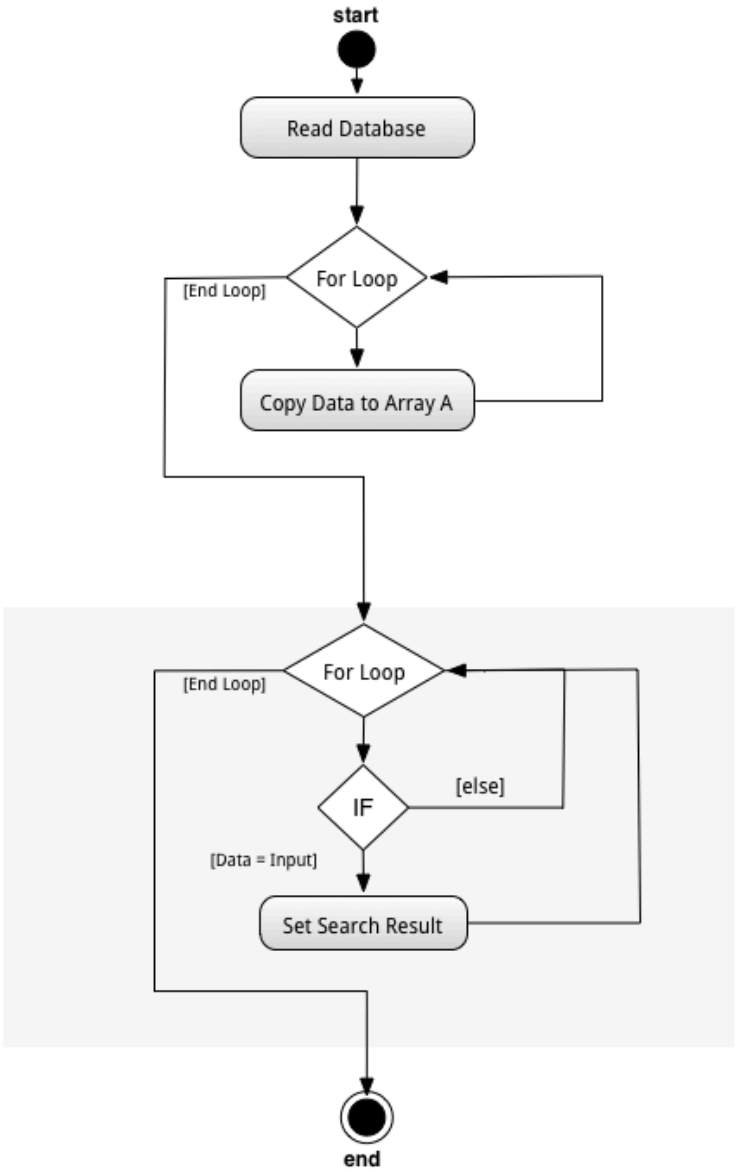
บริการที่ 2 จะทำการจัดเรียงคะแนนของผู้เล่นทั้งหมดในเกมส์ เรียงจากคะแนนมากที่สุดไปน้อยที่สุด ในโปรแกรมทดสอบนั้นได้มีการใช้โปรแกรมตัวอย่างของ NVIDIA CUDA มาใช้ในการทดสอบประสิทธิภาพ [GPU-ABiSort: Optimal Parallel Sorting on Stream Architectures] ฉะนั้นบริการนี้จึงมีเพียงแค่คำสั่งเรียกใช้โปรแกรมจัดเรียงคะแนนดังกล่าวเท่านั้น

```
1.  $ret=exec(“sorting.exe”);
```

3 บริการในการค้นหารายชื่อหุ้นที่มีขายอยู่ในเกมส์

บริการที่ 3 นี้โปรแกรมจะทำการค้นหารายชื่อหุ้นที่มีขายอยู่ในเกมส์ โดยรายชื่อหุ้นที่ตอบกลับมานั้นจะมีค่า Index เท่ากันกับ Input ในการทำงานบริการที่ 3 นี้โปรแกรมที่ประมวลผลบนหน่วยประมวลผลกลางจะทำงานในรูปแบบที่แตกต่างกันกับโปรแกรมที่ประมวลผลบนหน่วยประมวลผลทางกราฟิก

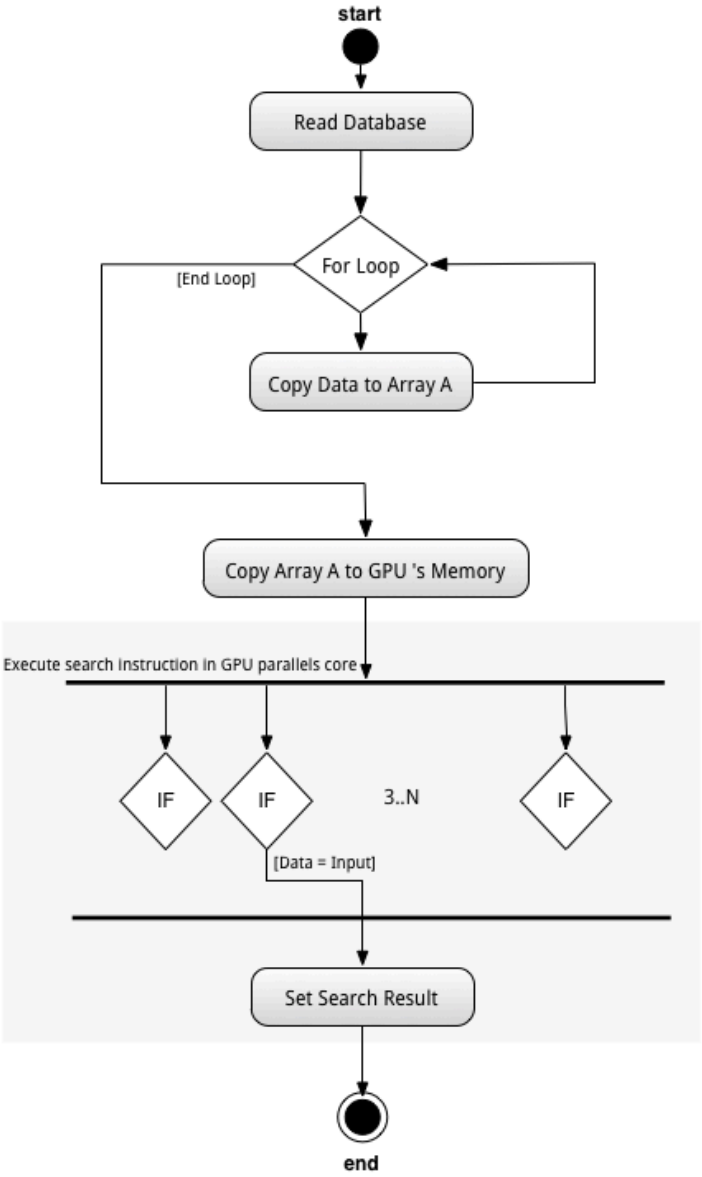
โดยที่โปรแกรมบนหน่วยประมวลผลกลางจะทำงานดังต่อไปนี้



รูปที่ 3.16 แผนภาพการทำงานของบริการค้นหารายชื่อหุ้นบนหน่วยประมวลผลกลาง

รูปที่ 3.16 นี้ได้แสดงให้เห็นว่าการทำงานขั้นตอนที่ 2 นี้ (ในพื้นที่สีเทา) โปรแกรมแม่ข่ายที่ประมวลผลบนหน่วยประมวลผลกลางจะใช้วิธีการวน for loop ทั้งหมดเพื่อที่จะค้นหาค่าใน Array A

โปรแกรมบนหน่วยประมวลผลกราฟิกจะทำงานดังต่อไปนี้



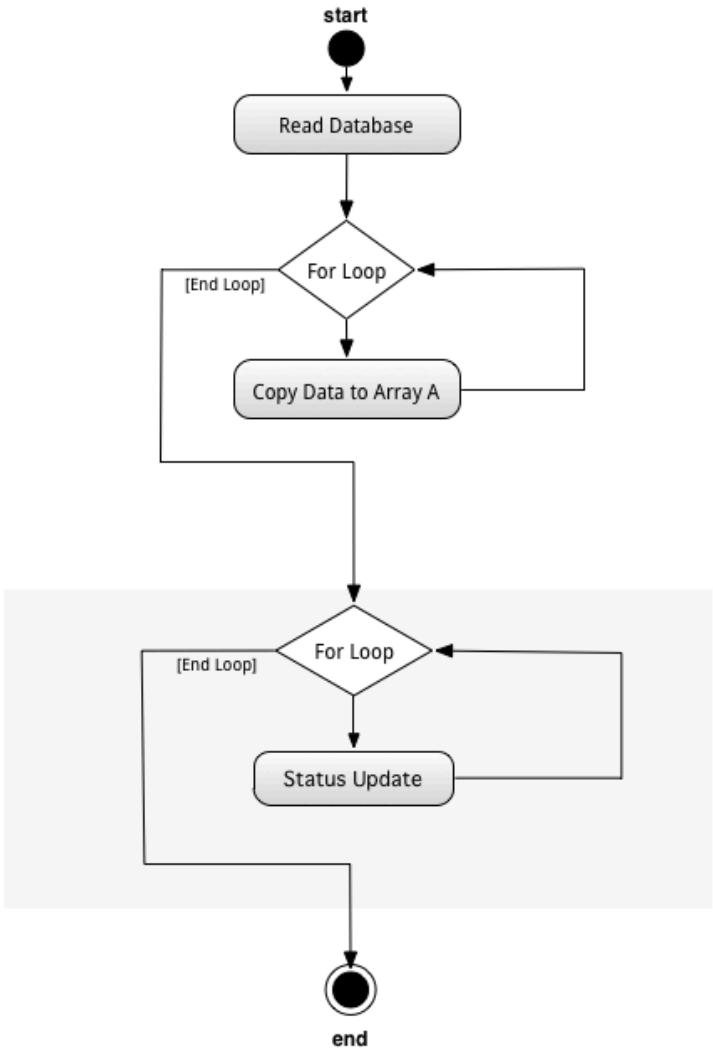
รูปที่ 3.17 แผนภาพการทำงานของบริการค้นหารายชื่อหุ้นบนหน่วยประมวลผลกราฟิก

รูปที่ 3.17 นี้ได้แสดงให้เห็นว่าการทำงานขั้นตอนที่ 2 นี้ (ในพื้นที่สีเทา) โปรแกรมแม่ข่ายที่ประมวลผลบนหน่วยประมวลผลกลางจะใช้วิธีการสั่งให้มีการรัน Kernel เพื่อรันคำสั่ง IF ทั้งหมดทุกตัวโดยรันในแบบขนานพร้อมๆกันทีล่ะ 248 คำสั่ง

4 บริการเพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเกมส์

บริการที่ 4 นี้โปรแกรมจะทำการเพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเกมส์ โดยทำการเพิ่มค่าเดิมตามค่า Input ที่เรียกใช้ในการทำงานบริการที่ 3 นี้โปรแกรมที่ประมวลผลบนหน่วยประมวลผลกลางจะทำงานในรูปแบบที่แตกต่างกันกับโปรแกรมที่ประมวลผลบนหน่วยประมวลผลทางกราฟิกโดยที่

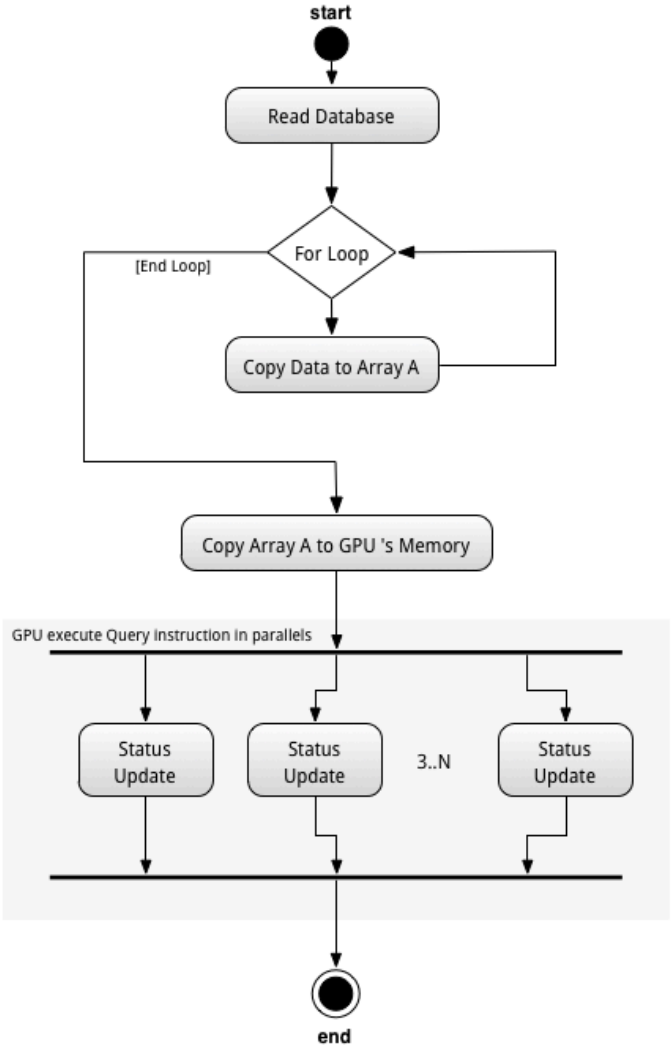
โปรแกรมบนหน่วยประมวลผลกลางจะทำงานดังต่อไปนี้



รูปที่ 3.18 แผนภาพการทำงานของบริการเพิ่มคะแนนบนหน่วยประมวลผลกลาง

รูปที่ 3.18 นี้ได้แสดงให้เห็นว่าการทำงานขั้นตอนที่ 2 นี้ (ในพื้นที่สีเทา) โปรแกรมแม่ข่ายที่ประมวลผลบนหน่วยประมวลผลกลางจะใช้วิธีการวน for loop ทั้งหมดเพื่อที่จะเพิ่มค่าใน Array A

โปรแกรมบนหน่วยประมวลผลกราฟิกจะทำงานดังต่อไปนี้



รูปที่ 3.19 แผนภาพการทำงานของบริการเพิ่มคะแนนบนหน่วยประมวลผลกราฟิก

รูปที่ 3.19 นี้ได้แสดงให้เห็นว่าการทำงานขั้นตอนที่ 2 นี้ (ในพื้นที่สีเทา) โปรแกรมแม่ข่ายที่ประมวลผลบนหน่วยประมวลผลกลางจะใช้วิธีการสั่งให้มีการรัน Kernel เพื่อเพิ่มค่าไปยัง Array A ทั้งหมดทุกตัวโดยรันในแบบขนานพร้อมๆกันทีล่ะ 248 คำสั่ง

5 บริการส่ง Output คืนเครื่องลูกข่าย

บริการที่ 4 จะทำการคืนค่า Output ที่ได้รับจาก บริการที่ผู้ใช้งานเลือกใช้ไปยังเครื่องลูกข่าย ใช้ภาษา PHP ในการเรียกใช้งานโดยใช้คำสั่งดังต่อไปนี้

```
1.  $answer = '<return>$pnt</return>'
2.  return $answer;
```

โดย $pnt คือค่าที่ถูกส่งกลับมาจากบริการที่เรียกใช้

บทที่ 4

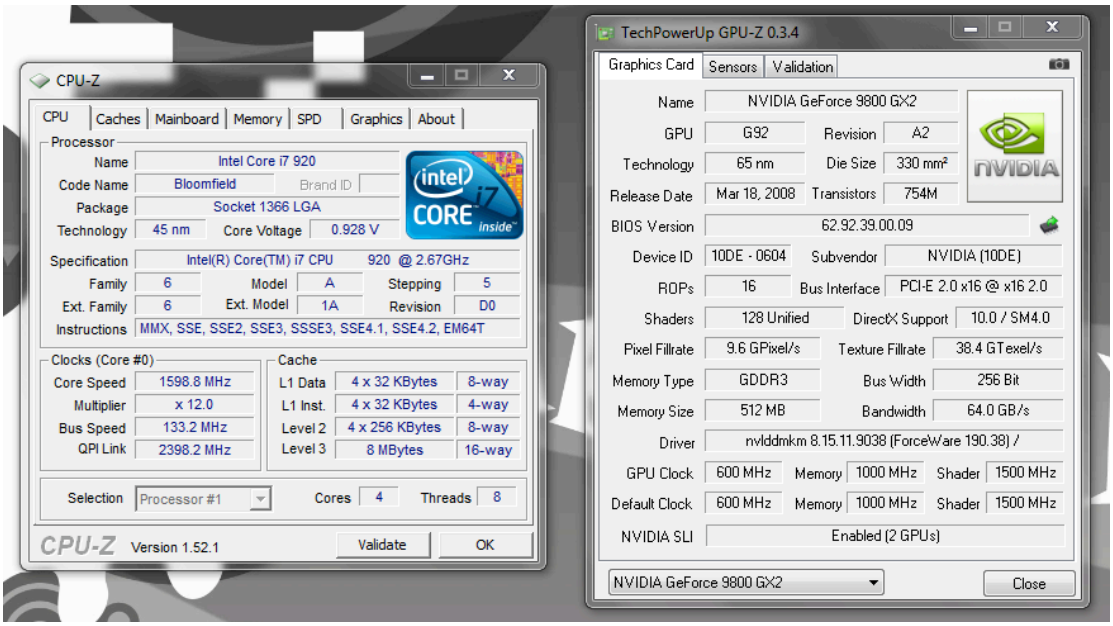
ผลการทดสอบ

4.1 เกริ่นนำ

บทนี้นำเสนอผลการทดสอบประสิทธิภาพของโปรแกรมแม่ข่ายเกมส์ออนไลน์ตัวอย่างที่ใช้สถาปัตยกรรมที่ได้เสนอแนะไว้ในบทที่ 3 ผ่านโปรแกรมแม่ข่ายเกมส์ออนไลน์ทั้ง 2 ประเภทคือ เกมส์ออนไลนชนิดมีผู้เล่นคนเดียว และ เกมส์ออนไลน์ชนิดที่มีผู้เล่นพร้อมๆกันหลายคน

4.2 ฮารด์แวร์ที่ใช้ในการทดสอบ

ในการทดสอบประสิทธิภาพของโปรแกรมแม่ข่ายเกมส์ออนไลน์ตัวอย่างนั้นได้ทำการทดสอบโดยใช้อุปกรณ์ที่มีขายทั่วไปตามท้องตลาดในการทดสอบโดยมีอุปกรณ์ที่ใช้ทดสอบดังนี้



รูปที่ 4.1 รายละเอียดของอุปกรณ์ที่ใช้ในการทดสอบโดยผ่านโปรแกรม CPU-Z (หน่วยประมวลผลกลาง) และ GPU-Z (หน่วยประมวลผลกราฟฟิก)

1. เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเครื่องแม่ข่าย 1 เครื่องมีคุณสมบัติดังต่อไปนี้

   หน่วยประมวลผลกลาง : Intel Core i7 920 @ 2.67 GHz (Quad Core CPU)

   หน่วยประมวลผลทางกราฟฟิก : NVIDIA GeForce 9800 GX2 (256 Stream processor) 1 GB GDDR3 Memory ทำงานอยู่บน PCI-E 2.0 Bus

   หน่วยความจำหลักของระบบ : 3 x 1 GB DDR III Run in Triple Channel Mode

   ชิปเซ็ท : Intel X58 Chipset

   ระบบปฏิบัติการ : Microsoft Windows 7 Ultimate 32-bit

   ไดรเวอร์ของหน่วยประมวลผลทางกราฟฟิก : NVIDIA ForceWare version 190.38
2. เครื่องคอมพิวเตอร์ที่ทำหน้าที่จำลองคำสั่งแล้วส่งไปยังเครื่องแม่ข่าย 1 เครื่อง
3. อุปกรณ์เน็ทเวิรค์ที่ใช้เชื่อมต่อเครื่องคอมพิวเตอร์ทั้ง 2 เครื่อง Apple AirPort Extreme base station โดยที่ทั้ง 2 เครื่องเชื่อมต่อกันโดยใช้มาตรฐาน Gigabit Ethernet ในการเชื่อมต่อ

4.3 การทดสอบเกมส์ออนไลน์แบบผู้เล่นคนเดียว

การทดสอบในสถานะการณแบบแรกคือการทดสอบการใช้ สถาปัตยกรรมที่ได้ออกแบบนี้ในเกมส์ออนไลน์แบบที่มีผู้เล่นเพียงคนเดียว แข่งกับเครื่องแม่ข่าย (เช่น เกมส์หมากรุกออนไลน์ที่แข่งกับ AI บนเซอร์เวอร์ หรือ Flash-based games ที่เก็บค่าคะแนนสูงสุดบนเครื่องแม่ข่าย เป็นต้น) โดยในการทดสอบจะใช้แม่ข่ายเกมส์ออนไลน์ตัวอย่างคือโปรแกรมแม่ข่ายเกมส์ Tic Tac Toe แบบเล่นผ่านเครือข่ายอินเตอร์เน็ท (ตามที่ได้ออกแบบไว้ในบทที่ 3 หัวข้อ 3.4.1)

ในการทดลองนี้มีจุดมุ่งหมาย 3 ประการคือ 1) ทดสอบว่าหน่วยประมวลผลทางกราฟฟิก จะสามารถเพิ่มประสิทธิภาพของเวลาการตอบกลับ ของโปรแกรมแม่ข่ายเกมส์ออนไลน์ในสถานการณ์แบบ 1:1 ได้หรือไม่ 2) หน่วยประมวลผลทางกราฟฟิกจะสามารถลดการใช้งานหน่วยประมวลผลกลางในเกมส์ออนไลนได้หรือไม่ และ 3) สามารถจัดแบ่งชนิดภาระงานที่เหมาะสมสำหรับการทำงานบนหน่วยประมวลผลทางกราฟฟิกบางประเภทได้

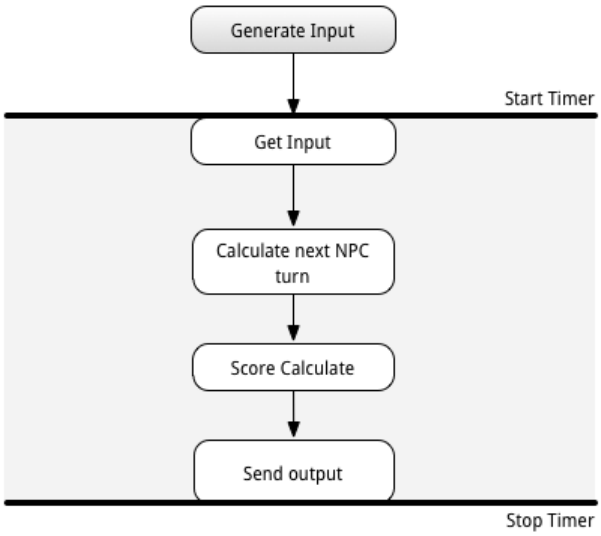
4.3.1 การทดลองด้านประสิทธิภาพในการประมวลผล

4.3.1.1 วัตถุประสงค์ในการทดสอบ

1. เพื่อทำการทดสอบว่าวิธีการแบ่งงานไปยังหน่วยประมวลผลทางกราฟิกสามารถเพิ่มประสิทธิภาพในด้านความเร็วในการประมวลผลได้เมื่อเทียบกับการทำงานบนหน่วยประมวลผลกลางเพียงอย่างเดียว
2. เพื่อทำการทดสอบว่าภาระงานชนิดใดที่เหมาะสมกับการแบ่งภาระงานนั้นไปประมวลผลบนหน่วยประมวลผลทางกราฟิก

4.3.1.2 วิธีการทดสอบ

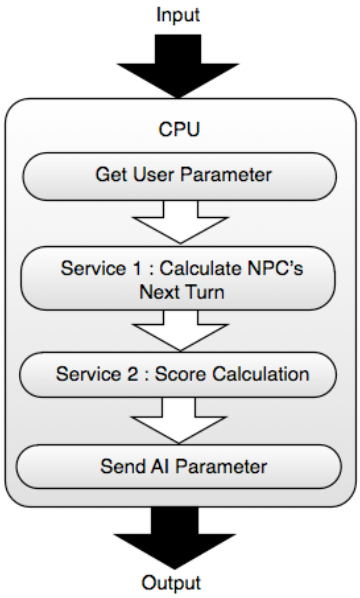
การทดลองด้านประสิทธิภาพในการประมวลผลคำสั่ง 1 ภาระงานจะทดสอบโดยทดสอบจับเวลาในการประมวลผลของโปรแกรมแม่ข่ายตั้งแต่เริ่มต้นการทำงานของโปรแกรมแม่ข่าย กล่าวคือ ตั้งแต่รับค่า parameter ของผู้ใช้งาน จนกระทั่งหยุดการจับเวลาเมื่อบริการทั้งหมดในส่วนโปรแกรมแม่ข่ายประมวลผลเสร็จสิ้นดั่งรูป



รูปที่ 4.2 ช่วงของการจับเวลาในการทดสอบที่ 4.2.1

ในการทดสอบที่ 1 นี้จะทำการแบ่งโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายโดยใช้สถาปัตย์กรรมเกมส์ออนไลน์ที่ได้ออกแบบไว้ในบทที่ 3 เพื่อการจัดลำดับงาน เป็น 4 รูปแบบ คือ

รูปแบบที่ 1 โปรแกรมแม่ข่ายทุกบริการทำงานบนหน่วยประมวลผลกลาง



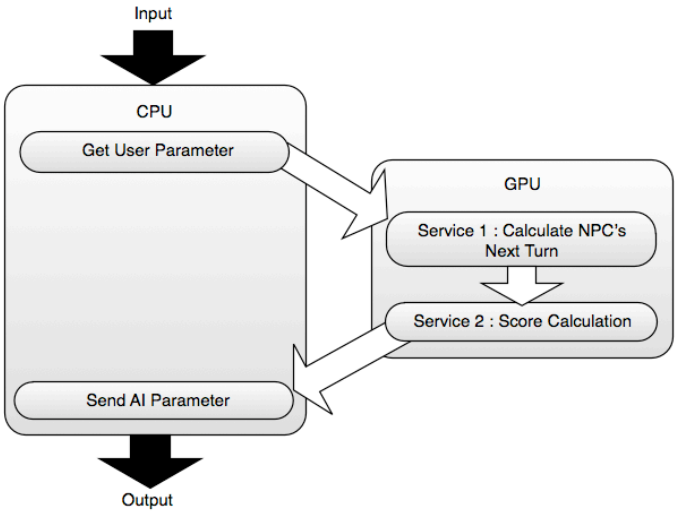
รูปที่ 4.3 โครงสร้างของโปรแกรมแม่ข่ายแบบทุกบริการทำงานบนหน่วยประมวลผลกลาง

รูปที่ 4.3 ได้แสดงให้เห็นถึงโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบที่ 1 ซึ่งเป็นการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบเดิม กล่าวคือ ภาระงานทุกชนิดจะถูกประมวลผลโดยใช้หน่วยประมวลผลกลางในการประมวลผล

รูปแบบที่ 2 โปรแกรมแม่ข่ายบริการที่ 1 และ 2 ถูกนำไปประมวลผลบนหน่วยประมวลผลทางกราฟิก

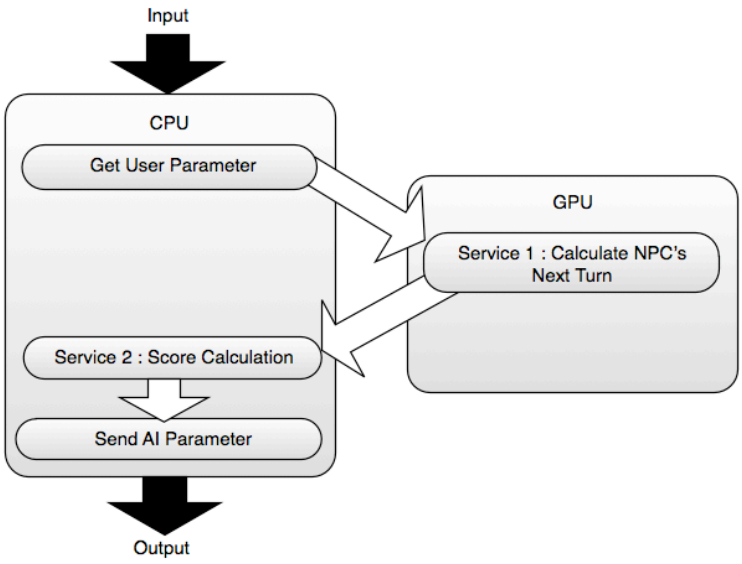
รูปที่ 4.4 ได้แสดงให้เห็นถึงโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบที่ 2 ซึ่งเป็นการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์โดยแบ่งบริการบางส่วนไปทำงานบนหน่วยประมวลผล

กราฟิกแต่ไม่ได้มีการแบ่งแยกประเภทภาระงานใดๆ ทุกบริการที่สามารถประมวลผลภาระงานบนหน่วยประมวลผลทางกราฟิกถูกย้ายมาประมวลผลที่หน่วยประมวลผลกราฟิกทั้งหมด



รูปที่ 4.4 โครงสร้างของโปรแกรมแม่ข่ายรูปแบบที่ 2

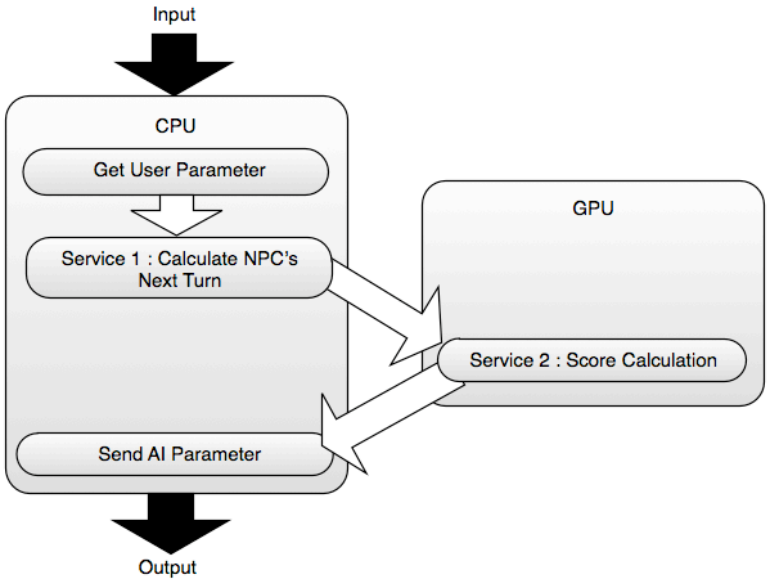
รูปแบบที่ 3 โปรแกรมแม่ข่ายเฉพาะบริการที่ 1 ถูกนำไปประมวลผลบนหน่วยประมวลผลทางกราฟิก



รูปที่ 4.5 โครงสร้างของโปรแกรมแม่ข่ายรูปแบบที่ 3

จากรูปที่ 4.5 ได้แสดงให้เห็นถึงโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบที่ 3 ซึ่งเป็นการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์โดยแบ่งบริการบางส่วนไปทำงานบนหน่วยประมวลผลทางการฟิก โดยแบ่งงานที่มีการประมวลผลที่มีจำนวนมากๆ (ในที่นี้คือการประมวลผลตำแหน่งเดินของ NPC 1 จาก 9! ค่าที่เป็นไปได้) ไปประมวลผลภาระงานบนหน่วยประมวลผลทางกราฟิก

รูปแบบที่ 4 โปรแกรมแม่ข่ายเฉพาะบริการที่ 2 ถูกนำไปประมวลผลบนหน่วยประมวลผลทางกราฟิก



รูปที่ 4.6 โครงสร้างของโปรแกรมแม่ข่ายรูปแบบที่ 4

จากรูปที่ 4.6 ได้แสดงให้เห็นถึงโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบที่ 4 ซึ่งเป็นการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์โดยแบ่งบริการบางส่วนไปทำงานบนหน่วยประมวลผลทางการฟิก โดยแบ่งงานที่มีการประมวลผลที่ไม่ได้มีความซับซ้อน (ในที่นี้คือการคำนวนคะแนนของผู้เล่นที่มีแค่การบวกค่าเพิ่มเท่านั้น) ไปประมวลผลภาระงานบนหน่วยประมวลผลทางกราฟิก

4.3.1.3 ผลการทดสอบ : เวลาตอบสนองของ 1 ภาระงาน (Tic-tac-toe)

หลังจากที่ได้ทำการแบ่งโปรแกรมแม่ข่ายเกมส์ออนไลน์ออกมาเป็น 4 รูปแบบที่มีการแบ่งภาระงานในลักษณะที่แตกต่างกันแล้วออกมาทั้ง 4 รูปแบบ จึงนำมาทำการทดสอบประสิทธิภาพในการประมวลผลภาระงานทั้งหมด ในการจับเวลาการทำงานนั้นจะทำการจับเวลาทั้งหมด 5 ครั้งต่อ 1 รูปแบบ แล้วจึงนำค่าเฉลี่ยของเวลาที่แต่ละรูปแบบใช้นั้นมาบันทึกเพื่อวิเคราะห์ผลในขั้นตอนต่อไป ผลที่ได้เป็นตารางดังต่อไปนี้

ตารางที่ 1 ผลจากการทดลองที่ 4.3.1.3 ด้านประสิทธิภาพในการประมวลผลภาระงาน

จำนวนคำสั่ง	รูปแบบที่ 1	รูปแบบที่ 2	รูปแบบที่ 3	รูปแบบที่ 4
1	1.209	0.074	0.046	1.237
10	11.235	0.742	0.112	12.154
100	98.362	4.235	0.312	101.165

4.3.1.4 สรุปผลการทดสอบ

จะเห็นได้ว่าผลจากการทดสอบนั้นได้แสดงให้เห็นว่าภารแบ่งภาระงานไปทำงานบนหน่วยประมวลผลกราฟิกได้เพิ่มประสิทธิภาพในการประมวลผลคำสั่งของโปรแกรมแม่ข่ายเกมส์ออนไลน์แบบผู้เล่นคนเดียวได้อย่างชัดเจน โดยเฉพาะเมื่อเทียบกับการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบเดิม (รูปแบบที่ 1) จะเห็นได้ว่าการแบ่งภาระงานของโปรแกรมแม่ข่ายเกมส์ออนไลน์ไปประมวลผลบนหน่วยประมวลผลทางกราฟิกนั้นสามารถเพิ่มประสิทธิภาพในด้านการตอบสนองของเกมส์ออนไลน์ส่วนแม่ข่ายได้อย่างมีนัยยะสำคัญ

เมื่อลองพิจารณาผลการทดสอบจะเห็นได้ว่า

1. กรณีที่จะช่วยเพิ่มประสิทธิภาพได้สูงที่สุดคือการแบ่งงานใน รูปแบบที่ 3 คือ การแบ่งงานที่มีการประมวลผลที่มีจำนวนมากๆ ไปประมวลผลภาระงานบนหน่วยประมวลผลทางกราฟิกโดยสามารเพิ่มประสิทธิภาพด้านความเร็วในการประมวลผลได้อย่างมีนัยยะสำคัญ เมื่อเทียบกับการ

พัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบเดิม (รูปแบบที่ 1 ทั้ง 2 ภาระงานประมวลผลบนหน่วยประมวลผลกลาง)

2. ในการทดสอบรูปแบบที่ 4 คือ การแบ่งงานที่มีการประมวลผลที่ไม่ได้มีความซับซ้อน (ในที่นี้คือการคำนวนคะแนนของผู้เล่นที่มีแค่การบวกค่าเพิ่มเท่านั้น) ไปประมวลผลภาระงานบนหน่วยประมวลผลกราฟิก และภาระงานที่มีความซับซ้อนมากๆประมวลผลบนหน่วยประมวลผลกลาง ปรากฏว่าการแบ่งงานไปประมวลผลบนหน่วยประมวลผลกราฟฟิกนั้นทำให้ประสิทธิภาพด้านความเร็วในการประมวลผลในการทำงานแย่ลง
3. ในการทดสอบรูปแบบที่ 2 คือภาระงานทั้ง 2 ชนิดทำงานบนหน่วยประมวลผลกราฟิก ถึงแม้ว่าจะไม่ได้เป็นวิธีการแบ่งภาระงานที่ดีที่สุด แต่ก็ยังสามารถเพิ่มประสิทธิภาพด้านความเร็วในการประมวลผลจากการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบเดิม (รูปแบบที่ 1 ทั้ง 2 ภาระงานประมวลผลบนหน่วยประมวลผลกลาง) ได้

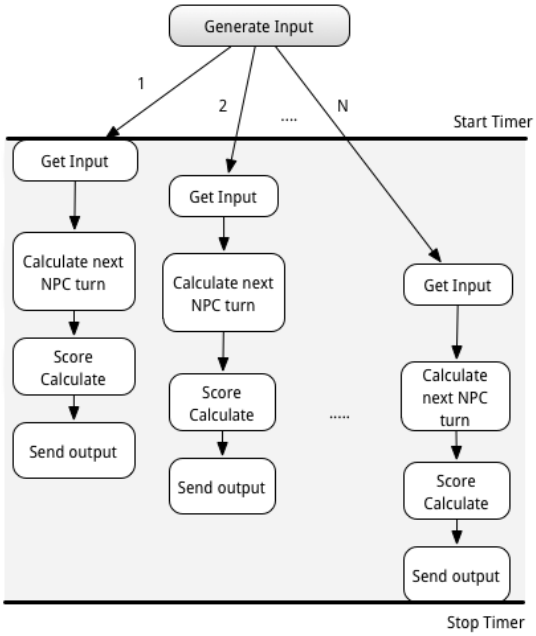
4.3.2 การใช้ทรัพยากรหน่วยประมวลผลกลาง

4.3.2.1 วัตถุประสงค์ในการทดสอบ

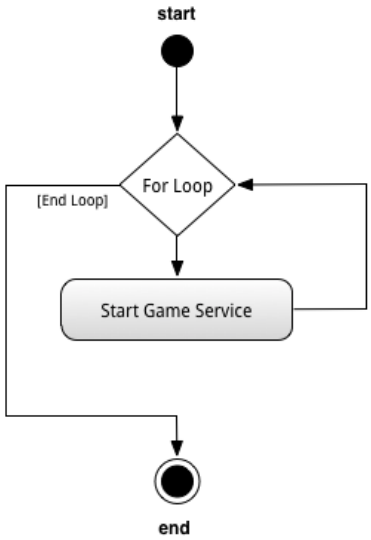
เพื่อทำการทดสอบว่าวิธีการแบ่งงานไปยังหน่วยประมวลผลทางกราฟิกสามารถเพิ่มประสิทธิภาพด้านการลดการใช้งานหน่วยประมวลผลกลางได้เมื่อเทียบกับการทำงานบนหน่วยประมวลผลกลางเพียงอย่างเดียว

4.3.2.2 วิธีการทดสอบ

ในการทดสอบที่ 2 นี้จะเป็นการทดสอบจำนวนการใช้งานทรัพยากรหน่วยประมวลผลกลางของโปรแกรมแม่ข่ายเกมส์ออนไลน์จะทดสอบโดยการจำลองคำสั่งเรียกใช้งานโปรแกรมแม่ข่ายจากเครื่องลูกข่ายเป็นจำนวน 100 คำสั่ง และ 1,000 คำสั่ง และทำการบันทึกผลการใช้งานหน่วยประมวลผลกลางตั้งแต่เริ่มส่งภาระงานที่ 1 จนประมวลผลภาระงานคำสั่งสุดท้ายเสร็จสิ้นดั่งรูป 4.8



รูปที่ 4.7 การบันทึกการใช้ทรัพยากรหน่วยประมวลผลกลาง ในการทดสอบที่ 4.3.2



รูปที่ 4.8 โครงสร้างการทำงานของโปรแกรมจำลองคำสั่งสำหรับการทดสอบที่ 4.3.2

ในการจำลองคำสั่งนั้นจะใช้มีการพัฒนาโปรแกรมโดยใช้โครงสร้างการทำงานในรูปที่ 4.9 ในการพัฒนา กล่าวคือจะมีการวนลูป for ทั้งหมด N ครั้ง (100 หรือ 1,000 ครั้ง) เพื่อที่จะเรียกใช้โปรแกรมในส่วนของเครื่องแม่ข่ายโดยใส่ parameter 2 ค่าดังต่อไปนี้

```
1.  gamearray=1532;
2.  userpoint=1000;
```

โดยค่า gamearray เป็นค่าสถานะล่าสุดของเกมส์ ในโปรแกรมจำลองค่านี้จะส่งค่า gamearray = 1532 ซึ่งมีความหมายว่า

การเล่นที่มีรูปแบบการเดินรอบแรกตำแหน่งที่ 1

การเดินในรอบที่ 2 เดินตำแหน่งที่ 5

การเดินในรอบที่ 3 เดินตำแหน่งที่ 3

การเดินในรอบที่ 4 เดินตำแหน่งที่ 2

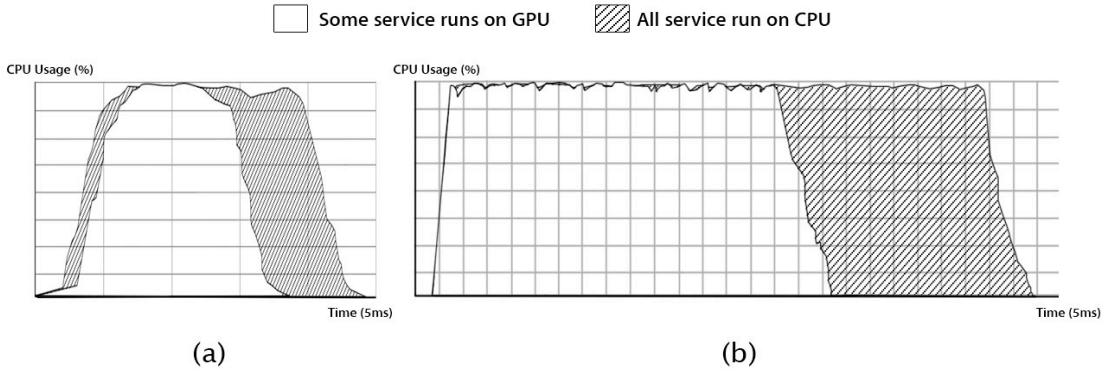
และ userpoint มีค่าเท่ากับ 1000 คะแนน

อนึ่งในการทดสอบการใช้ทรัพยากรหน่วยประมวลผลกลางของโปรแกรมเกมส์แม่ข่ายออนไลน์นี้นั้นในส่วนของรูปแบบการแบ่งภาระงานของโปรแกรมแม่ข่ายนั้นจะใช้ 2 รูปแบบในการทดสอบ คือ

1 โปรแกรมแม่ข่ายทุกบริการทำงานบนหน่วยประมวลผลกลาง (รูปแบบที่ 1)

2 โปรแกรมแม่ข่ายเฉพาะบริการที่ 1 ประมวลผลบนหน่วยประมวลผลกราฟิก (รูปแบบที่ 3)

4.3.2.3 ผลการทดสอบ



รูปที่ 4.9 การใช้ทรัพยากรหน่วยประมวลผลกลาง เมื่อรันโปรแกรม (a) 100 และ (b) 1,000 คำสั่ง

4.3.2.4 สรุปผลการทดสอบ

จากผลการทดสอบจะเห็นได้ว่าการแบ่งภาระงานจากหน่วยประมวลผลกลางมายังหน่วยประมวลผลทางกราฟิกจะช่วยลดการใช้งานหน่วยประมวลผลกลางอย่างเห็นได้ชัด กล่าวคือจะลดการใช้งานทรัพยากรหน่วยประมวลผลกลางประมาณ 33%

4.4 เกมส์ออนไลนในรูปแบบที่มีผู้เล่นพร้อมกันหลายคน (Massive multiplayer online game)

การทดสอบในสถานะการณที่สองคือการทดสอบการใช้ วิธีการนี้ในเกมส์ออนไลน์แบบที่มีผู้เล่นพร้อมๆกันหลายคน (เช่น เกมส์ MMORPG เป็นต้น) โดยในการทดสอบจะใช้โปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายตัวอย่าง คือ เกมส์จำลองการซื้อขายหุ้นในตลาดหลักทรัพย์ (Virtual online stock exchange games)

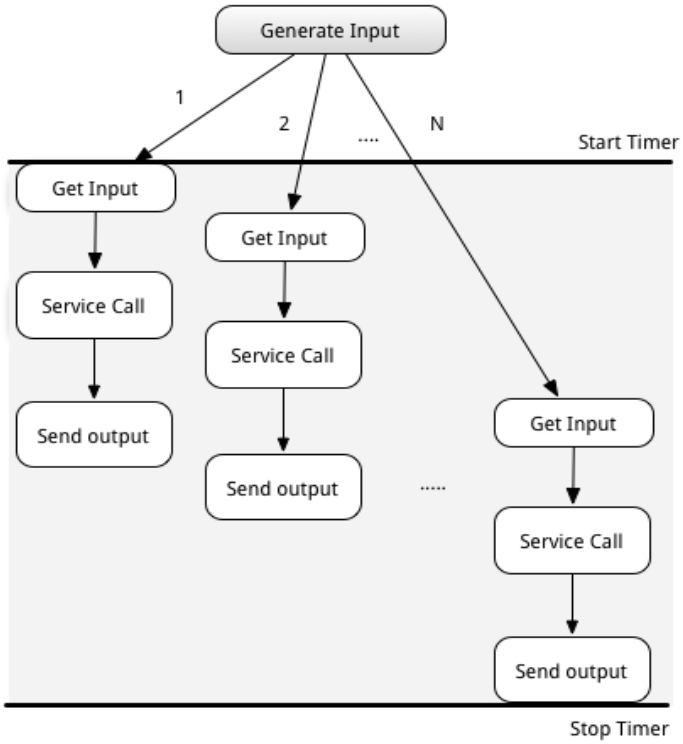
4.4.1 การทดสอบประสิทธิภาพของเวลาตอบสนองของโปรแกรม

4.4.1.1 วัตถุประสงค์ในการทดสอบ

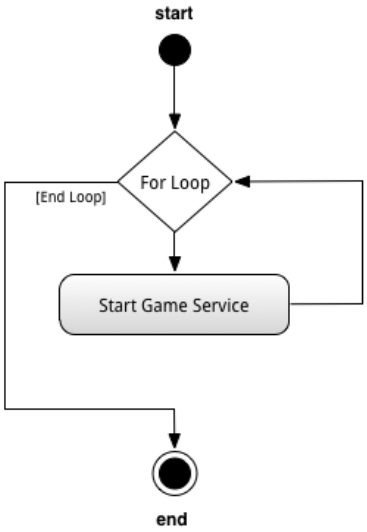
1. ทดสอบว่าหน่วยประมวลผลกราฟิกสามารถเพิ่มประสิทธิภาพในด้านเวลาตอบกลับของโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายในสถานการณ์แบบผู้เล่นจำนวนมากเล่นเกมส์เดียวกันได้ในระดับใด
2. สามารถจัดแบ่งชนิดงานที่เหมาะสมสำหรับการทำงานบนหน่วยประมวลผลกราฟิกบางประเภทได้ (โดยเฉพาะอย่างยิ่งงานในแบบอนุกรม์ในสถานการณ์ที่มีคนใช้งานพร้อมกันจำนวนมาก)

4.4.1.2 วิธีการทดสอบ

ในการทดสอบนี้จะเป็นการทดสอบประสิทธิภาพด้านเวลาตอบสนองของบริการชนิดต่างๆ โดยการทดสอบนั้นจะทำการจำลองคำสั่งเรียกใช้งานโปรแกรมแม่ข่ายจากเครื่องลูกข่ายเป็นจำนวนมากและทำการบันทึกผลการใช้งานหน่วยประมวลผลกลางตั้งแต่เริ่มส่งภาระงานที่ 1 จนประมวลผลภาระงานคำสั่งสุดท้ายเสร็จสิ้นดั่งรูปที่ 4.11



รูปที่ 4.10 การใช้ทรัพยากรหน่วยประมวลผลกลาง ในการทดสอบที่ 4.4.1



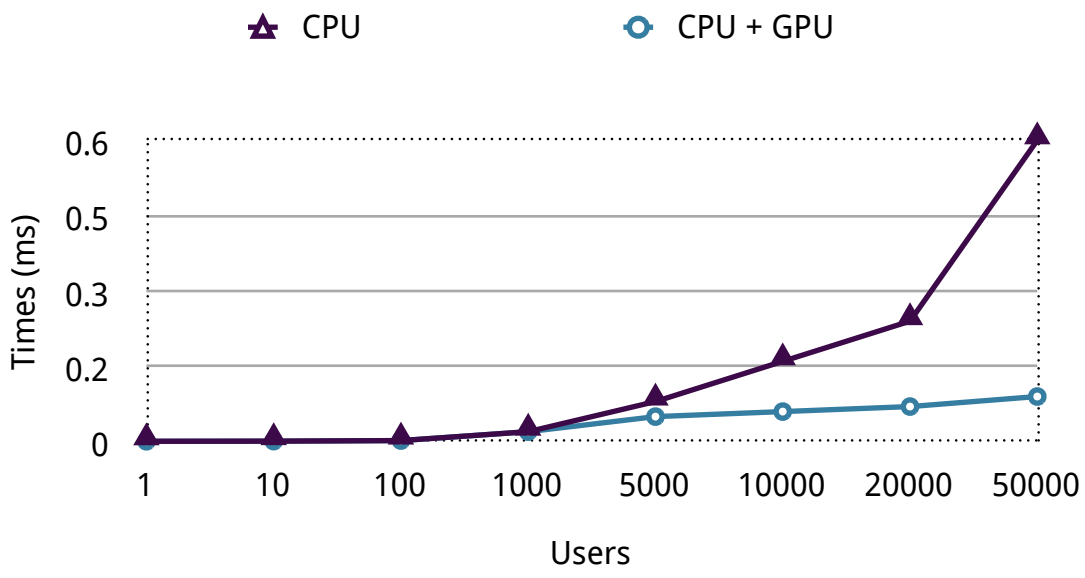
รูปที่ 4.11 โครงสร้างการทำงานของโปรแกรมจำลองคำสั่งสำหรับการทดสอบที่ 4.4.1

4.3.1.2 ผลการทดสอบ

การทดสอบเกมส์แม่ข่ายแบบผู้เล่นพร้อมๆกันหลายๆคนนั้นได้ทำการทดสอบ 3 การทดสอบดังต่อไปนี้

1 ผลการทดสอบประสิทธิภาพของเวลาตอบสนองบริการจัดเรียงคะแนน (Score ranking)

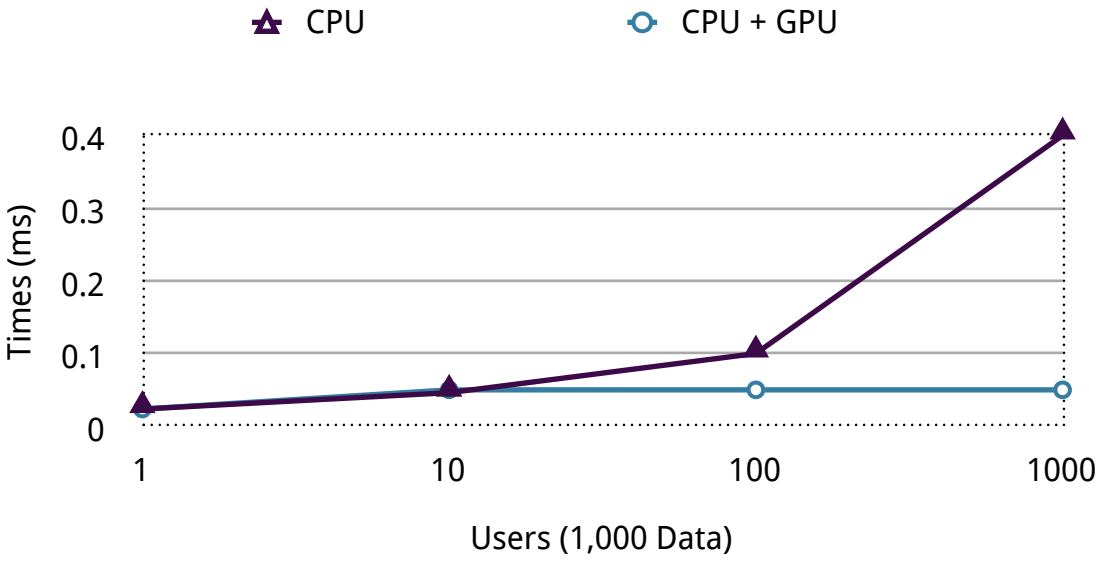
ในแง่มุมแรกที่ได้ทำการทดสอบคือ ประสิทธิภาพการตอบกลับที่ได้จากการรันบริการการจัดเรียงคะแนน โดยการทดสอบนั้นจะใช้วิธีการจับเวลาการตอบกลับจากเริ่มต้นเรียกภาระงานจนจบการเรียกภาระงาน โดยในขั้นตอนนี้จะมีการเพิ่มจำนวนข้อมูลของผู้ใช้งานเพิ่มขึ้นเรื่อยๆ โดยผลที่ได้เป็นดังต่อไปนี้



รูปที่ 4.12 ผลการทดสอบประสิทธิภาพเวลาตอบสนองบริการจัดเรียงคะแนน

2 ผลการทดสอบประสิทธิภาพของเวลาตอบสนองบริการค้นหารายชื่อหุ้น (Searching)

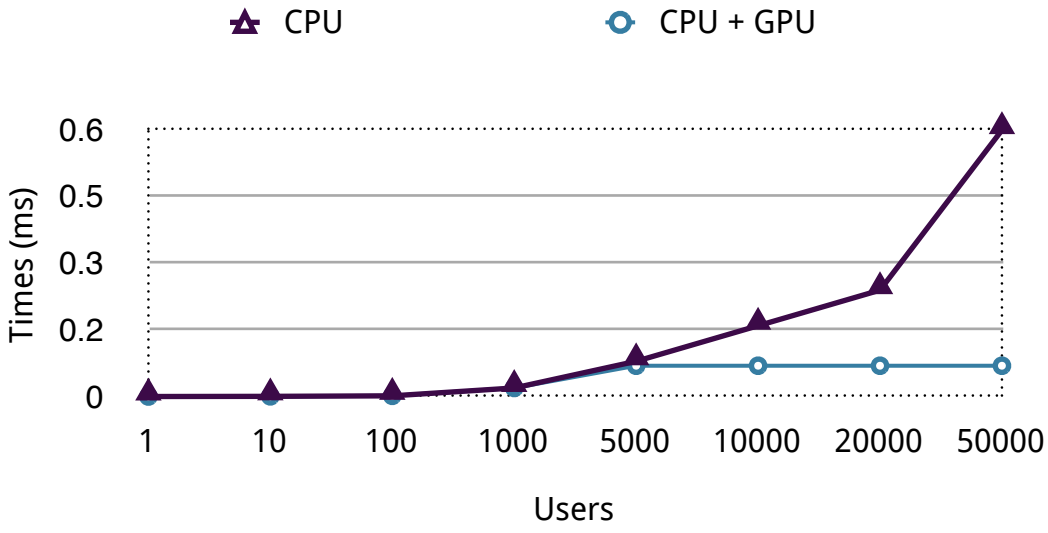
ในแง่มุมที่ 2 ที่ได้ทำการทดสอบคือ ประสิทธิภาพการตอบกลับที่ได้จากการรันบริการการค้นหาหุ้น 1 ตังจากทั้งหมด 1000 ตัว โดยการทดสอบนั้นจะใช้วิธีการจับเวลาการตอบกลับจากเริ่มต้นเรียกภาระงานจนจบการเรียกภาระงาน โดยในขั้นตอนนี้จะมีการจำลองเพิ่มจำนวนคำสั่งของผู้ใช้งานที่เรียกใช้ฟังชั่นนี้พร้อมกันเรื่อยๆ โดยผลที่ได้เป็นดังต่อไปนี้



รูปที่ 4.13 ผลการทดสอบประสิทธิภาพเวลาตอบสนองบริการค้นหารายชื่อหุ้น

3 ผลการทดสอบประสิทธิภาพของเวลาตอบสนองบริการปรับปรุงสถานะของผู้เล่น

ภาระงานที่ 3 ที่ได้ทดสอบนั้นเป็นภาระงานที่จะใช้เพิ่มคะแนนไปยังผู้เล่นทุกคนที่อยู่ในเซอร์เวอร์ โดยการทดสอบนั้นจะทำการจับเวลาการตอบกลับจากเริ่มต้นเรียกภาระงานจนจบการเรียกภาระงาน โดยในขั้นตอนนี้จะมีการจำลองเพิ่มจำนวนคำสั่งของผู้ใช้งานที่เรียกใช้ฟังชั่นนี้พร้อมกันเรื่อยๆ โดยผลที่ได้เป็นดังต่อไปนี้



รูปที่ 4.14 ผลการทดสอบประสิทธิภาพเวลาตอบสนองบริการปรับปรุงสถานะของผู้เล่น

4.4.1.3 สรุปผลผลการทดสอบ

จากการทดสอบที่ 4.4 ทั้ง 3 การทดสอบนั้นสามารถสรุปผลได้ดังต่อไปนี้

1. ด้านประสิทธิภาพในการรักษาประสิทธิภาพในการด้านเวลาตอบกลับ

จากผลการทดสอบทั้ง 3 ตอนนั้น ได้แสดงให้เห็นว่าหน่วยประมวลผลกราฟิก สามารถช่วยรักษาคุณภาพเวลาตอบกลับได้อย่างมีประสิทธิภาพโดยเฉพาะอย่างยิ่ง ในสภาวะที่มีผู้ใช้พร้อมกันจำนวนมาก

2 . ด้านการแบ่งภาระงาน

จากผลการทดสอบที่ 1 และ 2 ได้ยืนยันผลจากการทดสอบในตอนที่ 1 ที่ว่าเมื่อ ภาระงานสามารถแบ่งเป็นหลายๆส่วนได้ จะมีความเหมาะสมมากที่จะนำมาประยุกต์ใช้งานบนหน่วยประมวลผลกราฟิก ตามผมการทดสอบนั้นจะเห็นได้ว่าผลจากการที่สามารถแบ่งงานเป็นจำนวนมากได้นั้น บนหน่วยประมวลผลกราฟิกจะมีความเร็วในการตอบกลับสูงกว่าการใช้งานหน่วยประมวลผลกลางอยู่มากเลยทีเดียว

แต่ในการทดสอบที่ 3 ได้แย้งข้อสรุปจากผลการทดสอบที่ 4.3 ที่ว่าภาระงานที่ไม่สามารถแบ่งส่วนได้จะไม่เหมาะกับการประยุกต์บนหน่วยประมวลผลกราฟิก การทดสอบที่ 3 ได้แสดงให้เห็นว่าถึงแม้ว่าจะเป็นงานที่ไม่เหมาะต่อการใช้งานบนหน่วยประมวลผลกราฟิกก็ตาม แต่ถ้าหากมีจวนวนการทำงานพร้อมกันมากๆแล้วหน่วยประมวลผลกราฟิกก็สามารถช่วยแบ่งเบาภาระงานของหน่วยประมวลผลกลางได้เช่นกัน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 ผลสรุปจากวัตถุประสงค์งานวิจัย

วิทยานิพนธ์นี้ได้นำเสนอแนวทางการแก้ไขปัญหาในด้านประสิทธิภาพการตอบสนองของการให้บริการเกมส์ออนไลน์ โดยใช้หน่วยประมวลผลกราฟิกเข้ามาช่วย ซึ่งผลการวิจัยในบทที่ผ่านมานั้นได้ตอบคำถามจากวัตถุประสงค์ในงานวิจัยนี้ทั้ง 2 ข้อดังต่อไปนี้

5.1.1 เพื่อศึกษาประสิทธิภาพการจัดการปัญหาความคับคั่งที่เครื่องแม่ข่ายให้บริการเกมส์ออนไลน์โดยใช้แนวทางในการพัฒนาโปรแกรมด้วยสถาปัตยกรรมเชิงบริการและเทคนิค GPGPU

จากผลการวิจัยในบทที่ 3 และ 4 ได้แสดงให้เห็นอย่างชัดเจนว่าแนวคิดนี้สามารถลดเวลาในการประมวลผลโดยรวมได้อย่างมีประสิทธิภาพ โดยที่ในการทดสอบในบทที่ 4 ตอนที่ 4.3 และ 4.4 ได้แสดงให้เห็นว่า ถ้าหากแบ่งภาระงานบางส่วนของโปรแกรมแม่ข่ายเกมส์ออนไลน์สามารถเพิ่มประสิทธิภาพด้านความเร็วในการประมวลผลคำสั่ง 1 ภาระงานได้ถึง 26 เท่า และยังสามารถลดเวลาการตอบกลับในสภาวะที่มีผู้ใช้งานพร้อมกันได้อย่างมีนัยยะสำคัญ เมื่อเทียบกับการพัฒนาโปรแกรมแม่ข่ายเกมส์ออนไลน์ในรูปแบบเดิม (งานทั้งหมดทำงานบนหน่วยประมวลผลกลาง) ซึ่งจะส่งผลโดยตรงต่อการลดความคับคั่งที่เครื่องแม่ข่าย เนื่องจากภาระงานที่คับคั่งอยู่นั้นจะสามารถประมวลผลจนแล้วเสร็จได้เร็วขึ้นนั้นเอง

5.1.2 เพื่อเปรียบเทียบลักษณะของงานประเภทต่างๆ ของเกมส์ออนไลน์ที่เหมาะสม สำหรับการนำไปประมวลผลบนหน่วยประมวลกราฟิก

ความแตกต่างของลักษณะงานของโปรแกรมแม่ข่ายเกมส์ออนไลน์ที่ถึงแบ่งไปประมวลผลบนหน่วยประมวลผลทางกราฟิกจะส่งผลต่อประสิทธิภาพของระบบโดยรวมอย่างชัดเจนดั่งที่ได้แสดง ให้เห็นในการทดสอบ โดยที่จะเห็นได้อย่างชัดเจนว่า

1. ลักษณะงานที่เหมาะสมกับหน่วยประมวลผลกราฟิก คือ ภาระที่แบ่งส่วนงานจำนวนมากได้

2. ลักษณะงานที่ไม่เหมาะสมกับหน่วยประมวลผลกราฟิก คือ ภาระที่แบ่งส่วนงานไม่ได้

จากข้อสรุปทั้ง 2 ข้อดังกล่าวนั้นผลมาจากความแตกต่างของลักษณะสถาปัตยกรรม์ของหน่วยประมวลผลทั้งคู่ตามที่ได้อธิบายไว้ในบทที่ 2 หัวข้อ 2.2.1 ที่ว่า สถาปัตย์กรรมของหน่วยประมวลผลกราฟิกและหน่วยประมวลกลางนั้นมีความใกล้เคียงกันมากขึ้นแตกต่างที่จำนวนของ ALU ภายในหน่วยประมวลผลกราฟิกนั้นมีจำนวนมากกว่าหน่วยประมวลผลกลางอยู่มาก แต่อย่างไรก็ตามประสิทธิภาพของ ALU ภายในหน่วยประมวลผลกลาง (ในที่นี้คือ CPU Core) ยังมีประสิทธิภาพที่ดีกว่า ALU ภายในหน่วยประมวลผลกราฟิก (ในที่นี้คือ Stream processor) อยู่มากเช่นกัน ดังนั้นจึงทำให้งานที่ไม่สามารถแบ่งส่วนประมวลผลได้นั้นไม่ได้รับข้อดีจากสถาปัตยกรรมของหน่วยประมวลผลกราฟิกจึงเป็นที่มาของผลการทดสอบดังกล่าว ดังนั้นถ้าหากพิจารณางานในสาขาการออกแบบโปรแกรมเกมส์ออนไลนส่วนแม่ข่ายจะพบว่า

1 บริการที่เหมาะสมกับการทำงานบนหน่วยประมวลผลกราฟิก คือ บริการที่สามารถแบ่งงานให้ประมวลผลในแบบขนานได้ เช่น บริการประมวลผลคะแนนจำนวนที่มีค่าเหมือนกัน (เช่น บวกคะแนนเท่าๆกันให้กับผู้ใช้ทุกคนในระบบ) บริการค้นหาข้อมูลของตัวแปรต่างๆ เป็นต้น

2 บริการที่ไม่เหมาะสมกับการทำงานบนหน่วยประมวลผลกราฟิก คือ

2.1 บริการที่ไม่สามารถแบ่งงานประมวลผลแบบขนานได้ เช่น งานคำนวนคะแนนแต่ล่ะบุคคล

2.2 บริการที่จะต้องมีส่งค่าทางเครือข่ายเน็ทเวิรค เช่น บริการการรับ/ส่งคำสั่งจากผู้เล่น

2.3 บริการที่มีการใช้งานฮารด์แวรชนิดอื่นของเครื่องแม่ข่ายโดยตรง

จากเกณฑ์การแบ่งกลุ่มบริการและผลการทดสอบในบทที่ 3 รวมกับบทสรุปข้างต้น นั้นทำให้สามารถสรุปกลุ่มบริการที่เหมาะสมหรือไม่เหมาะสมในการกระจายภาระงานของโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายไปยังหน่วยประมวลผลกราฟิกดังที่ได้สรุปในตารางที่ 2

ตารางที่ 2 ข้อเสนอแนะจากผลการทดสอบ

บริการ	คำแนะนำ	เหตุผล
1 ภาระงานด้านที่เกี่ยวกับการคำนวน (Calculation-Based Service)		
1.1 การคำนวน AI		
1.1.1 การคำนวน AI ในรูปแบบ Decision Tree	ไม่เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	สถาปัตยกรรมหน่วยประมวลผลทางกราฟิกไม่เหมาะสมกับการประมวลผลบริการรูปแบบนี้ และจากผลทดสอบตอนที่ 4.3.3.1
1.1.2 การคำนวน AI ในรูปแบบ Parallels	เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	จากผลทดสอบตอนที่ 4.3.3.1
1.2 การคำนวนคะแนน		
1.2.1 รูปแบบการคำนวนคะแนนแบบ 1:1	ไม่เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	จากผลทดสอบตอนที่ 4.3.3.1
1.2.2 รูปแบบการคำนวนคะแนนแบบ 1:Many	เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	จากผลทดสอบตอนที่ 4.4.1.2 (3)
1.2.3 รูปแบบการคำนวนคะแนนแบบ Many:Many	เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	จากผลทดสอบตอนที่ 4.4.1.2 (3)
2 ภาระงานด้านที่เกี่ยวกับข้อมูล (Information-Based Service)		
2.1 การเก็บข้อมูลลงหน่วยความจำหลัก หรือลงอุปกรณ์เก็บข้อมูล (Store)	ไม่เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	หน่วยประมวลผลทางกราฟิกสามารถเรียกใช้งานข้อมูลที่อยู่ในหน่วยความจำของตนได้เท่านั้น ไม่สามารถเข้าถึงหน่วยความจำหลักของระบบได้
2.2 การค้นหาข้อมูล (Search)	เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	จากผลทดสอบตอนที่ 4.4.1.2 (2)

บริการ	คำแนะนำ	เหตุผล
2.3 การจัดเรียงข้อมูล (Sorting)	เหมาะสมที่จะใช้งานบนหน่วยประมวลผลกราฟิก	จากผลทดสอบตอนที่ 4.4.1.2 (1)
3 ภาระงานด้านที่เกี่ยวกับการเชื่อมต่อ (Connection-Based Service)		
3.1 การจัดการการเชื่อมต่อระหว่างเครื่องลูกข่ายด้วยกัน	ไม่สามารถประมวลผลบนหน่วยประมวลผลกราฟิกได้	หน่วยประมวลผลทางกราฟิกไม่สามารถสั่งงานอุปกรณ์อื่นๆได้
3.2 การจัดการการเชื่อมต่อระหว่างเครื่องลูกข่ายและแม่ข่าย	ไม่สามารถประมวลผลบนหน่วยประมวลผลกราฟิกได้	หน่วยประมวลผลทางกราฟิกไม่สามารถสั่งงานอุปกรณ์อื่นๆได้

5.2 ข้อเสนอแนะ

5.2.1 การใช้หน่วยประมวลผลกราฟิกเข้าถึงหน่วยความจำหลักโดยตรง

เนื่องจากเวลาในระหว่างที่ทำงานวิจัยนี้นั้นยังไม่มีหน่วยประมวลผลกราฟิกตัวใดที่สามารถเข้าถึงหน่วยความจำหลักของระบบได้โดยตรงจึงจำเป็นต้องมีการคัดลอกข้อมูลจากหน่วยความจำหลักสำหรับการใช้งาน ซึ่งทำให้เกิดปัญหาขอขวดในการย้ายข้อมูลดังกล่าว ดังนั้นหากในอนาคตที่สถาปัตยกรรมหน่วยประมวลผลกราฟิกเอื้อต่อการอ้างอิงข้อมูลความจำหลัก การนำเอาหน่วยประมวลผลกราฟิกมาใช้ในโปรแกรมเกมส์ออนไลน์ส่วนแม่ข่ายอาจจะมีประสิทธิภาพสูงกว่า งานวิจัยในวิทยานิพนธ์นี้

5.2.2 ความปลอดภัยในการส่งข้อมูล

การส่งข้อมูลแบบเว็บเซอร์วิสนั้นมีการส่งในรูปแบบเอกสาร XML ซึ่งมีการใช้ภาษาที่มนุษย์เข้าใจได้ง่าย และยังไม่มีการเข้ารหัสในชั้นการสื่อสาร นั้นอาจจะก่อให้เกิดปัญหาในด้านความปลอดภัยในการส่งข้อมูลในเกมส์ออนไลน์ เช่น คะแนนของผู้เล่น ไอเท็มที่ได้รับ ซึ่งถ้าหากข้อมูลเหล่านี้สามารถเข้าถึงได้จะส่งผลต่อการให้บริการเกมส์ออนไลน์ในด้านการโกงเกมส์ ดังนั้นจึงควรพิจารณาในเรื่องความปลอดภัยในการส่งข้อมูลด้วย อาทิเช่น การใช้มาตรฐาน WS-Security มาใช้ในการส่งข้อมูล

เอกสารอ้างอิง

[1] Ian Bogost , GDC 2008: Reinventing MMOs: A Metaplace Antemortem, http:// www.watercoolergames.org/archives/000897.shtml , February 22 , (2008)

[2] Glenn-Anderson et al. , GPU-based Desktop Supercomputing. enparallel.com, (2008)

[3] Douglas K. Barry, Service-oriented architecture (SOA) definition, http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html, (2010)

[4] Hsu et al., On the Design of Multiplayer Online Video Game Systems , SPIEitcom03 Proceeding (2003)

[5] Yang et al. In-Memory Grid Files on Graphics Processors. Proceedings of the Third International Workshop on Data Management on New Hardware (DaMoN 2007) (2007)

[6] NVIDIA Corp 2009. NVIDIA CUDA Programming Guide 2.2 (2009)

[7] IBM, Building a simple yet powerful MMO game architecture, Part 1: Introduction. (2008)

[8] Shaikh et al. , Implementation of a service platform for online games. Proceedings of 3rd ACM SIGCOMM workshop (2004)

[9] W3C Working Group, Web Services Architecture (2004)

[10] Huang et al. , Modeling System Performance in MMORPG , Globecom 2004 Workshops IEEE Communications Society (2004)

[11] Trinta et al. , Evaluating a middleware for crossmedia games. Proceedings of the 5th international workshop on Middleware , (2007)

[12] Apple. OpenCL Technology Brief. Apple Mac OS X Snow Leopard (2009)

[13] Fujiki et al. NEAT-o-Games: blending physical activity and fun in the daily routine. Computers in Entertainment (CIE) (2008)

[14] Segatto et al. Mobio threat: A mobile game based on the integration of wireless technologies. Computers in Entertainment (CIE) (2008)

[15] Lang et al. Massively Multiplayer Online Worlds as a Platform for Augmented Reality Experiences. Virtual Reality Conference (2008)

[16] Cowley et al. Toward an understanding of flow in video games. Computers in Entertainment (CIE) (2008)

[17] Reynaud. GPU Powered Malware. ruxcon.org.au

[18] Abreu et al. Exposing Stream Processors as Grid Services: a GPGPU Example. inesc-id.pt

[19] Emerging GPU Technologies: ATI Stream, OpenCL and Larrabee. (2009)

[20] Christensen et al. Developing a hybrid of MMORPG and LARP using usability methods: the case of Takkar. DIGRA conference “Level Up” (2003)

[21] IBM, Building a simple yet powerful MMO game architecture, Part 2: Gaming and Web integration. (2008)

[22] IBM, Building a simple yet powerful MMO game architecture, Part 3: Capabilities and limitations. (2009)

[23] Phelan. Paper Title: The Application of Genetic Programming to General Purpose GPU Computation. ncra.ucd.ie

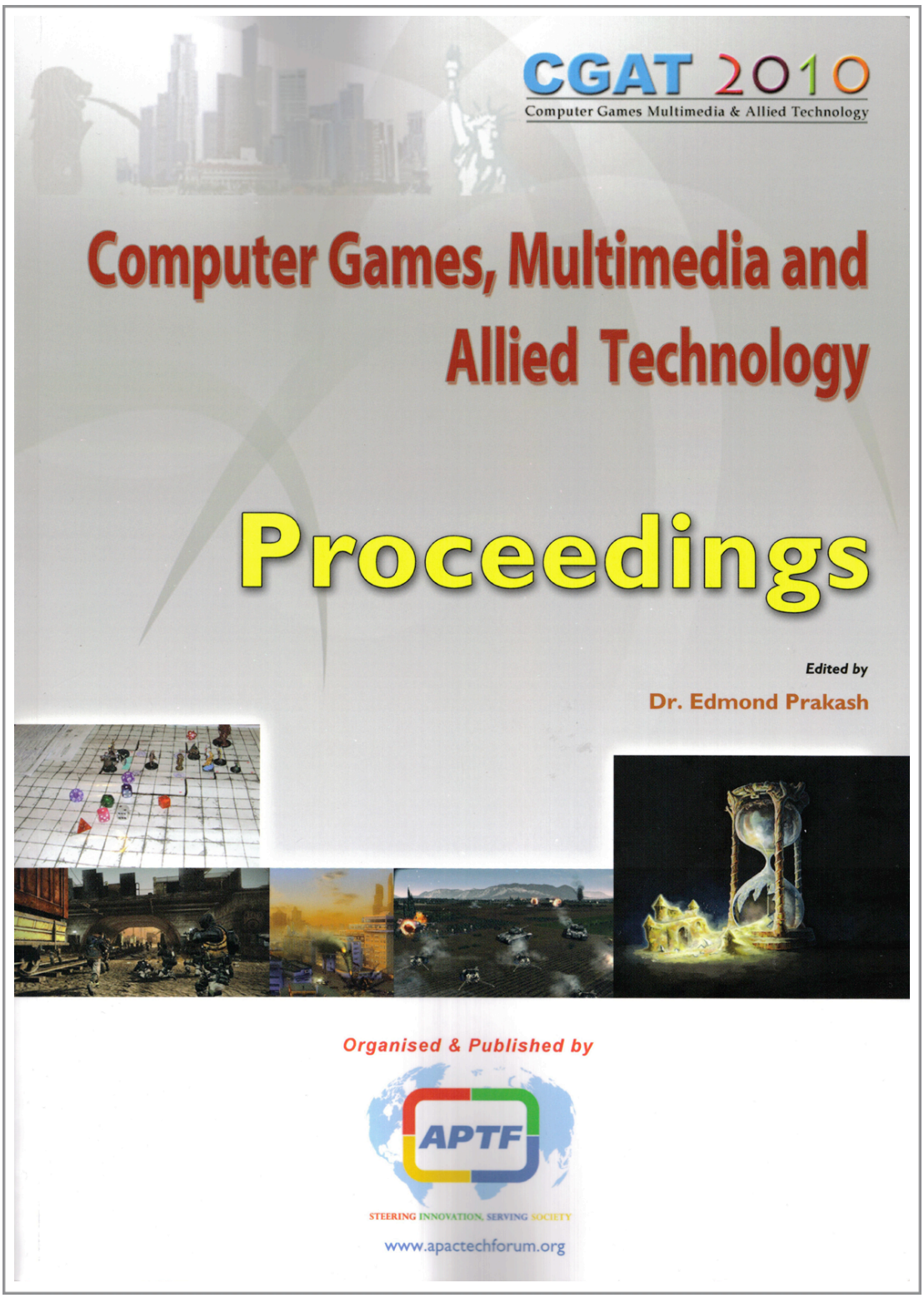
[24] Caltagirone et al. Architecture for a massively multiplayer online role playing game engine. Journal of Computing Sciences in Colleges (2002)

[25] Zamith et al. A game loop architecture for the GPU used as a math coprocessor in real-time applications. Computers in Entertainment (CIE) (2008)

[26] Owens et al. A Survey of General-Purpose Computation on Graphics Hardware. Computer Graphics Forum (2007)

ภาคผนวก

ภาคผนวก 1 ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์



3rd Annual International Conference on

Computer Games, Multimedia and Allied Technology (CGAT 2010)

April 6-7, 2010, at Singapore Management University, Singapore.

Improving Performance of Online Game Services via Graphic Processor: An Empirical Investigation

Rittichai Jitpukdeebodintra
Department of Computer Engineering
Faculty of Engineering, Prince of Songkla University
P.O. Box 2 Khohong, Hatyai, Songkhla, Thailand
Telephone: +66 86 9938555
rittichai@capsuledna.com

Suntorn Witosurapot
Department of Computer Engineering
Faculty of Engineering, Prince of Songkla University
P.O. Box 2 Khohong, Hatyai, Songkhla, Thailand
Telephone: +66 74 287369
wsuntorn@coe.psu.ac.th

ABSTRACT
A method for maintaining quality of service in game servers when there exists with excessive users is often done by increasing the number of game servers. While this method is straightforward, it demands a number of new machines to be invested without realizing local utilization of resources available in the current machines. In this paper, we argue that graphic processor (GPU) working in parallel with local central processor (CPU) inside a machine can be a good candidate for reducing the workload, before attempting to distribute it to the other machines. By using the empirical study, we investigate in what level the GPU can give benefits to different types of online game servers. As a result, we can give suggestion how the GPU should be involved so that the performance of game services can be improved. We believe that this result can give benefits to online game developers who may want to gain performance of their applications without requiring any extra resources.

Keywords
Online Game, Graphic processor utilization, GPGPU

1. INTRODUCTION
Multiplayer online game has been drastically gaining in strength and popularity in the game industry. To date, it becomes common that an online game may be served for hundreds or even thousands of concurrent players at same time. In this regards, the response time of multiplayer online games is crucial, especially during the high load of massive players. Hence, in order to lighten the load, the popular approach is often done by upgrading the current game server machine with a higher performance one. While it is easy and straightforward, this approach demands a new machine to be invested without utilizing some local resources, such as Graphic Processor Unit (GPU) and its memory, that are already available in the game server machine.

In this paper, we argue to take the GPU, which has shown an unbelievable excellent performance in many kinds of application, into consideration so that online games' service quality in term of response time can be maintained before attempting to distribute the excessive workload to the other machines. To support this argument, we describe our empirically evaluation and demonstrate our GPU-utilized online games. The contribution of this paper is in twofold; a) we advocate the GPU can be a good companion with the central processing unit (CPU) located in a machine for reducing the workload or accelerating some task during the heavy load, and b) we suggest on the different levels of GPU support for giving benefits to different service type of online game servers.

The remainder of the paper is organized as follows. In Section 2, we give a brief overview the GPU architecture, the difference of GPU and CPU architecture, and the programming features of new generation GPU. In Section 3, we describe in what way GPU can be involved into the online game server. In Section 4, we describe our experiments and results, following with. a suggested guideline on improving the performance of online game service via the GPU utilization. In Section 5 we conclude the paper.

2. BACKGROUND
2.1 Graphic Processor Unit Architecture
Due to the insatiably demanded high-definition 3D graphics in most modern games, the GPU hardware architecture has evolved into an enormous compute-intensive, highly parallel multithreaded multi-core processor [1], as illustrated in Figure 1 in comparison of general multi-core Central Processing Unit (CPU). As consequence, the GPU is then well-suited to speed up many computational applications, where a large data set can be processed via the mapping of data elements to parallel processing threads. In this case, the GPU can be referred as General Purpose computing on Graphic Processor Unit (GPGPU) technology. Hence, it is clear that the modern GPU architecture can potentially give benefits to most operations of game server machines, in order to maintain its service quality.

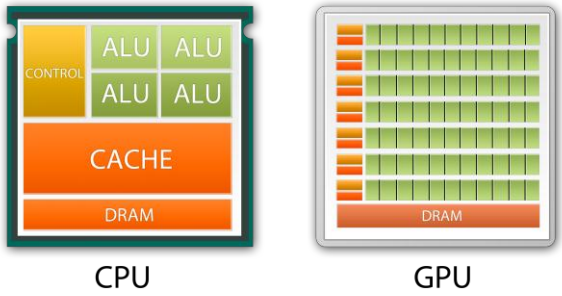


Figure 1. CPU and GPU Architecture.

2.2 Related Works
Many research works have been found on improving the service quality of online game servers during the times of congestion. However, we are particularly interested in workload distributed techniques of online game server. For instance, the works in [2, 3] suggest how CPU resources of many computers in federation can be work in cooperation for serving as a unified infrastructure for hosting on-line games based on grid technology. In [4], it takes the similar approach to design a flexible and

powerful infrastructure for Massive Multiplayer online (MMO) games, which cooperates among a number of servers working in the backend for different functionalities such as game server, Web application server, and database. Our work aims to achieve the same objective of CPU workload distribution, but focusing on the local GPU utilization for basic improvement, before attempting to distribute workload (i.e. game services) to the other machines of game servers. The concept can be easily seen in Figure 2.

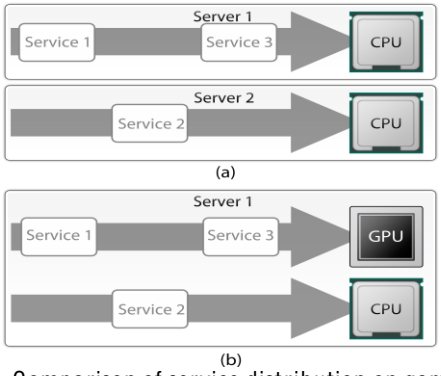


Figure 2. Comparison of service distribution on game server machine: (a) General approach, and (b) Our proposed one.

3. PROPOSED MODEL

As mentioned previously in the Section 2, the GPU is so powerful that it can be simply used as a general-purpose GPU (GPGPU) for accelerating large data-parallel computations, but needs to work in collaboration with the local CPU on the same machine. However, in the context of online game server applications, it is doubtful in what level the GPU can give benefits to them. To explore this, we setup an collaboration model for further investigation in the next section. In Figure 3, the process flows occurring in our model can be described as in follows: When a client makes a request in (1) and the server should perform some computation and return a response accordingly (6).

1. In case of sole computing on the local CPU, all the client related parameters will be locally performed and kept in the main memory.
2. In case of distributed computing on the GPU, all the client related parameters will be copied to the GPU's memory and further executed under the command of CPU by means of the service wrapper functionality. When finished, the results will be later transferred to store on the main memory.

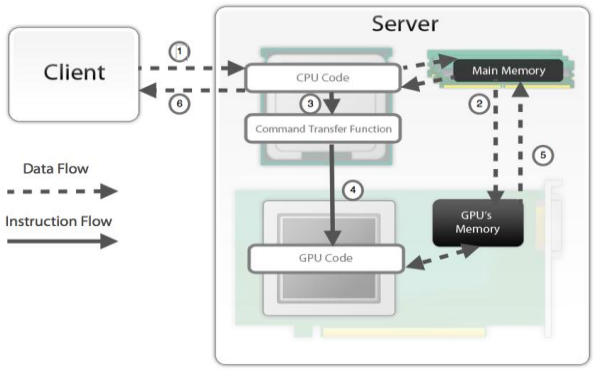


Figure 3. **GPU Function process**

4. EMPIRICAL EVALUATION

We group the propose solution's empirical evaluation into 2 parts. 1 testing the propose solution in one player online game server and 2 testing the propose solution in massive multiplayer online games server. In this case, both parts using same experiment platform consist of 2.66 GHz Quad-Core Intel Core i7 (4 Core 8 threads CPU), 3GB DDR3 main memory, Windows 7 x86 with NVIDIA ForceWare version 190.38 as GPU driver and NVIDIA GeForce 9800GX2 graphics cards (Consists 256 Stream processors)

For the purpose of testing, we build two versions of each experimental online game; the version running solely on the CPU (Built with Microsoft Visual C# 2008) and the other version running on both CPU and GPU (Built with NVIDIA CUDA SDK 2.2). We repeat five times for each testing and then average the results to yield the final result showed in this paper.

4.1 Experiment 1: Single Player Online Game

4.1.1 The model

We implement the Tic-tac-toe [10] program, where two players (one character is human and the other is computer (called a non-player character or NPC)) compete for placing three respective marks in a horizontal, vertical or diagonal row to win the game, as illustrated in Figure 4. However, two key services are particularly concerned in our developed program; a) AI-based service for computing NPC's next turn and b) Score calculation service for accumulating the user scores. It's noted that the AI-based algorithm can take direct advantage from the co-existed GPU by distributing the workload searching for the best strategy across GPU's parallel cores. This is not the case for the score calculation, where a single calculated operation is only processed.

In our experiments, either of these services will take different patterns for execution on both types of processors. The objectives are to observe in which context (or in what level) the GPU can give benefits to the execution of Tic-tac-toe game servers.

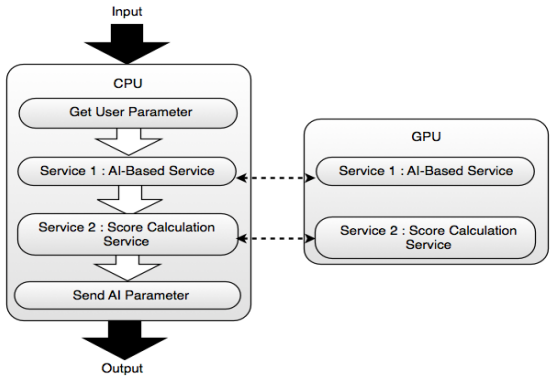


Figure 4. **Workflow in Tic Tac Toe online game**

4.1.2 Results

As we can see from the Table 1, the AI-based service running on the GPU noticeably outperforms (in term of producing overall response time) this service running on the CPU, while the Score calculation service is better running on the CPU, rather the GPU counterpart. This implies the shift of separable workloads to be executed on the GPU can potentially reduce the processing time.

Table 1. Overall response time of single thread (Tic-tac-toe)

Test Case Number	AI-based Service running on	Score Calc. Service running on	Elapsed Time (ms)
1	CPU	CPU	1.209
2	GPU	CPU	0.046
3	CPU	GPU	1.237
4	GPU	GPU	0.074

However, to ensure the better performance of overall system, we repeat the above scenarios, but increasing the number of inputs to 100 and 1000 clients. As the result shown in Figure 5, the overall CPU usage time in the Test Case number 2 is relatively shorter than that of the Test Case number 1.

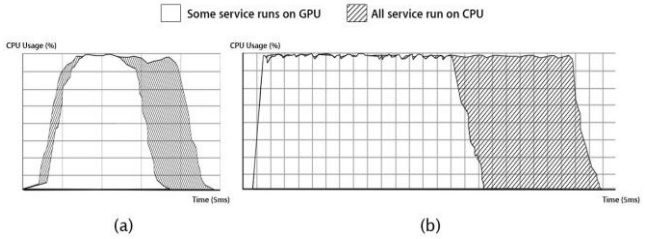


Figure 5. **Overall CPU loads comparison in 2 scenarios: (a) 100 workloads and (b) 1000 workloads**

4.2 Massive Player Online Game

4.2.1 The model

Here, we are interested in massive multiplayer online game server (like Final Fantasy XI Online, Ragnarok Online), but decide on the implementation of a massively Muliplayer online role-playing game (MMORPG) server called the virtual online stock exchange games. It is a kind of game that simulates the stock exchange environment as illustrated in Figure 6, and three key services are mainly concerned in our developed program; a) Score ranking service for sorting all users' scores in the game server, b) Trader searching service for locating the stock trader in the database, and c) User status update service for updating all users' sores in the game server. Here, only the user status update service unlikely obtains the benefits from the co-existed GPU since a single calculated operation is only processed upon a request for a user.

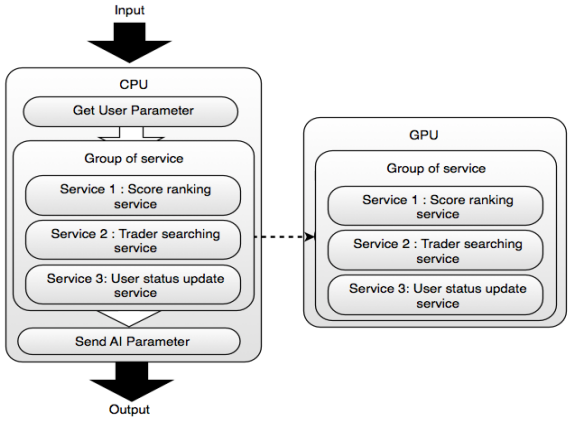
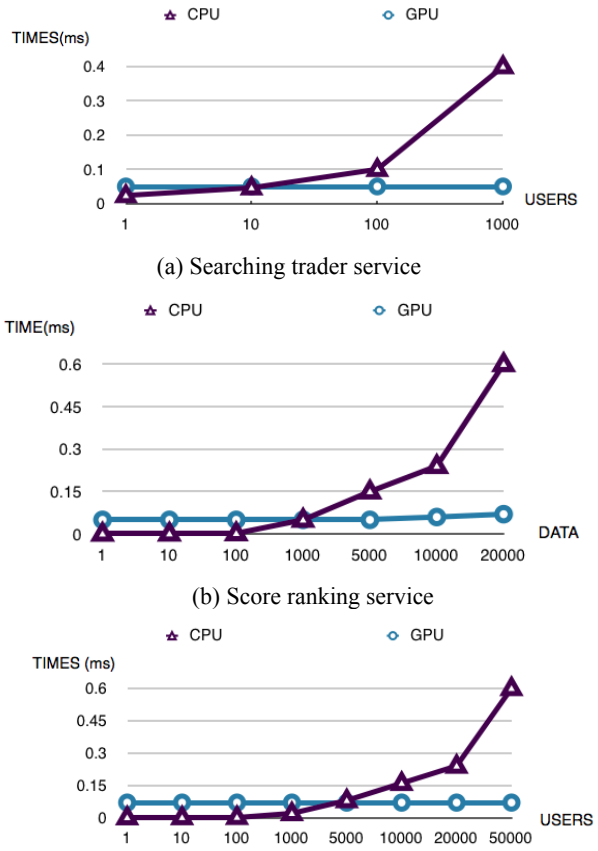


Figure 6. **Workflow in Virtual online stock exchange game**

Here, each service will be individually performed by varying the number of input from 1 to 20000 clients and overall response time will be observed.

4.2.2 Results

As seen in Figure 7, with a significantly large number of clients (approximately 2000 upward), the GPU can give the extreme benefits for all separable tasks such found in the searching trader service and the score ranking service. However, it is not the case when there is a small number of user data executed on the GPU due to the burden of associated overheads.



(a) Searching trader service

(b) Score ranking service

(c) User status update service

Figure 7. Overall **response time** on various services

5. CONCLUSION

We have presented a possible approach to lighten CPU workloads of online game server application by favorable means of utilizing the local GPU computing facility. The objective is to express that the GPU can be a helpful factor for prolonging the fast response time to massive game players as long as the workload of game server machine is not overloaded. However, as clearly seen from our experimental results, it is not always the case that all sorts of online game will gain benefits owing to the cooperative works of these two processors. We then give suggestion on what sort of workload should be effectively executed on the GPU, therefore giving an extreme benefit not only to online game developers, but also the overall online game industry.

6. REFERENCES

[1] NVIDIA Corp 2009. NVIDIA CUDA Programming Guide 2.2 (2009)

[2] D Saha, S Sahu and A Shaikh 2003. A service platform for on-line games. In Proceedings of Workshop on Network and System Support for Games (2003)

[3] A Shaikh, S Sahu, M Rosu, M Shea and D Saha 2003. Implementation of a Service Platform for Online Games. Proceedings of 3rd ACM SIGCOMM workshop (2003)

[4] Hyun Sung Chu 2008. Building a simple yet powerful MMO game architecture (2008)

[5] S Caltagirone, M Keys, B Schlief and M Willshire 2002. Architecture for a massively multiplayer online role playing game engine. Journal of Computing Sciences in Colleges (2002)

[6] J Owens, D Luebke, N Govindaraju and M Harris 2007. Computer Graphics Forum (2007)

[7] J Glenn-Anderson 2009. GPU-based Desktop Supercomputing (2009)

[8] Gao Huang, Meng Ye and Long Cheng 2004. Modeling System Performance in MMORPG. IEEE Communications Society Globecom 2004 Workshops (2004)

[9] B Cowley, D Charles, M Black and R Hickey 2008. Toward an Understanding of Flow in Video Games. Computers in Entertainment (CIE) (2008)

[10] Tic-tac-toe, Wikipedia, the free encyclopedia (2010), http://en.wikipedia.org/wiki/Tic-tac-toe

The 6th National Conference on Computing and Information Technology, (NCCIT 2010)

สถาปัตยกรรมเชิงบริการสำหรับแม่ข่ายเกมที่ใช้ประโยชน์หน่วยประมวลผลกราฟิก
Service-oriented architecture for graphic processor enabled game server

ฤทธิชัย จิตภักดีบดินทร์(Rittichai Jitpukdeebodintra)[1] และ สุนทร วิทูสุรพจน์ (Suntorn Witosurapot)[2]

[1 2] ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

rittichai@capsuledna.com , wsuntorn@coe.psu.ac.th

บทคัดย่อ

ธุรกิจบริการเกมออนไลน์ขนาดใหญ่นิยมเพิ่มจำ นวนเครื่องแม่ข่ายเพื่อรักษาให้ประสิทธิภาพของบริการอยู่ในระดับที่ยอมรับได้แม่ว่าจะมีงานวิจัยที่แนะนำให้ลดค่าใช้จ่ายนี้โดยการใช้แนวทางเชิงสถาปัตยกรรมบริการเพื่อกระจายภาระงานไปประมวลผลยังเครื่องแม่ข่ายอื่นๆอย่างไรก็ตามในบท ความนี้เห็นแย้งให้เริ่มจากการนำหน่วยประมวลผลทางกราฟิก (GPU) ซึ่งมีอยู่แล้วในเครื่องแม่ข่ายทั่วไปเพื่อประสานการทำ งานร่วมกับหน่วยประมวลผลหลักในเบื้องต้นก่อนการกระจายภาระงานไปยังเครื่องอื่นๆต่อไปซึ่งจะทำให้ได้รับประสิทธิภาพสูงสุดเมื่อเปรียบเทียบกับวิธีการพื้นฐานอื่นๆ โดยมีผลสนับสนุนจากการทดสอบเบื้องต้นของระบบที่ได้นำเสนอไว้

คำสำคัญ: เกมออนไลน์ สถาปัตยกรรมเชิงบริการ การใช้ประโยชน์หน่วยประมวลผลกราฟิก

Abstract

A large-scale of online games usually invests the additional hardware facility, in order to maintain the acceptable service quality. Although there exists a research work that aims to decrease this additional hardware cost by adopting the service-oriented architecture and distributing the load to be executed on the other machines. In this paper, we argue that the Graphic processor (GPU) located already in the typical machines can be useful for boosting the game service's performance, particularly when it is worked in conjunction with the Central processor (CPU). We further show the evidence of our experimental prototype given in the paper.

Keywords: Online Game, Service Oriented Architecture, Graphic processor utilization

1. บทนำ

การให้บริการเกมออนไลน์ขนาดใหญ่นั้นจำเป็นต้องมีการลงทุนทั้งในด้าน โครงข่ายในการเชื่อมต่อ และ กลุ่มเครื่องแม่ข่ายที่ให้บริการ เป็นจำนวนมาก ประกอบกับวิธีการที่ผู้ให้บริการเกม ส่วนใหญ่มักจะใช้นั้น คือการเตรียมทรัพยากรเหล่านั้นสำหรับเกมแต่ล่ะเกมซึ่งนั้นทำให้เกิดความเสี่ยงต่อการใช้ทรัพยากรอย่างไม่คุ้มค่า ปัญหาดังกล่าวถูกแก้ไขได้ในระดับหนึ่งโดยการนำเอารูปแบบสถาปัตยกรรมบริการ (Service Oriented Architecture : SOA) มาใช้โดยการจัดกลุ่มบริการของเกมออนไลน์ ดังเช่นในงานวิจัย [1] [2] อย่างไรก็ตาม มวิธีการดังกล่าวยังขาดการใช้งานหน่วยประมวลผลอีกชนิดนอกเหนือจาก CPU (หน่วยประมวลผลกลาง) คือ GPU (หน่วยประมวลผลกราฟิก) ที่มีอยู่ในเครื่องคอมพิวเตอร์ในปัจจุบันทุกเครื่อง มีประสิทธิภาพสูงในการช่วยการลดภาระงาน และเพิ่มประสิทธิภาพในการให้บริการเกมออนไลน์

ในบทความนี้ได้นำเสนอรายละเอียดของกลไลผสานงานของบริการระหว่าง CPU และ GPU ในเครื่องแม่ข่ายเกมออนไลน์โดยใช้ เทคโนโลยีการพัฒนาโปรแกรมบนหน่วยประมวลผลกราฟิก (GPGPU) และ มาตรฐาน SOA ที่ใช้อย่างกว้างขวางในปัจจุบันโดยคาดหวังถึง ความยืดหยุ่นในการกระจายบริการ การใช้ทรัพยากรของระบบอย่างเต็มประสิทธิภาพ และ ประสิทธิภาพที่จะเพิ่มขึ้น

ส่วนที่เหลือของบทความได้ถูกวางให้แสดงดังต่อไปนี้ ในตอนที่ 2 จะเป็นการแนะนำความรู้เบื้องต้นที่เกี่ยวข้องกับงานวิจัยนี้ ตอนที่ 3 เป็นการอธิบายวิธีการทำงานของโครงสร้างที่นำเสนอ ตอนที่ 4 เป็นการเสนอผลทดสอบเบื้องต้นผ่านเกมออนไลน์ตัวอย่างที่ประยุกต์ใช้งานวิธีการดังกล่าว บทที่ 4 จะแนะนำงานที่เกี่ยวข้อง และสรุปผลในตอนที่ 5

2. ความรู้เบื้องต้น

2.1 Service-oriented architecture (SOA)

SOA เป็นรูปแบบในการพัฒนาซอฟท์แวร์โดยที่มุ่งเน้นไปที่การให้บริการโดยที่จะมองส่วนประกอบของโปรแกรมนั้นออกเป็นการให้บริการ (Software as a service : SaaS) SOA ถูกออกแบบมาให้ผู้พัฒนานั้นสามารถแลกเปลี่ยนข้อมูลกันระหว่างแอพลิเคชั่นที่แตกต่างกันได้ โดยที่ไม่จำกัดภาษาในการออกแบบโปรแกรมในลักษณะ SOA นี้จะต้องลดเงื่อนไขและข้อกำหนดของโปรแกรมให้น้อยที่สุดเพื่อที่จะทำให้ แอพลิเคชั่นสามารถเรียกใช้จากแพลตฟอร์ม หรือ ระบบปฏิบัติการใดๆได้ [4]

ในงานวิจัยนี้ได้นำเอาข้อดีของ SOA มาใช้ในการลดความซ้ำซ้อนของบริการในการพัฒนาโปรแกรมแม่ข่ายเกมออนไลน์ การต่อขยายโปรแกรมแม่ข่ายเกม และเพิ่มประสิทธิภาพในการใช้งานทรัพยากรในเครื่องแม่ข่ายเกมออนไลน์ ผ่านการใช้งานมาตร ฐาน Web Services เช่น SOAP (Simple Object Access Protocol : โปรโตคอลสำหรับส่งข้อมูลระหว่าง Service) และ WSDL (Web Services Description Language : ภาษาที่ใช้ระบุบริการของ Web services) ซึ่งเป็นมาตรฐาน ที่ใช้อย่างกว้างขวางในการพัฒนาโปรแกรมที่มีโครงสร้างแบบ SOA

2.2 General-purpose computing on graphics processing units (GPGPU)

GPGPU (บางครั้งเรียกว่า GPGP หรือ GP²) เป็นเทคนิคในการพัฒนาโปรแกรมโดยการใช้ GPU (Graphic Processor Unit) ซึ่งในปกตินั้นจะใช้ในการประมวลผลส่วนการแสดงผลของโปรแกรมคอมพิวเตอร์ มาใช้ในการประมวลผลโปรแกรมทั่วๆไป เช่น การคำนวณค่าทางคณิตศาสตร์ต่างๆ , การประมวลผลฐานข้อมูล หรืองานอื่น ๆ เช่นเดียวกันกับงานที่ประมวลผลบน CPU (Central Processing Unit) ในปัจจุบัน

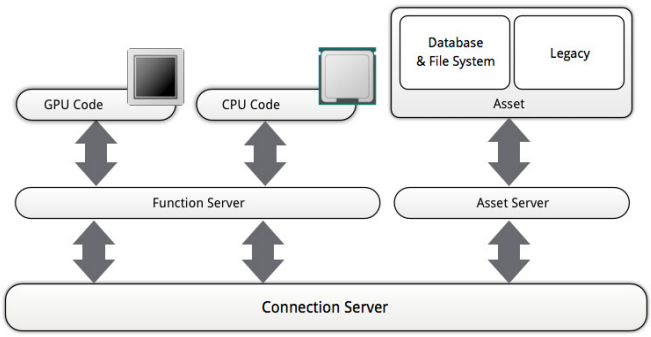
ในบทความนี้ได้มีการนำเอาประสิทธิภาพของ GPU ที่สามารถช่วยการลดภาระงาน และเพิ่มประสิทธิภาพในการให้บริการเกมออนไลน์ดังที่แสดงในงานวิจัย [3] ผ่านเทคโนโลยี GPGPU ที่ใช้อย่างกว้างขวางในปัจจุบัน

2.3 งานวิจัยที่เกี่ยวข้อง

มีงานวิจัยจำนวนมากที่มุ่งปรับปรุงประสิทธิภาพของโปรแกรมแม่ข่ายเกมออนไลน์ที่บทความนี้สนใจอย่างมากคือในการนำเอา SOA มาใช้ในการกระจายภาระงานการให้บริการเกมออนไลน์เพิ่มเพิ่มประสิทธิภาพการใช้งานทรัพยากรของระบบให้เต็มความสามารถมากยิ่งขึ้น เช่น งานวิจัย [1] ที่ใช้วิธีการรวมกลุ่มบริการของเกมออนไลน์เพื่อที่เพิ่มประสิทธิภาพในการใช้งานทรัพยากรระบบ หรืองานวิจัย [5] ที่แสดงให้เห็นถึงการใช้ SOA เข้ามาใช้ในการขายประสิทธิภาพของระบบซึ่งงาน ทั้ง 2 นั้นแตกต่างจากงานวิจัยในบทความนี้ตรงที่ งานวิจัยนี้ได้เพิ่มประสิทธิภาพในการเข้าใช้ทรัพยากรของระบบให้มากขึ้นกว่าเดิมโดยเพิ่ม GPU ซึ่งเป็นหน่วยประมวลผลที่มีอยู่ในคอมพิวเตอร์ทุกเครื่องอยู่แล้วและมีประสิทธิภาพสูงเข้ามาเพิ่มประสิทธิภาพในการให้บริการเกมออนไลน์

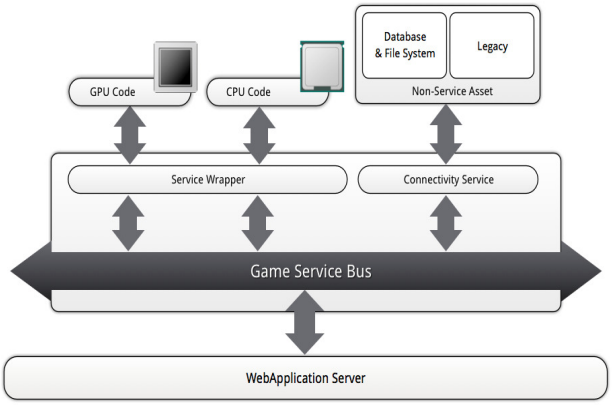
3. โครงสร้างที่นำเสนอ

จากงานวิจัย [3] ที่ไม่ได้มีการใช้ SOA ในการออกแบบสถาปัตยกรรมเกมออนไลน์ ดังแสดงในภาพที่ 1



ภาพที่ 1: โครงสร้างแม่ข่ายเกมออนไลน์ในงานวิจัย [3]

ทำให้โครงสร้างดังกล่าวขาดความสะดวกในการกระจายบริการไปยังหน่วยประมวลผลต่างๆอันเนื่องมาจากภาษาที่ ใช้พัฒนานั้นไม่เหมือนกัน การขยายบริการไปสู่ทรัพยากรเครื่องอื่นๆในระบบยังไม่สามารถทำได้โดยง่าย และ ปัญหาความซ้ำซ้อนของบริการในการพัฒนาโปรแกรม ดังนั้นบทความนี้จึงได้เสนอโครงสร้างเกมออนไลน์ ดังแสดงในภาพที่ 2

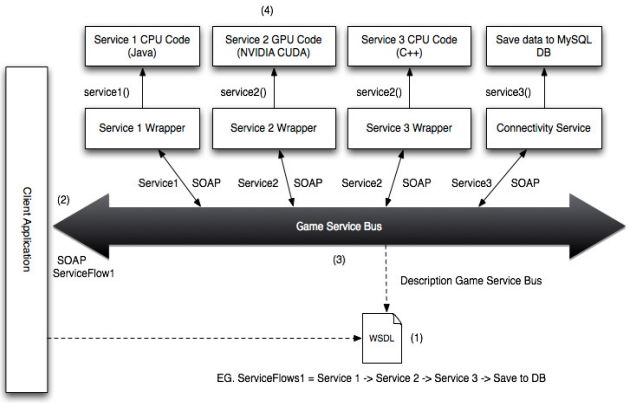


ภาพที่ 2: โครงสร้างแม่ข่ายเกมออนไลน์ที่ได้ออกแบบ

โครงสร้างที่นำเสนประกอบไปด้วย 4 ส่วนหลักดังภาพที่ 2

1) Game Service Bus (GSB) ทำหน้าที่เป็นตัวประสาน งานบริการทั้งหมดของระบบเป็นตัวกลางในการสั่งการทำงานของบริการต่างๆ (Service exploits) และแลกเปลี่ยนสถานะในแต่ละบริการ (Messaging middleware) GSB จัดรูปแบบและเส้นทางการทำงานบริการ GSB จะทำการติดต่อกับฟังก์ชั่น Service Wrapper ผ่านทาง SOAP ในสถาปัตยกรรมที่ได้ออกแบบนี้ถือได้ว่า GSB มีบทบาทมากที่สุดในการสั่งการทำงานของ บริการต่างๆให้ทำงานบน CPU หรือ GPU เนื่องจาก GSB จะมีข้อมูลการทำงาน ของทุกบริการซึ่งนั้นทำให้ GSB รู้ว่ามีภาระงานมากแค่ ไหนในระบบนั้นเอง
2) Service Wrapper ทำหน้าที่เป็นตัวเชื่อมต่อระหว่าง CPU Code , GPU Code กับ GSB, Service Wrapper จะทำหน้าที่ในการสั่งคำสั่งที่ได้รับจาก GSB มาทำการสั่ง การ CPU Code และ GPU Code
3) CPU Code จะทำหน้าที่สั่งการประมวลผลภาระงาน ต่างๆ บนหน่วยประมวลผลกลาง
4) GPU Code จะทำหน้าที่สั่งการประมวลผลภาระงาน ต่างๆ บนหน่วยประมวลผลกราฟิก

ภาพที่ 3 แสดงให้เห็นถึงขั้นตอนการทำงานที่ เครื่องลูกข่ายจะรับรู้ว่าการทำงานของบริการต่างๆจะต้องการ Input ใดผ่านทาง WSDL (1) และส่งinput ผ่านSOAP (2) เมื่อเครื่องแม่ข่ายรับ input แล้ว GSB จะทำการจัดเรียงลำดับของบริการ (ในตัวอย่างคือ Service1 , Service2 , Service3 และ Save to DB เรียงต่อกัน) จากนั้นจึงส่ง input ที่บริการนั้นๆต้องการผ่านทาง SOAP (3) และด้วยลักษณะเฉพาะของการพัฒนาโปรแกรมโดยใช้สถาปัตยกรรม SOA ทำให้ไม่ว่าบริการนั้นๆจะถูกพัฒนาด้วยภาษาใดๆก็สามารถทำงานร่วมกันได้



ภาพที่ 3: ขั้นตอนการทำงานของโครงสร้างที่ได้ออกแบบ

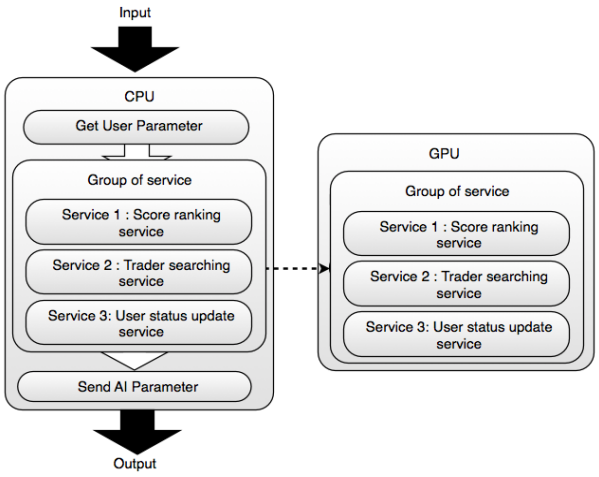
4. การทดสอบประสิทธิภาพ

4.1 เกมตัวอย่างที่ใช้ในการทดสอบ

เราได้ทำการทดสอบประสิทธิภาพของวิธีการที่นำเสนอในบทความนี้ผ่าน เกมออนไลน์ตัวอย่าง คือ โปรแกรมแม่ ข่ายเกมจำลองการซื้อขายหุ้นในตลาดหลักทรัพย์ (Virtual online stock exchange games) โดยโปรแกรม นี้จะประกอบไปด้วย 3 บริการหลักที่จะทำงานต่อเมื่อ มีผู้เล่นเรียกใช้ดังต่อไปนี้ : 1 บริการในการจัดเรียงคะแนน ของผู้เล่นทั้งหมดในเกมเพื่อจัด

อันดับคะแนน 2 บริการในการค้นหารายชื่อหุ้นที่มีขายอยู่ในเกม และ 3 บริการเพิ่มคะแนนไปยังผู้เล่น ทุกคนที่อยู่ในเกม แต่ละบริการที่ทดสอบนั้นเราได้จำลอง ผู้ใช้งานระหว่าง 1 ถึง 20,000 คนเพื่อเก็บค่าเวลาในการ ตอบรับโดยรวมของระบบ โดยทำการทดสอบเปรียบเทียบ กันกับวิธีการที่ไม่ได้มีการเพิ่มกลไกที่นำเสนอลงไป กับ วิธีการที่นำเสนอ

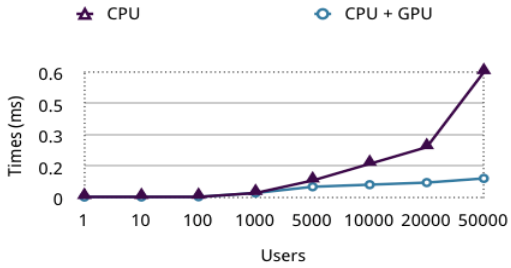
ในการทดสอบเราใช้เครื่องคอมพิวเตอร์ที่มีคุณสมบัติดังต่อไปนี้ในการทดสอบ : 2.66 GHz Quad-Core Intel Core i7 (4 Core 8 threads CPU) , 3GB DDRII main memory , Windows 7 x86 , NVIDIA GeForce 9800GX2 graphics cards การทดสอบ ทุกขั้นตอนจะทำการทดสอบทั้งหมด 5 ครั้ง และทำการหาค่าเฉลี่ยเป็นผลการทดสอบ



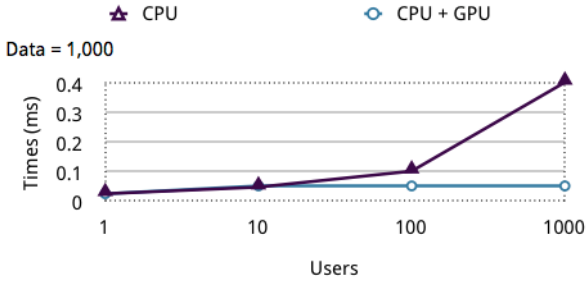
ภาพที่ 4: ขั้นตอนการทำงานของเกมออนไลน์ตัวอย่าง

4.2 ผลการทดสอบ

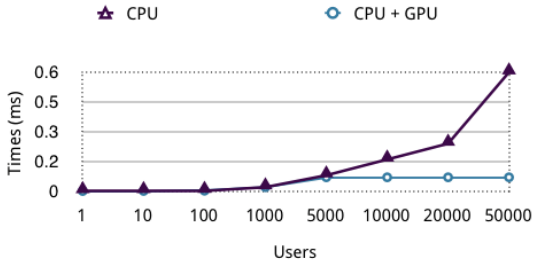
ภาพที่ 5 ได้แสดงให้เห็นถึงประสิทธิภาพในการตอบสนองของโปรแกรมแม่ข่ายเกมออนไลน์ที่เพิ่มขึ้นโดยเฉพาะอย่างยิ่งในสภาวะที่มีผู้ใช้งานพร้อมกันจำนวนมาก (มากกว่า 1,000 คนขึ้นไป) โดยสามารถเพิ่มความเร็วในการให้บริการได้สูงสุดถึง 6 เท่า (ดังแสดงในภาพที่ 5 (a))



(a) บริการจัดเรียงคะแนน



(b) บริการเพิ่มคะแนนไปยังผู้เล่น



(c) บริการในการค้นหารายชื่อหุ้น

ภาพที่ 5 ผลการทดลองประสิทธิภาพด้านเวลาตอบสนองของเกมออนไลน์ตัวอย่าง

6. สรุป

ในบทความนี้เราได้นำเสนอรายละเอียดของกลไกผสานงานของบริการระหว่าง CPU และ GPU ในเกมออนไลน์ ส่วนแม่ข่าย ซึ่งผลจากการทดสอบได้แสดงให้เห็นถึงประสิทธิภาพที่เพิ่มขึ้นจากการประยุกต์โครงสร้างนี้กับเกมออนไลน์ตัวอย่างได้อย่างมีนัยยะสำคัญ

7. เอกสารอ้างอิง

[1] D Saha, S Sahu and A Shaikh 2003. A service platform for on-line games. *Proceedings of Workshop on Network and System Support for Games*, 2003.

[2] A Shaikh, S Sahu, M Rosu, M Shea and D Saha 2003. Implementation of a Service Platform for Online Games. *Proceedings of 3rd ACM SIGCOMM workshop*, 2003.

[3] R Jitpukdeebodintra and S Witosurapot 2010, Improving Performance of Online Game Services via Graphic Processor: An Empirical Investigation , *Computer Games, Multimedia and Allied Technology (CGAT 2010) Proceedings* , 2010.

[4] Service-Oriented Architecture, Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Service-oriented_architecture , 2010.

[5] Hyun Sung Chu 2008. Building a simple yet powerful MMO game architecture, 2008.

[6] S Caltagirone, M Keys, B Schlief and M Willshire 2002. Architecture for a massively multiplayer online role playing game engine. *Journal of Computing Sciences in Colleges*, 2002.

[7] J Owens, D Luebke, N Govindaraju and M Harris 2007. *Computer Graphics Forum* ,2007.

[8] J Glenn-Anderson 2009. GPU-based Desktop Supercomputing, 2009.

ประวัติผู้เขียน

ชื่อ สกุล นายฤทธิชัย จิตภักดีบดินทร์

รหัสประจำตัวนักศึกษา 5110120085

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมกรรมศาสตร์บัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2551

การตีพิมพ์เผยแพร่ผลงาน

1. Rittichai Jitpukdeebodintra and Suntorn Witosurapot, Improving Performance of Online Game Services via Graphic Processor: An Empirical Investigation, 3rd Annual International Conference on Computer Games, Multimedia and Allied Technology (CGAT 2010), (2010)
2. ฤทธิชัย จิตภักดีบดินทร์ และ สุนทร วิทูสุรพจน์. 2553. สถาปัตยกรรมเชิงบริการสำหรับแม่ข่ายเกมส์ที่ใช้ประโยชน์หน่วยประมวลผลกราฟิก. การประชุมวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีครั้งที่ 6 (NCCIT 2010), (2010)