ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

ปริญญา

วิศวกรรมคอมพิวเตอร์	วิศวกรรมคอมพิวเตอร์
สาขา	**ภาควิชา**

เรื่อง การพัฒนาเทคนิคการประมวลผลการซื้อขายหลักทรัพย์สมรรถนะสูงด้วยหน่วยประมวลผลแบบจีพียู

The Development of High-Performance Security Trading Techniques Using GPU

นามผู้วิจัย นางสาวเกษรินทร์ รุ่งเรือง

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(รองศาสตราจารย์อานนท์ รุ่งสว่าง, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์อนันต์ ผลเพิ่ม, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การพัฒนาเทคนิคการประมวลผลการซื้อขายหลักทรัพย์สมรรถนะสูงด้วยหน่วยประมวลผลแบบจีพียู

The Development of High-Performance Security Trading Techniques Using GPU

โดย

นางสาวเกษรินทร์ รุ่งเรือง

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

พ.ศ. 2557

เกษรินทร์ รุ่งเรือง 2557: การพัฒนาเทคนิคการประมวลผลการซื้อขายหลักทรัพย์สมรรถนะสูงด้วยหน่วยประมวลผลแบบจีพียู ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D. 45 หน้า

การประมวลผลด้านการเงินโดยเฉพาะการจับคู่การซื้อขายหลักทรัพย์โดยอัลกอริธึมแบบ AOM ที่ใช้กันงานอย่างกว้างขวางนั้น เป็นงานที่ต้องจัดการกับข้อมูลด้วยความเร็วสูง ทำให้หลายองค์กรที่เกี่ยวข้องกับการเงิน มีความต้องการวิธีการที่จะเร่งสมรรถนะการคำนวณดังกล่าว

วิทยานิพนธ์นี้ ได้นำเสนอแนวคิดของการนำระบบประมวลผลแบบ จีพียู มาช่วยเร่งการประมวลผล ในการทำงานของการจับคู่ซื้อขายหลักทรัพย์แบบ AOM โปรแกรมที่พัฒนาขึ้นสามารถเร่งกระบวนการทำงานให้เร็วขึ้นได้ แต่ก็มีขีดจำกัดอยู่บ้างเนื่องจากการส่งผ่านข้อมูลจากหน่วยความจำไปยังระบบจีพียู ผลจากการพัฒนาทำให้เกิดแนวทางที่สามารถนำระบบประมวลผลจีพียูที่ใช้กันแพร่หลายและราคาถูกมาใช้เร่งความเร็วในการจับคู่ซื้อขายหลักทรัพย์ได้

_____ _____ ___/___/___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Ketsarin Rungraung 2014: The Development of High-Performance Security Trading Techniques Using GPU. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Assistant Professor Putchong Uthayophas, Ph.D. 45 pages.

Financial computation, especially the security trading and matching based on widely used AOM algorithm, must perform a high speed data processing. Therefore, many financial organizations need a way to accelerate this computation.

In this thesis, the use of GPU is proposed as a way to speed up the AOM stock trading algorithm. The software and algorithm developed result in a faster execution time. Nevertheless, there is still some limitation due to the need to move data from main menory to GPU system. This work presents a guideline that enable users to use cost effective GPU system to solve the AOM stock trading problem.

_____ _____ ___/___/___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ขอขอบคุณผู้ช่วยศาสตราจารย์ ดร.ภุชงค์ อุทโยภาศ อาจารย์ที่ปรึกษาโครงการของข้าพเจ้าในระดับปริญญาโท ผู้ให้แนวคิดในการพัฒนาโครงการและการสนับสนุนโอกาสในทุกๆ ด้าน ขอขอบพระคุณอาจารย์อานนท์ รุ่งสว่าง อาจารย์ที่ปรึกษาร่วมโครงการของข้าพเจ้า ผู้ให้คำปรึกษาและความช่วยเหลือหลายๆ ด้าน รวมไปถึงอาจารย์ในภาควิชาทุกๆ ท่านด้วย

ขอขอบคุณสมาชิกห้องปฏิบัติการณ์ HPCNC ทุกท่านที่ให้คำแนะนำ และคำปรึกษาต่างๆ โดยเฉพาะคุณธนาวุฒิ และคุณนววิธ ที่ให้ความช่วยเหลือเสมอมา พร้อมทั้งขอขอบคุณพี่ๆน้องๆ และเพื่อนๆในรุ่นทุกท่านที่ได้ร่วมศึกษาด้วยกัน

ท้ายที่สุด ข้าพเจ้าขอกราบขอบพระคุณพ่อและคุณแม่ ที่ให้การสนับสนุนในทุกด้าน และเป็นกำลังใจให้กับข้าพเจ้าเสมอมา ความสำเร็จทางการศึกษาที่ได้รับวันนี้แม้จะมีค่าแต่ก็คงไม่เพียงพอจะตอบแทนบุญคุณท่านทั้งสองได้ หากหวังว่าจะได้ตอบแทนด้วยการดูแลท่านทั้งสองต่อจากนี้ไป

เกษรินทร์ รุ่งเรือง

มิถุนายน 2557

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

ภาพที่ **หน้า**

การพัฒนาเทคนิคการประมวลผลการซื้อขายหลักทรัพย์สมรรถนะสูงด้วยหน่วยประมวลผลแบบจีพียู

The Development of High-Performance Security Trading Techniques Using GPU

คำนำ

ข้อมูลทางการเงินเป็นข้อมูลขนาดใหญ่ และมีแนวโน้มที่จะใหญ่ขึ้นเรื่อยๆ ปัจจุบันมีการพัฒนาเทคโนโลยีที่ใช้ในการประมวลผลข้อมูลขนาดใหญ่ด้วยวิธีการแบบใหม่ๆ ที่ไม่จำกัดเพียงฐานข้อมูลหลักเท่านั้น มีหลายองค์กรหันมาสนพัฒนาเทคโนโลยี และหาแนวทางการปฏิบัติใหม่ๆ เพื่อสามารถ รวบรวม ประมวลผล ค้นหา และจัดเก็บข้อมูลขนาดใหญ่ทั้งแบบที่มีโครงสร้าง และไม่มีโครงสร้างได้อย่างรวดเร็ว และมีประสิทธิภาพ

การประมวลผลเป็นหนึ่งในขั้นตอนสำคัญของการจัดการข้อมูลขนาดใหญ่ ระบบจะใช้การทำงานแบบคู่ขนานที่มีประสิทธิภาพสูงเพื่อประมวลผลข้อมูลในแต่ละจุดอย่างรวดเร็ว ต่อจากนั้นในแต่ละจุดจะได้รับข้อมูลในรูปแบบของชุดข้อมูลที่สามารถนำไปใช้ดำเนินการได้อย่างรวดเร็ว ซึ่งมีนักวิจัยสนใจศึกษาการทำงานแบบขนานด้วยวิธีต่างๆ เพื่อช่วยเพิ่มความสามารถในการประมวลผลแบบเดิม

การประมวลผลบนหน่วยประมวลผลกราฟิกเป็นอีกวิธีที่มีการประมวลผลแบบขนาน จากขีดความสามารถที่เพิ่มขึ้นของหน่วยประมวลผลกราฟิก ทำให้สามารถทำการประมวลผลงานทั่วไปที่นอกเหนือจากงานกราฟิกอย่างเดียว

งานวิจัยนี้เสนอการนำหน่วยประมวลผลกราฟิกมาช่วยเร่งความเร็วในการประมวลผลอัลกอริทึมในงานด้านการเงิน โดยทำการศึกษาการประมวลผลอัลกอริทึมการซื้อขายหลักทรัพย์ภายใต้สถาปัตยกรรมคู่ค้า เพื่อทำการวิเคราะห์สมรรถนะการประมวลผลระหว่างการประมวลผลบนหน่วยประมวลผลกลางอย่างเดียว และการนำหน่วยประมวลผลกราฟิกมาช่วยในการประมวลผล พร้อมศึกษาปัจจัยที่มีผลกระทบต่อสมรรถนะของการประมวลผล

วัตถุประสงค์

1. เสนอการนำหน่วยประมวลผลกราฟิกมาช่วยเร่งความเร็วการคำนวณอัลกอริทึมทางด้านการเงิน

2. นำหน่วยประมวลผลกราฟิก (GPU) มาใช้ในการคำนวณงานด้านการเงิน โดยเร่งความเร็วการประมวลผลการจับคู่คำสั่งซื้อขายหุ้นด้วยวิธี AOM

3. ประเมินผลการทดสอบสมรรถนะด้านความเร็วของการคำนวณอัลกอริทึมทางด้านธุรกิจ ระหว่างการหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิก

การตรวจเอกสาร

ในหัวข้อนี้จะกล่าวถึงความรู้พื้นฐานและงานวิจัยก่อนหน้าที่เกี่ยวข้อง การประมวลผลทั่วไปบนหน่วยประมวลผลกราฟิก (GPGPU) สถาปัตยกรรมคูด้า วิธีการซื้อขายหลักทรัพย์ และงานที่เกี่ยวข้อง

การประมวลผลทั่วไปบนหน่วยประมวลผลกราฟิก (GPGPU)

GPGPU ย่อมาจาก General-Purpose computation on Graphics Processing Units (Halfhill, 2008; Cascarano, Rolando, Risso และ Sisto, 2010; Mohan และ Cavazos, 2010) คือ การประมวลผลทั่วไปบนหน่วยประมวลผลกราฟิก ซึ่งการประมวลผลดังกล่าวมีประสิทธิภาพสูงมาก เนื่องจากหน่วยประมวลผลกราฟิกมีจำนวนคอร์โปรเซสเซอร์อยู่เป็นจำนวนมาก ทำให้มีความสามารถในการคำนวณสูง ภาษาที่ใช้ในการพัฒนาเป็นภาษามาตรฐาน คือ C/C++ เพื่อให้เป็นภาษามาตรฐานสำหรับนักพัฒนาซอฟต์แวร์

GPGPU เป็นการใช้เทคโนโลยี ที่มีอยู่แล้วในกราฟิกการ์ด ในส่วนของหน่วยประมวลผลกราฟิกที่เราเรียกว่า shader มาใช้ในการประมวลผลข้อมูล ที่มีลักษณะเป็นกลุ่มของข้อมูล ที่ต้องการคำสั่งในการทำงานเดียวกัน (SIMD) ซึ่งสามารถช่วยเร่งความเร็วในการทำงานได้มากกว่าการใช้ CPU ประมวลผล แต่ไม่ใช่ว่าข้อมูลต่างๆสามารถประมวลบน GPU แล้วเร็วทั้งหมด เพราะความเร็วนั้นยังขึ้นอยู่กับการเลือกกลุ่มข้อมูล อัลกอริธึม ที่ใช้ในการประมวลคำสั่งอีก

ในปัจจุบันก็มีการพัฒนาเครื่องมือต่างๆ มากมายที่ช่วยในการดึงประสิทธิภาพ GPU มาใช้งานในการประมวลผลโดยทั่วไปที่ไม่ใช่เฉพาะงานกราฟิกเท่านั้น แบ่งเป็นสองค่ายใหญ่ๆ คือ Nvidia ใช้ CUDA, AMD (ATI) ใช้ ATI Stream SDK นอกจากการใช้เครื่องมือที่ติดต่อกับกราฟิกการ์ดโดยตรงจากผู้ผลิตโดยตรงแล้ว เรายังสามารถที่จะ เขียนโดยอ้างการใช้งาน API ได้แก่ Opengl (ภาษา GLSL), Direct 3D (HLSL) ได้อีก ปัจจุบัน OpenCL เป็นการร่วมกันของผู้ผลิตฮาร์ดแวร์ซอฟท์แวร์มาร่วมกันสร้างมาตรฐานให้กับการใช้งาน GPGPU

จากสององค์กรใหญ่ที่ผลิต GPU คือ ATI และ NVIDIA ซึ่งปี 2006 NVIDIA ได้ออกแบบเครื่องมือซอฟต์แวร์มาเพื่อลดความซับซ้อนของการพัฒนาโปรแกรมประยุกต์ GPGPU ส่วน ATI

ได้ผลิต Close-to-the-Metal (CTM) ซึ่งเป็นอินเตอร์เฟซที่มีระดับค่อนข้างต่ำสำหรับการเขียนโปรแกรมที่ส่งไปยัง API กราฟิก จากนั้น NVIDIA ได้มองเห็นถึงจุดอ่อนดังกล่าว จึงทำการผลิตเครื่องมือชื่อว่า CUDA สำหรับให้นักเขียนโปรแกรมสามารถพัฒนาระบบทั่วไปบน GPU ได้ด้วยภาษา C ซึ่ง CUDA จะสนับสนุนกับ NVIDIA GeForce ทั้งหมด รวมถึง Quadro และ ผลิตภัณฑ์ของ Tesla การพัฒนาระบบทั่วไปด้วย CUDA ได้รับความนิยมกันอย่างแพร่หลาย และงานนี้จะมุ่งเน้นเฉพาะการพัฒนาโปรแกรมโดยใช้ CUDA

สถาปัตยกรรมคูด้า

NVIDIA ได้ออกแบบเครื่องมือสำหรับพัฒนาซอฟต์แวร์ในหน่วยประมวลผลกราฟิก เพื่อลดความซับซ้อนในการพัฒนา เรียกเครื่องมือนี้ว่า CUDA

CUDA หรือ Compute Unified Device Architecture (John Nickolls, Ian Buck, Michael Garland และ Kevin Skadron, 2008; Shifu Chen, Jing Qin, Yongming Xie, Junping Zhao และ Pheng-Ann Heng) เป็นสถาปัตยกรรมฮาร์ดแวร์ และซอฟต์แวร์ที่ช่วยในการออกแบบ และจัดการของการคำนวณบน GPU สถาปัตยกรรม CUDA ถูกสร้างขึ้นโดย NVIDIA โดยจะประกอบด้วยตัวกราฟิกการ์ด, Programming Model, คอมไพเลอร์ ที่จะช่วยในการเขียนโปรแกรมแบบขนาน (Parallel Programming) ที่ทำงานบนกราฟิกการ์ดได้ จากการทำงานแบบขนานของ CUDA ทำให้สามารถทำการประมวลผลอัลกอริทึมเดิมแบบหลายครั้งบน GPU ได้

โปรแกรมภาษาของ CUDA เป็นภาษาที่ตั้งอยู่บนพื้นฐานของมาตรฐานของภาษา C ซึ่งสถาปัตยกรรมของ CUDA จะสนับสนุนในระดับของอุปกรณ์ของ API และสนับสนุน OpenCL และ DirectX 11

CUDA Code ถูกประมวลผลบน CPU และ GPU ซึ่งถูกเก็บไว้ในไฟล์ข้อมูลเดียวกัน จากไฟล์สกุล ".cu" และใช้ "nvcc" เป็นตัวคอมไพล์

CUDA มีขั้นตอนการทำงานดังต่อไปนี้

1) จัดสรรข้อมูลบนอุปกรณ์
2) ถ่ายโอนข้อมูลจากโฮสต์ไปยังอุปกรณ์
3) เริ่มต้นหน่วยความจำอุปกรณ์ (ถ้าต้องการ)
4) กำหนดค่าขององค์ประกอบในการดำเนินงาน
5) ทำการประมวลผล (ผลลัพธ์ที่ได้จะเก็บไว้ในหน่วยความจำอุปกรณ์)
6) โอนถ่ายข้อมูลจากอุปกรณ์มายังโฮสต์

สรุปข้อดีและข้อเสียของ CUDA ข้อดีคือ สามารถเขียนโปรแกรมที่ทำงานบนกราฟิกการ์ดได้ เป็นการแบ่งเบาภาระการประมวลผลของ CPU ลง และโปรแกรมสามรถทำงานได้เร็วขึ้น แต่ว่าข้อเสียคือ CUDA เป็นสถาปัตยกรรมของ NVIDIA จึงสนับสนุนกราฟิกการ์ดของ NVIDIA เท่านั้น

วิธีการซื้อขายหลักทรัพย์

การซื้อขายหลักทรัพย์โดยผ่านระบบการซื้อขายของตลาดหลักทรัพย์ได้ 2 วิธี ได้แก่

1. Automatic Order Matching (AOM)

เป็นวิธีการซื้อขายที่ผู้ซื้อและผู้ขายส่งการเสนอซื้อและเสนอขายด้วยคอมพิวเตอร์ผ่านเข้ามายังระบบ การซื้อขายของตลาดหลักทรัพย์ โดยที่ระบบคอมพิวเตอร์ของตลาดหลักทรัพย์จะทำการเรียงลำดับ และจับคู่คำสั่งซื้อขายให้โดยอัตโนมัติ

การจัดเรียงลำดับคำสั่งซื้อขายเมื่อสามารถส่งคำสั่งซื้อขายเข้ามา ระบบการซื้อขายจะเก็บคำสั่งซื้อขายไว้ตั้งแต่เวลาที่ส่งคำสั่งซื้อขาย จนถึงสิ้นวันทำการ และจัดเรียงคำสั่งซื้อขายตามลำดับของราคาและเวลาที่ดีที่สุด (Price then Time Priority) โดยมีหลักการคือ

- คำสั่งซื้อที่มีราคาเสนอซื้อสูงที่สุดจะถูกจัดเรียงไว้ในลำดับที่หนึ่ง และถ้ามีราคาเสนอซื้อที่สูงกว่าถูกส่งเข้ามาใหม่ จะจัดเรียง ราคาเสนอซื้อที่สูงกว่าเป็นการเสนอซื้อในลำดับแรกก่อนและถ้ามีการเสนอซื้อในแต่ละราคามากกว่าหนึ่งรายการ ให้จัดเรียงตามเวลา โดยการเสนอซื้อที่ปรากฏในระบบการซื้อขายก่อนจะถูกจัดไว้เป็นการเสนอซื้อในลำดับก่อน

- คำสั่งขายที่มีราคาเสนอขายต่ำที่สุดจะถูกจัดเรียงไว้ในลำดับที่หนึ่ง และถ้ามีราคาเสนอขายที่ต่ำกว่าถูกส่งเข้ามาใหม่จะจัดเรียงราคาเสนอขายที่ต่ำกว่า เป็นการเสนอขายในลำดับแรกก่อนละถ้ามีการเสนอขายในแต่ละราคามากกว่าหนึ่งรายการให้จัดเรียงตามเวลา โดยการเสนอขายที่ปรากฏในระบบการซื้อขาย ก่อนจะถูกจัดไว้เป็นการเสนอขายในลำดับก่อน

การจับคู่การซื้อขาย (Matching) เมื่อคำสั่งซื้อขายผ่านเข้ามาในระบบซื้อขายแล้ว ระบบซื้อขายจะตรวจสอบว่าคำสั่งนั้นสามารถจับคู่กับคำสั่ง ด้านตรงข้ามได้ทันทีหรือไม่ ถ้าคำสั่งนั้นสามารถจับคู่ได้ทันที ระบบก็จะทำการจับคู่ให้ แต่ถ้าคำสั่งนั้น ไม่สามารถจับคู่ได้ ระบบจะจัดเรียงคำสั่ง ซื้อขายนั้นตามหลักการ Price then Time Priority ตามที่กล่าวข้างต้น เพื่อรอการจับคู่คำสั่งต่อไป

2. Trade Report

เป็นวิธีการซื้อขายที่ผู้ซื้อและผู้ขายได้ทำการเจรจาต่อรองเพื่อตกลงซื้อขายกัน (Dealing) แล้วจึงบันทึกรายการซื้อขายนั้นเข้ามา ในระบบการซื้อขาย (Trade Report) โดยบริษัทสมาชิกสามารถประกาศ โฆษณา (Advertise) การเสนอซื้อหรือ เสนอขายของตนผ่านระบบการซื้อขายได้

การซื้อขายด้วยวิธี Trade Report แบ่งเป็น 2 ประเภทคือ

1) การซื้อขายระหว่างสมาชิก (Two-firm Trade Report) โดยเมื่อมีการตกลงซื้อขายกันแล้วให้สมาชิกผู้ขายและสมาชิกผู้ซื้อบันทึกรายการซื้อขายเข้ามาในระบบการซื้อขาย

2) การซื้อขายโดยสมาชิกผู้ซื้อและผู้ขายเป็นรายเดียวกัน (One-firm Trade Report) โดยเมื่อสมาชิกสามารถจับคู่การซื้อขายระหว่างผู้ซื้อและผู้ขายได้แล้ว ให้สมาชิกบันทึกรายการซื้อขายเข้ามาในระบบการซื้อขาย

งานวิจัยนี้ได้เลือกใช้วิธีการซื้อขายหลักทรัพย์แบบ AOM ในงานวิจัย ซึ่งได้ทำการศึกษาการประมวลผลของขั้นตอนการเรียงลำดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขาย โดยทำการออกแบบการทำงานของทั้งสองขั้นตอนเป็นแบบขนาน เพื่อรองรับการทำงานแบบขนานของหน่วย

ประมวลผลกราฟิก พร้อมแสดงเวลาในการประมวลผล เพื่อทำการวิเคราะห์สมรรถนะการประมวลผลด้วยหน่วยประมวลผลกราฟิก และปัจจัยที่มีผลกระทบต่อสมรรถนะในการประมวลผลด้วย

งานที่เกี่ยวข้อง

ข้อมูลหลักทรัพย์ถือเป็นข้อมูลที่สำคัญในงานด้านการเงิน มีหลายหน่วยงานสนใจศึกษาการพัฒนาปรับปรุงการทำงานในส่วนต่างๆของงานด้านการเงิน เช่น การกำหนดราคาหลักทรัพย์ ซึ่งมีหลายสูตรทฤษฎีในการทำการศึกษาพฤติกรรมความเสี่ยง เพื่อคำนวณการกำหนดราคา

ทฤษฎี Model Base Pricing เป็นสูตรตามทฤษฎีการกำหนดราคาหลักทรัพย์ จากนั้นก็จะสามารถคำนวณผลตอบแทน หรือแยกองค์ประกอบของผลตอบแทนออกมาแล้วจึงศึกษาพฤติกรรมความเสี่ยง การกำหนดราคาโดยใช้สูตรตามทฤษฎีนั้นทำให้การคำนวณง่าย แต่มีข้อเสีย คือ ต้องรู้ว่าจะต้องใช้สูตรหรือสมการใด, แต่ละสูตร มีข้อสมมติฐานซึ่งอาจไม่สอดคล้องกับความเป็นจริง เช่น สมมติฐานการกระจายของข้อมูลเป็นแบบโค้งปกติ และ บางหลักทรัพย์ที่ซับซ้อน อาจไม่สามารถมีสูตรสำเร็จในการหาราคาและวิเคราะห์องค์ประกอบ

ทฤษฎี Monte Carlo Simulation เป็นวิธีการกำหนดราคาโดยใช้วิธีการจำลองสถานการณ์ (Simulation) ซึ่งเป็นวิธีการทางคณิตศาสตร์เพื่อหาผลลัพธ์ของเกิดเหตุการณ์หรือคำตอบโดยการจำลองสถานการณ์ (Simulation) ที่มาของชื่อของวิธีการมาจากนักคณิตศาสตร์ที่หาคำตอบโดยการทดสอบผลลัพธ์จากสถานที่คาซิโนในเมืองมอนติคาร์โล

ในทางการเงินใช้มอนติคาร์โลเพื่อประเมินว่าราคาหลักทรัพย์ในอนาคตจะมีค่าเป็นเท่าใด เมื่อประเมินราคาหลักทรัพย์ได้ก็สามารถนำไปคำนวณราคาตราสารอนุพันธ์อื่นที่ซับซ้อน เช่น ออปชันแบบอาเซียน ที่มีราคาอ้างอิงกับราคาหลักทรัพย์ (หุ้น) ได้

ปัจจุบันได้มีการพัฒนาเทคโนโลยี และเทคนิคใหม่ๆ เพื่อช่วยในการคำนวณการกำหนดราคาหลักทรัพย์ ซึ่งการนำ GPUs มาช่วยในการคำนวณถือเป็นอีกวิธีในการพัฒนาดังกล่าว

GPUs ได้ถูกนำมาช่วยในการจำลอง Monte Carlo ซึ่งเป็นการสร้างแบบจำลองที่มีความซับซ้อน เพราะ GPUs มีหน่วยประมวลผลแบบขนาน สามารถทำงานในส่วนต่างๆของปัญหาได้พร้อมกันได้อย่างมีประสิทธิภาพมากขึ้นกว่าการคำนวนด้วย CPU เพียงอย่างเดียว

GPUs กำลังถูกใช้ในบางส่วนของงานด้านการเงิน ในปี 2008 เริ่มใช้ GPUs ในการจำลอง Monte Carlo สำหรับคำนวณราคาหลักทรัพย์ที่ยากต่อการกำหนดราคา เพื่อช่วยพัฒนาเครื่องมือทางการเงินที่ซับซ้อน เช่น การซื้อขายสัญญาซื้อขายล่วงหน้า และใน 2011, JP Morgan กลายเป็นธนาคารแรกที่ใช้ GPUs ช่วยในการคำนวณการจำลอง Monte Carlo

Wood (2010) ได้ทำการศึกษาการนำ GPUs มาช่วยในการคำนวณการจัดการความเสียงทางการเงิน โดยมุ่งเน้นใช้กับระบบงานจริงในปัจจุบัน เพื่อพัฒนาเทคโนโลยีของระบบงานที่อยู่ให้ได้รับการประเมินและวิเคราะห์อย่างต่อเนื่อง ซึ่งงานวิจัยนี้ได้เลือกแอพพิเคชั่นของการจัดการความเสี่ยง โดยนำเอา GPUs มาช่วยพัฒนา เพื่อแสดงให้เห็นว่า GPU แพลตฟอร์มสามารถช่วยการคำนวณข้อมูลทางการเงินได้จากประสิทธิภาพการทงานในปริมาณจริง และนำทรัพยากรฮาร์ดแวร์ที่มีอยู่ในระบบมาปรับใช้ให้เกิดประโยชน์สูงสุด

Dang (2011) ได้พัฒนากรอบการสร้างแบบจำลองที่มีประสิทธิภาพผ่านทางวิธีการบางส่วนของสมการเชิงอนุพันธ์ (PDE) สำหรับตราสารอนุพันธ์ทางการเงินหลายปัจจัยที่มีความสำคัญในสามรูปแบบปัจจัย และศึกษาการใช้งานที่มีประสิทธิภาพสูงของวิธีการเชิงตัวเลขที่มีประสิทธิภาพสูงของสถาปัตยกรรมคอมพิวเตอร์ที่มีความสำคัญ โดยเฉพาะอย่างยิ่งในหน่วยประมวลผลกราฟิก (GPUs) และ multi-GPU platforms/clusters ของ GPUs

Grauer-Gray, Killian, Searles และ Cavazos (2013) ได้นำเอาพลังการประมวลผลแบบขนานของ GPU มาใช้ในการเร่งการคำนวณโปรแกรมประยุกต์ทางการเงินใน QuantLib เช่น Black-Scholes, Monte-Carlo พันธบัตร และ Repo ซึ่งมีการเร่งการคำนวนโดยใช้ CUDA และ OpenCL ซึ่งแสดงให้เห็นถึงการเร่งความเร็วอย่างมีนัยสำคัญโดยใช้วิธีการแบบขนาน และยังสามารถเพิ่มการเร่งความเร็วได้

อุปกรณ์และวิธีการ

อุปกรณ์

อุปกรณ์ที่ใช้ในงานวิจัย แบ่งออกเป็น 3 ส่วน ดังต่อไปนี้

1. ฮาร์ดแวร์ ในส่วนนี้แสดงรายละเอียดของเครื่องคอมพิวเตอร์ หน่วยความจำหลัก (RAM) และกราฟิกการ์ด

2. ซอฟต์แวร์ ในส่วนนี้แสดงรายละเอียดของระบบปฏิบัติการคอมพิวเตอร์ คอมไพเลอร์ และ MPI

3. เครื่องมือ CUDA ในส่วนนี้แสดงรายละเอียดของเครื่องมือ คือ จำนวนโปรเซสเซอร์ จำนวนคอร์ต่อโปรเซสเซอร์ ขนาดหน่วยความจำ ขนาดหน่วยความจำที่แชร์ต่อบล็อก จำนวนเทรดต่อบล็อก และขนาดของวาร์ป

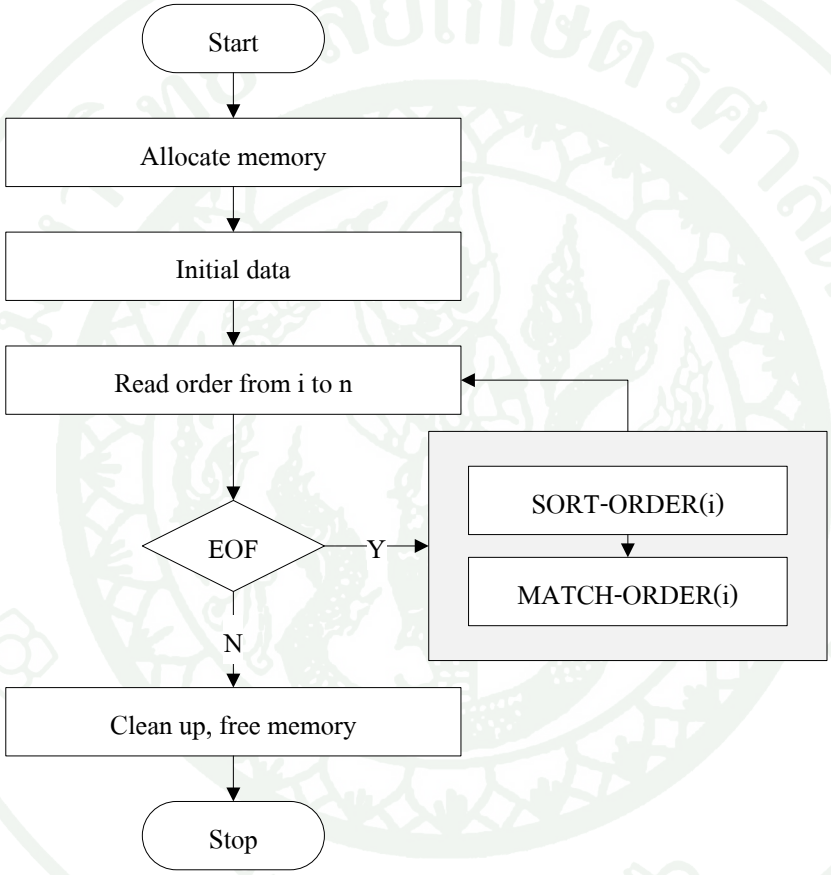
จากรายละเอียดของอุปกรณ์ทั้ง 3 ส่วน ฮาร์ดแวร์ ซอฟต์แวร์ และเครื่องมือ CUDA สามารถแสดงในรูปแบบของตาราง ดังนี้

ตารางที่ 1 รายละเอียดของอุปกรณ์ ฮาร์ดแวร์ ซอฟต์แวร์ และเครื่องมือ CUDA

Hardware/Software	**Configurations/Specifications**
Host	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz
RAM	16 GB
Device	NVIDIA Tesla M2050
# of Multi-processors	14
# of cores per Multi-processor	32
Memory	2817982464 bytes
Shared memory per block	49152 bytes
Max # of threads per block	1024
Warp size	32
Operating System	Rocks Cluster 5.4 (x86-64)
Compiler	gcc version 4.1.2 20080704 (Red Hat 4.1.2-48)
MPI	OpenMPI 1.4.3

วิธีการ

งานวิจัยนี้ได้ศึกษาการนำหน่วยประมวลผลกราฟิก หรือ Graphics Processing Unit (GPU) มาช่วยเร่งความเร็วในการประมวลผลข้อมูลในงานด้านการเงิน โดยทำการศึกษาการประมวลผลอัลกอริทึมการซื้อขายขายหลักทรัพย์ด้วยวิธี AOM (Automatic Order Matching) การซื้อขายด้วยวิธีนี้เป็นการซื้อขายหลักทรัพย์แบบอัตโนมัติ โดยระบบทำการเรียงลำดับคำสั่งซื้อขาย และจับคู่คำสั่งซื้อขายโดยอัตโนมัติ โดยมีขั้นตอนการทำงานดังนี้



ภาพที่ 1 แผนผังการทำงานของโปรแกรม

ขั้นตอนการทำงานของโปรแกรมเริ่มจากจัดสรรหน่วยความจำ กำหนดค่าเริ่มต้นตัวแปร อ่านรายการคำสั่งซื้อคำสั่งขาย ประมวลผลคำสั่ง และ คืนค่าหน่วยความจำ ตามลำดับ โดยแต่ละขั้นตอนมีการทำงาน ดังนี้

1) จัดสรรหน่วยความจำ เป็นขั้นตอนการจัดสรรหน่วยความจำที่ใช้ในการประมวลผล
2) กำหนดค่าเริ่มต้นตัวแปร เป็นการกำหนดค่าเริ่มต้นของตัวแปรก่อนทำการประมวลผล

3) อ่านข้อมูลรายการซื้อขาย เป็นขั้นตอนอ่านข้อมูลรายการซื้อขาย โดยทำการอ่านข้อมูลจากรายการเริ่มต้น ถึงรายการที่ต้องการทำงานประมวลผล

4) ประมวลผลข้อมูล เป็นขั้นตอนการประมวลผลคำสั่งการจับคู่คำสั่งซื้อขาย โดยทำการเรียงลำดับคำสั่งซื้อขายก่อน แล้วทำการจับคู่คำสั่งซื้อขายเป็นลำดับถัดไป

6) คืนค่าหน่วยความจำ เป็นขั้นตอนการคืนค่าหน่วยความจำที่ได้จัดสรรเอาไว้

จากขั้นตอนการทำงานของโปรแกรม งานวิจัยนี้ได้นำหน่วยประมวลผลกราฟิกมาช่วยประมวลผลในขั้นตอนการประมวลผลการซื้อขายหลักทรัพย์ ในขั้นตอนเรียงลำดับคำสั่งซื้อขายและจับคู่คำสั่งซื้อคำสั่งขาย โดยเลือกใช้ CUDA เป็นเครื่องมือในการติดต่อกับกราฟิกการ์ด พร้อมทำการเปรียบเทียบเวลาที่ใช้ในการประมวลผลระหว่างประมวลผลบนหน่วยประมวลผลกลางอย่างเดียว และนำหน่วยประมวลผลกราฟิกมาช่วยประมวลผล ทั้งสองวิธีมีคำสั่งการทำงานเหมือนกันต่างกันที่หน่วยประมวลผลกลางเป็นการทำงานแบบเรียงลำดับ ส่วนหน่วยประมวลกราฟิกเป็นการทำงานแบบขนาน การประมวลผลการซื้อขายหลักทรัพย์ แบ่งขั้นตอนการทำงานออกเป็น 2 ส่วน คือ การเรียงลำดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขาย ซึ่งแต่ละส่วนมีการทำงาน ดังนี้

การเรียงลำดับคำสั่งซื้อขาย

การเรียงลำดับคำสั่งซื้อขายเป็นหนึ่งในขั้นตอนการซื้อขายหลักทรัพย์ ได้ทำการเรียงลำดับคำสั่งซื้อขายตาม ชื่อหุ้น ประเภทการซื้อขาย (ซื้อ/ขาย) และราคาซื้อขาย ตามลำดับ หลังจากทำการเรียงลำดับคำสั่งซื้อขายเรียบร้อยแล้วจึงเริ่มทำการจับคู่คำสั่งซื้อขายในลำดับถัดไป โดยมีขั้นตอนการทำงานของการเรียงลำดับคำสั่งซื้อขายตามภาพ 2

```
//Sorting-----------------------------------------------------------------
for (iCurrStock=0;iCurrStock<MAX_STOCK;iCurrStock++){
        if (HeadBuy[iCurrStock] != -1 and TailBuy[iCurrStock] != -1){ //Buy
                swapped = true, iLoopSort = 0;
                while (swapped){
                        swapped = false;
                        iLoopSort++;
                        iCurrBuy = HeadBuy[iCurrStock];
                        iNextBuy = Next[iCurrBuy];
                        for (long i=0;i<CntBuy[iCurrStock]-iLoopSort;i++){
                                if (H_Price[iCurrBuy] < H_Price[iNextBuy]){
                                        //Exchange Order[iCurrBuy] <-> Order[iNextBuy]();
                                        swapped = true;
                                }
                                iCurrBuy = iNextBuy;
                                iNextBuy = Next[iCurrBuy];
                        }
                }
        }

        if (HeadSell[iCurrStock] != -1 and TailSell[iCurrStock] != -1){ //Sell
                swapped = true, iLoopSort = 0;
                while (swapped){
                        swapped = false;
                        iLoopSort++;
                        iCurrSell = HeadSell[iCurrStock];
                        iNextSell = Next[iCurrSell];
                        for (long i=0;i<CntSell[iCurrStock]-iLoopSort;i++){
                                if (H_Price[iCurrSell] > H_Price[iNextSell]){
                                        //Exchange Order[iCurrSell] <-> Order[iNextSell]();
                                        swapped = true;
                                }
                                iCurrSell = iNextSell;
                                iNextSell = Next[iCurrSell];
                        }
                }
        }
}
```

ภาพที่ 2 การทำงานของอัลกอริทึมการเรียงลำดับคำสั่งซื้อขาย

จากภาพที่ 2 เป็นการแสดงการทำงานของการเรียงดับคำสั่งซื้อขาย ซึ่งการทำงานแบบนี้จะเริ่มทำการประมวลผลตั้งแต่ iCurrStock ถึง MAX_STOCK โดยกำหนดให้ iCurrStock แทนเลขที่หุ้นที่ทำการจัดเรียง และ MAX_STOCK แทนเลขที่หุ้นสุดท้ายที่ทำการจัดเรียง การทำงานด้วยวิธีนี้เป็นการจัดเรียงลำดับคำสั่งซื้อขายของหุ้นตัวแรกจนครบตามจำนวนหุ้นทั้งหมด

ในขั้นตอนการเรียงลำดับคำสั่งซื้อขายยังบอกข้อมูลตำแหน่งเริ่มต้น ตำแหน่งสิ้นสุด และถัดไปของรายการแต่ละที่หุ้น ดังตัวอย่าง

StockId		head	tail
0	bid	-1	-1
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	-1	-1

		head	tail
2	bid	-1	-1
2	offer	-1	-1

		head	tail
3	bid	-1	-1
3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(a)

StockId		head	tail
0	bid	0	0
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	1	1

		head	tail
2	bid	2	2
2	offer	-1	-1

		head	tail
3	bid	3	3
3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(b)

StockId		head	tail
0	bid	0	0
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	**4**	1

		head	tail
2	bid	2	2
2	offer	-1	-1

		head	tail
3	bid	3	3
3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	**1**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(c)

StockId		head	tail
0	bid	0	0
0	offer	-1	-1

		head	tail
1	bid	4	1
1	offer	4	1

		head	tail
2	bid	2	2
2	offer	**5**	**5**

		head	tail
3	bid	3	3
3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	**1**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(d)

StockId		head	tail
0	bid	0	0
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	4	1

		head	tail
2	bid	2	2
2	offer	5	5

		head	tail
3	bid	3	**6**
3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	**1**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(e)

ภาพที่ 3 ตัวอย่างการกำหนดตำแหน่งเริ่มต้น ตำแหน่งสิ้นสุด และตำแหน่งถัดไปของรายการ

StockId		head	tail			head	tail			head	tail			head	tail
0	bid	0	0	1	bid	-1	-1	2	bid	2	2	3	bid	3	6
0	offer	-1	-1	1	offer	7	1	2	offer	5	5	3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(f)

StockId		head	tail			head	tail			head	tail			head	tail
0	bid	0	0	1	bid	-1	-1	2	bid	2	2	3	bid	3	6
0	offer	-1	-1	1	offer	7	8	2	offer	5	5	3	offer	-1	-1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(g)

StockId		head	tail			head	tail			head	tail			head	tail
0	bid	0	0	1	bid	-1	-1	2	bid	2	2	3	bid	3	6
0	offer	-1	-1	1	offer	7	8	2	offer	5	5	3	offer	9	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	2	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	6450	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	38100	4900	20500	23700	24600	45600	25900	37300	13400	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(h)

StockId		head	tail			head	tail			head	tail			head	tail
0	bid	0	0	1	bid	-1	-1	2	bid	2	2	3	bid	3	6
0	offer	-1	-1	1	offer	7	8	2	offer	5	5	3	offer	9	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	-1	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	-1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	24700	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	-1	-1	1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(i)

StockId		head	tail			head	tail			head	tail			head	tail
0	bid	0	0	1	bid	-1	-1	2	bid	2	11	3	bid	3	6
0	offer	-1	-1	1	offer	7	8	2	offer	5	5	3	offer	9	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	-1	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	-1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	24700	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	-1	-1	11	-1	1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(j)

ภาพที่ 3 (ต่อ)

StockId		head	tail
0	bid	0	12
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	2	11
2	offer	5	5

		head	tail
3	bid	3	6
3	offer	9	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	2	3	1	2	3	1	1	3	-1	2	0	2	2	3	2	1	2	0
SignId	0	1	0	0	1	1	0	1	1	1	-1	0	0	1	0	1	0	0	1	0
Price	11300	8650	12550	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	24700	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	49700	18100	32500	4300	8900	14600	24300
Next	12	-1	11	-1	1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(k)

StockId		head	tail
0	bid	0	12
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	11	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	2	1	2	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	0	0	1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	-1	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	25000	18100	32500	4300	8900	14600	24300
Next	12	-1	11	-1	1	13	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(l)

StockId		head	tail
0	bid	0	12
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	14	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	2	1	2	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	0	0	1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	-1	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	25000	18100	32500	4300	8900	14600	24300
Next	12	-1	11	-1	1	13	-1	4	-1	-1	-1	-1	-1	-1	11	-1	-1	-1	-1	-1

(m)

StockId		head	tail
0	bid	0	12
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	14	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	2	1	2	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	0	0	1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	12150	7350	7200	8150
Volume	40300	23500	-1	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	25000	18100	32500	4300	8900	14600	24300
Next	12	-1	11	-1	1	13	-1	4	-1	15	-1	-1	-1	-1	11	-1	-1	-1	-1	-1

(n)

StockId		head	tail
0	bid	0	12
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	7	8

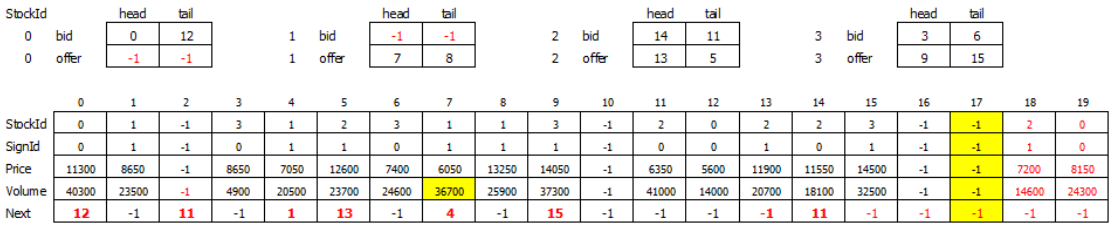
		head	tail
2	bid	14	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	-1	1	2	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	-1	0	1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	-1	7350	7200	8150
Volume	40300	23500	-1	4900	20500	23700	24600	45600	25900	37300	-1	41000	14000	20700	18100	32500	-1	8900	14600	24300
Next	12	-1	11	-1	1	13	-1	4	-1	15	-1	-1	-1	-1	11	-1	-1	-1	-1	-1

(o)

ภาพที่ 3 (ต่อ)



StockId		head	tail
0	bid	0	12
0	offer	-1	-1

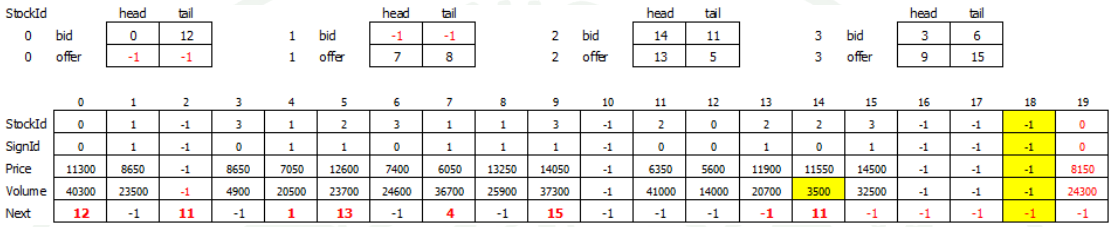
		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	14	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	-1	-1	2	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	-1	-1	1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	-1	-1	7200	8150
Volume	40300	23500	-1	4900	20500	23700	24600	36700	25900	37300	-1	41000	14000	20700	18100	32500	-1	-1	14600	24300
Next	12	-1	11	-1	1	13	-1	4	-1	15	-1	-1	-1	-1	11	-1	-1	-1	-1	-1

(p)



StockId		head	tail
0	bid	0	12
0	offer	-1	-1

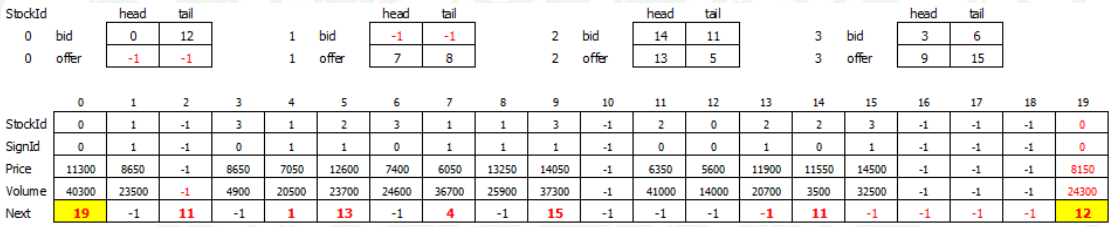
		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	14	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	-1	-1	-1	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	-1	-1	-1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	-1	-1	-1	8150
Volume	40300	23500	-1	4900	20500	23700	24600	36700	25900	37300	-1	41000	14000	20700	3500	32500	-1	-1	-1	24300
Next	12	-1	11	-1	1	13	-1	4	-1	15	-1	-1	-1	-1	11	-1	-1	-1	-1	-1

(q)



StockId		head	tail
0	bid	0	12
0	offer	-1	-1

		head	tail
1	bid	-1	-1
1	offer	7	8

		head	tail
2	bid	14	11
2	offer	13	5

		head	tail
3	bid	3	6
3	offer	9	15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
StockId	0	1	-1	3	1	2	3	1	1	3	-1	2	0	2	2	3	-1	-1	-1	0
SignId	0	1	-1	0	1	1	0	1	1	1	-1	0	0	1	0	1	-1	-1	-1	0
Price	11300	8650	-1	8650	7050	12600	7400	6050	13250	14050	-1	6350	5600	11900	11550	14500	-1	-1	-1	8150
Volume	40300	23500	-1	4900	20500	23700	24600	36700	25900	37300	-1	41000	14000	20700	3500	32500	-1	-1	-1	24300
Next	19	-1	11	-1	1	13	-1	4	-1	15	-1	-1	-1	-1	11	-1	-1	-1	-1	12

(r)

ภาพที่ 3 (ต่อ)

จากภาพที่ 3 เป็นการแสดงขั้นตอนการกำหนดตำแหน่งการประมวลผลจากการเรียงลำดับคำสั่งซื้อขายของแต่ละหุ้นตามเลขที่หุ้น จากตัวอย่างมีหุ้นทั้งหมด 4 หุ้น โดยมีเลขที่หุ้นเป็น 0, 1, 2 และ 3 ตามลำดับ การประมวลผลทำการประมวลผลทีละหุ้นจาก เลขที่หุ้น (0) ถึงเลขที่หุ้น (3) โดยกำหนดตำแหน่งเริ่มต้น ตำแหน่งถัดไป และตำแหน่งสุดท้ายของรายการแต่ละเลขที่หุ้น ภายใต้การจัดเรียงลำดับการซื้อขาย ซึ่งการจัดเรียงลำดับตำแหน่งรายการซื้อขายทำการแยกการเรียงจำดับคำสั่งซื้อ และคำสั่งขายออกจากัน การกำหนดตำแหน่งของรายการซื้อขายเริ่มจากกำหนดให้ตำแหน่งแรกของรายการซื้อขายนั้นเป็นตำแหน่งเริ่มต้น และสิ้นสุดของแต่ละเลขที่หุ้น จากนั้นทำการอ่านรายการซื้อขายถัดไป ถ้ารายการถัดไปเป็นข้อมูลของเลขที่หุ้นเดียวกันจะทำการจัดเรียงคำสั่งซื้อขายกับรายการซื้อขายของหุ้นตามตำแหน่งที่ระบุไว้ในตัวแปรของแต่ละหุ้น โดยทำการจัดเรียงรายการคำสั่งซื้อขายจากตำแหน่งเริ่มต้นถึงสิ้นสุด พร้อมทั้งบอกตำแหน่งใหม่เมื่อมีการเปลี่ยนแปลงข้อมูลตำแหน่งดังกล่าว

การจับคู่คำสั่งซื้อขาย

การจับคู่คำสั่งซื้อขาย เป็นขั้นตอนการจับคู่คำสั่งซื้อกับคำสั่งขาย ตามราคาและจำนวนที่สามารถทำการซื้อขายได้ โดยมีขั้นตอนการทำงานของการจับคู่คำสั่งซื้อขายตามภาพ 4

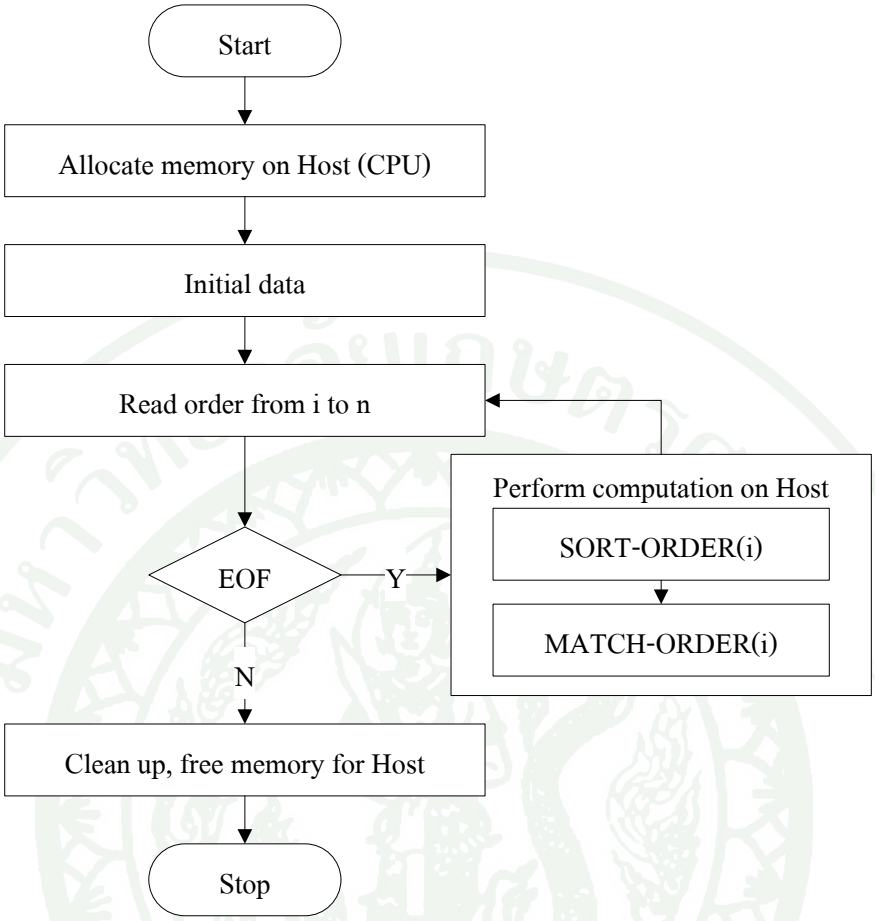
```
//Matching-----------------------------------------------------------------
for (iCurrStock=0;iCurrStock<MAX_STOCK;iCurrStock++){
        fBuy = true;
        iCurrBuy = HeadBuy[iCurrStock];
        while (iCurrBuy != -1 && iCurrBuy <= TailBuy[iCurrStock] && fBuy == true){
                fSell = true;
                iCurrSell = HeadSell[iCurrStock];
                while (H_Volume[iCurrBuy] > 0 && iCurrSell != -1 && iCurrSell <= TailSell[iCurrStock] && fSell == true){
                        if (H_Price[iCurrBuy] >= H_Price[iCurrSell]){
                                if (H_Volume[iCurrSell] > 0){
                                        if (H_Volume[iCurrBuy] >= H_Volume[iCurrSell]){
                                                //Write log file ();
                                                H_Volume[iCurrBuy] = H_Volume[iCurrBuy] - H_Volume[iCurrSell];
                                                H_Volume[iCurrSell] = 0;
                                        }
                                        else {
                                                //Write log file ();
                                                H_Volume[iCurrSell] = H_Volume[iCurrSell] - H_Volume[iCurrBuy];
                                                H_Volume[iCurrBuy] = 0;
                                        }
                                }
                                iCurrSell = Next[iCurrSell];
                        }
                        else{
                                fSell = false;
                        }
                }

                if (H_Volume[iCurrBuy] > 0){
                        fBuy = false;
                }

                iCurrBuy = Next[iCurrBuy];
        }
}
```

ภาพที่ 4 การทำงานของอัลกอริทึมการจับคู่คำสั่งซื้อขาย

งานวิจัยนี้ได้แบ่งการประมวลผลอัลกอริทึมการซื้อขายหลักทรัพย์ออกเป็น 2 วิธี คือ 1) ทำการประมวลผลบนหน่วยประมวลผลการอย่างเดียว 2) ทำการประมวลผลด้วยหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิก ซึ่งมีขั้นตอนการประมวล ดังนี้

การประมวลผลอัลกอริทึมบนหน่วยประมวลผลกลางอย่างเดียว

การประมวลผลด้วยวิธีนี้เป็นการประมวลผลแบบเรียงลำดับ (Sequential Processing) มีขั้นตอนการประมวลผล ดังนี้

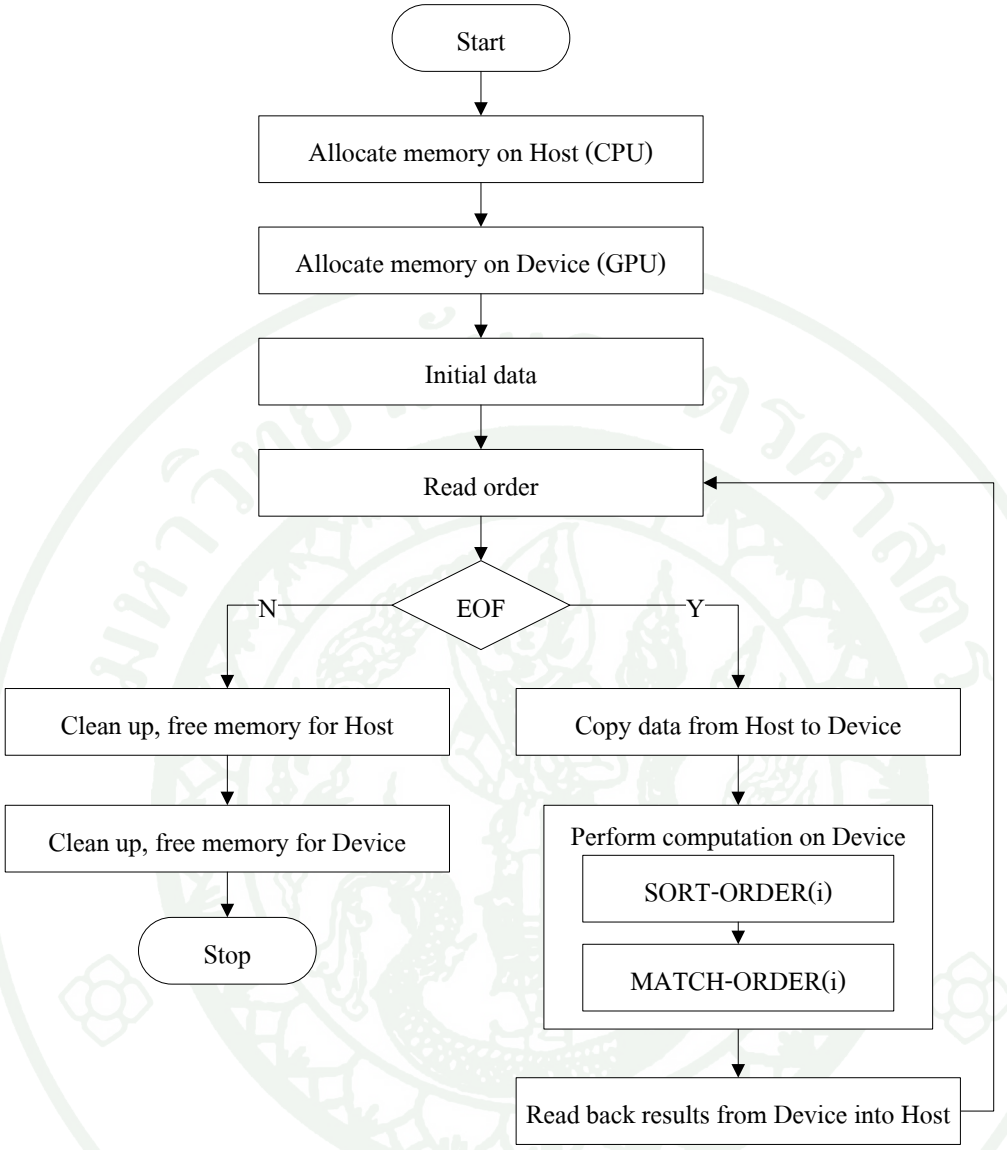


ภาพที่ 5 แผนผังการประมวลผลอัลกอริทึมด้วย CPU

จากภาพที่ 5 เป็นขั้นตอนการประมวลผลอัลกอริทึมด้วยหน่วยประมวลผลกลางเพียงอย่างเดียว วิธีนี้เป็นทำงานแบบเรียงลำดับ โดยทำการประมวลผลคำสั่งได้ทีละคำสั่งทีละหุ้นจนครบทุกหุ้น

การประมวลผลอัลกอริทึมบนหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิก

การประมวลผลด้วยวิธีนี้เป็นการประมวลผลแบบขนาน ทำงานแบบ Multi-Processors วิธีนี้เป็นการใช้เทคโนโลยีที่มีอยู่แล้วในกราฟิกการ์ดมาใช้ในการประมวลผลข้อมูล และต้องการประมวลผลคำสั่งในการทำงานเดียวกัน ซึ่งวิธีนี้สามารถช่วยเร่งความเร็วในการประมวลผลคำสั่งได้ มีขั้นตอนการทำงาน ดังนี้



ภาพที่ 6 แผนผังการประมวลผลอัลกอริทึมด้วย CPU และ GPU

จากภาพที่ 6 จะเห็นว่าการประมวลผลอัลกอริทึมด้วยวิธีนี้มีขั้นตอนการทำงานเพิ่มขึ้นจากการประมวลผลด้วยหน่วยประมวลผลกลางอย่างเดียว คือ ขั้นตอนจัดสรรหน่วยความจำของกราฟิกการ์ด ขั้นตอนการถ่ายโอนข้อมูลไปยังกราฟิกการ์ด ขั้นตอนการส่งผลลัพธ์กลับมายังโฮสต์ และขั้นตอนการคืนหน่วยความจำของกราฟิกการ์ด จากขั้นตอนที่เพิ่มขึ้นส่งผลให้เวลาในการประมวลผลคำสั่งเพิ่มมากขึ้นด้วย การทำงานด้วยวิธีนี้ได้นำเอาหน่วยประมวลผลกราฟิก มาช่วยประมวลผลคำสั่งในขั้นตอนการเรียงลำดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขาย ซึ่งเป็นการนำเอา

ขีดความสามารถของกราฟิกการ์ดมาช่วยในการประมวลผลงานทั่วไปที่นอกเหนือจากการประมวลผลงานด้านกราฟิกอย่างเดียว

หน่วยประมวลผลกราฟิกมีความสามารถในการประมวลผลคำสั่งแบบขนาน ภายใต้คำสั่งเดียวกัน เหมาะกับงานที่ต้องการประมวลผลคำสั่งเดียวกันมากกว่า 1 งาน ความเหมาะในการแบ่งการประมวลผลคำสั่งบนหน่วยประมวลผลกราฟิกขึ้นอยู่กับขีดความสามารถของกราฟิกการ์ด และลักษณะของงาน งานวิจัยนี้ได้นำเอาหน่วยประมวลผลกราฟิกกมาช่วยในการประมวลผลคำสั่งของอัลกอริทึมการซื้อขายหลักทรัพย์ ในขั้นตอนการเรียงลำดับคำสั่งซื้อขาย และขั้นตอนการจับคู่คำสั่งซื้อขาย โดยทำการออกแบบการประมวลผลแบบขนานตามหุ้น เพื่อทำการแบ่งประมวลคำสั่ง

การประมวลผลขั้นตอนการเรียงลำดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขาย ถูกออกแบบให้รองรับการประมวลผลคำสั่งแบบขนาน โดยทำการแบ่งประมวลผลการเรียงลำดับคำสั่งซื้อขายตามหุ้น ดังนี้

```
idx = blockIdx.x * blockDim.x + threadIdx.x;
//Sorting-----------------------------------------------------------------
if (idx<D_MAX_ITEMS){
        iCurrStock = idx;
                if (D_HeadBuy[iCurrStock] != -1 and D_TailBuy[iCurrStock] != -1){ //Buy
                        swapped = true, iLoopSort = 0;
                        while (swapped){
                                swapped = false;
                                iLoopSort++;
                                iCurrBuy = D_HeadBuy[iCurrStock];
                                iNextBuy = D_Next[iCurrBuy];
                                for (long i=0;i<D_CntBuy[iCurrStock]-iLoopSort;i++){
                                        if (D_Price[iCurrBuy] < D_Price[iNextBuy]){
                                                //Exchange Order[iCurrBuy] <-> Order[iNextBuy]();
                                                swapped = true;
                                        }
                                        iCurrBuy = iNextBuy;
                                        iNextBuy = D_Next[iCurrBuy];
                                }
                        }
                }

                if (D_HeadSell[iCurrStock] != -1 and D_TailSell[iCurrStock] != -1){ //Sell
                        swapped = true, iLoopSort = 0;
                        while (swapped){
                                swapped = false;
                                iLoopSort++;
                                iCurrSell = D_HeadSell[iCurrStock];
                                iNextSell = D_Next[iCurrSell];
                                for (long i=0;i<D_CntSell[iCurrStock]-iLoopSort;i++){
                                        if (D_Price[iCurrSell] > D_Price[iNextSell]){
                                                //Exchange Order[iCurrSell] <-> Order[iNextSell]();
                                                swapped = true;
                                        }
                                        iCurrSell = iNextSell;
                                        iNextSell = D_Next[iCurrSell];
                                }
                        }
                }
        idx = idx + D_blockMax;
}
```

ภาพที่ 7 ขั้นตอนการประมวลผลการเรียงลำดับแบบขนาน

จากภาพที่ 7 เป็นการแสดงการทำงานของการเรียงดับคำสั่งซื้อขายแบบขนาน โดยได้ออกแบบการแบ่งการประมวลผลแบบขนานตามจำนวนหุ้น และอยู่ภายใต้ขีดความสามารถในการทำการงานแบบขนานของหน่วยประมวลผลกราฟิก ซึ่งการทำงานแบบนี้ได้ทำการประมวลผลคำสั่งพร้อมกันได้ตามจำนวนหุ้นทั้งหมด โดยกำหนดให้ idx แทนการทำงานของหุ้นแต่ละตัว และ D_MAX_ITEMS แทนจำนวนรายการทั้งหมดที่ทำการประมวลผล โดยทำการแบ่งประมวลผลการจับคู่คำสั่งซื้อขายตามหุ้น ดังนี้

```
idx = blockIdx.x * blockDim.x + threadIdx.x;
//Matching-----------------------------------------------------------------
if (idx<D_MAX_ITEMS){
	iCurrStock = idx;
		fBuy = true;
		iCurrBuy = D_HeadBuy[iCurrStock];
		while (iCurrBuy != -1 && iCurrBuy <= D_TailBuy[iCurrStock] && fBuy == true){
			fSell = true;
			iCurrSell = D_HeadSell[iCurrStock];
			while (D_Volume[iCurrBuy] > 0 && iCurrSell != -1 && iCurrSell <= D_TailSell[iCurrStock] && fSell == true){
				if (D_Price[iCurrBuy] >= D_Price[iCurrSell]){
					if (D_Volume[iCurrSell] > 0){
						if (D_Volume[iCurrBuy] >= D_Volume[iCurrSell]){
							//Write log file();
							D_Volume[iCurrBuy] = D_Volume[iCurrBuy] - D_Volume[iCurrSell];
							D_Volume[iCurrSell] = 0;
						}
						else {
							//Write log file();
							D_Volume[iCurrSell] = D_Volume[iCurrSell] - D_Volume[iCurrBuy];
							D_Volume[iCurrBuy] = 0;
						}
					}
					iCurrSell = D_Next[iCurrSell];
				}
				else{
					fSell = false;
				}
			}

			if (D_Volume[iCurrBuy] > 0){
				fBuy = false;
			}

			iCurrBuy = D_Next[iCurrBuy];
		}
	idx = idx + D_blockMax;
}
```

ภาพที่ 8 อัลกอริทึมการจับคู่คำสั่งซื้อขายแบบขนาน

จากภาพที่ 8 เป็นการแสดงการทำงานของการเรียงดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขายแบบขนาน โดยทำการออกแบบการแบ่งการประมวลผลออกตามจำนวนหุ้น และอยู่ภายใต้ขีดความสามารถในการทำงานงานแบบขนานของหน่วยประมวลผลกราฟิก คือ สามารถทำการประมวลผลได้พร้อมกันสูงสุดเท่ากับจำนวนเทรดของหน่วยประมวลผลกราฟิก ซึ่งการทำงานแบบนี้เป็นประมวลผลคำสั่งพร้อมกันได้ตามจำนวนหุ้น โดยกำหนดให้ idx แทนการทำงานงานของหุ้นแต่ละตัว และ D_MAX_ITEMS แทนจำนวนรายการทั้งหมดที่ทำการประมวลผล โดยทำการแบ่งประมวลผลการจับคู่คำสั่งซื้อขายตามหุ้น ดังนี้

ในขั้นตอนการประมวลผลคำสั่งบนหน่วยประมวลผลกราฟิก ทั้งในส่วนของการเรียงลำดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขาย ได้ทำการออกแบบการทำงานแบบขนาน และแบ่งการทำงานของแต่ละงานออกตามจำนวนหุ้น โดยกำหนดให้ idx แทนการทำงานของหุ้นแต่ละตัว เพื่อรองรับการทำงานแบบขนาน

ผลและวิจารณ์

ในการทดสอบการนำหน่วยประมวลผลกราฟิกมาช่วยประมวลผลทั่วไปนอกเหนือจากประมวลผลด้านกราฟิกอย่างเดียว โดยทำการศึกษาการประมวลผลอัลกอริทึมการซื้อขายหลักทรัพย์ด้วยวิธี AOM (Automatic Order Matching) เป็นการเรียงลำดับคำสั่งซื้อขาย และการจับคู่คำสั่งซื้อขายแบบอัตโนมัติ งานวิจัยนี้ทำการแบ่งการประมวลผลออกเป็น 2 แบบ คือ 1) ประมวลผลอัลกอริทึมด้วยหน่วยประมวลผลกลางอย่างเดียว 2) ประมวลผลอัลกอริทึมด้วยหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิก การประมวลผลของทั้งสองวิธีมีขั้นตอนการทำงานของคำสั่งเหมือนกัน ต่างกันที่วิธีที่ 2 ทำการประมวลผลการทำงานแบบขนาน คือ ทำการประมวลผลคำสั่งพร้อมกันได้มากกว่า 1 งานในเวลาเดียวกัน

ข้อมูลที่นำมาทำการทดสอบได้จากข้อมูลจำลองรายการคำสั่งซื้อ และคำสั่งขาย ซึ่งมีรายละเอียดของข้อมูลจำลอง ดังนี้ ชื่อหุ้น ประเภทการซื้อขาย(ซื้อ/ขาย) ราคาเสนอซื้อ/เสนอขาย โดยข้อมูลจำลองดังกล่าวถูกจำลองจากข้อมูลจริง ตามข้อมูลหุ้น ณ วันที่ทำการสร้างข้อมูลจำลอง ข้อมูลจำลองสำหรับนำมาทำการทดสอบมีจำนวนรายการทั้งหมด 2000000 รายการ โดยแบ่งข้อมูลสำหรับทดสอบการประมวลผลออกเป็น 4 กลุ่มข้อมูล คือ 500000 1000000 1500000 และ 2000000 ตามลำดับ ข้อมูลแต่ละกลุ่มมีขั้นตอนการประมวลผลข้อมูลเหมือนกัน จากการแบ่งกลุ่มข้อมูลจำลองออกเป็น 4 กลุ่ม เพื่อทดสอบเวลาที่ใช้ในการประมวลผลอัลกอริทึมของแต่ละกลุ่มข้อมูล และทำการเปรียบเทียบเวลาที่ใช้เมื่อนำหน่วยประมวลผลกราฟิกมาช่วยในการประมวลผล กับเวลาที่ใช้ในการประมวลผลด้วยหน่วยประมวลผลการเพียงอย่างเดียว

```
1  MDX, 5, 128, 400
2  SAT, S, 1011, 4500
3  MIDA, S, 74, 1000
4  TFI, S, 99, 800
5  IRP, S, 1188, 100
6  OISCHI, S, 4200, 20000
7  IMI, S, 515, 1000
8  ROJNA, B, 890, 500
9  NMIT, B, 985, 1000
10 NMG, S, 460, 2000
11 CCET, B, 358, 200
12 US, S, 310, 1700
13 URBNPF, B, 763, 1000
14 LTX, S, 3133, 1000
15 F&D, B, 1050, 100
16 TCCC, B, 565, 1000
17 SIM, B 176, 2500
18 KTB, S, 950, 1000
19 PTTEP, B, 14427, 5000
20 KCAR, B, 474, 500
```

ภาพที่ 9 แสดงข้อมูลจำลองรายการคำสั่งซื้อ คำสั่งขาย

ในการทดสอบการนำหน่วยประมวลผลกราฟิกมาช่วยเร่งความเร็วในการประมวลผล ได้มีการกำหนดค่าจำหนวนการทำงานแบบขนานที่ใช้ประมวลผลเป็น 64 128 256 512 1024 2048 4096 8192 16384 และ 32768 เพื่อดูเวลาที่ใช้ในการการประมวลผลของแต่ละจำนวน โดยทำการทดสอบการประมวลผลทั้งหมด 5 ครั้ง ผลการทดสอบตามตาราง ดังนี้

ตารางที่ 2 ผลทดสอบสมรรถนะการประมวลผลอัลกอริทึมการซื้อขายหลักทรัพย์

	Orders (n)			
blockSizes/Threads	**500000**	**1000000**	**1500000**	**2000000**
Time#1 (ms)				
Computation on CPU				
1	5.61	28.64	71.60	134.29
Computation on CPU+GPU				
64	5.65	41.62	95.21	169.87
128	5.82	23.84	54.21	96.73
256	3.69	14.98	34.30	60.21
512	3.43	13.85	31.62	56.16
1024	3.41	13.73	31.41	56.66
2048	3.44	13.82	31.29	56.15
4096	3.41	13.73	31.61	56.18
8192	3.43	13.85	31.61	56.42
16384	3.44	13.81	31.61	56.10
32768	3.46	13.93	32.01	57.08
Time#2 (ms)				
Computation on CPU				
1	5.70	28.74	71.82	134.52
Computation on CPU+GPU				
64	5.65	41.61	95.21	169.97
128	5.82	23.84	54.36	96.73
256	3.70	14.99	34.20	59.90

ตารางที่ 3 (ต่อ)

	Orders (n)			
blockSizes/Threads	**500000**	**1000000**	**1500000**	**2000000**
Time#1 (ms)				
512	3.41	13.86	31.63	56.16
1024	3.41	13.73	31.25	56.66
2048	3.43	13.99	31.61	56.21
4096	3.41	13.78	31.61	56.12
8192	3.43	13.82	31.60	56.67
16384	3.44	13.85	31.62	56.63
32768	3.45	13.88	32.01	56.47
Time#3 (ms)				
Computation on CPU				
1	5.62	28.59	71.47	133.51
Computation on CPU+GPU				
64	5.63	41.51	94.95	169.54
128	5.83	23.89	54.43	96.89
256	3.72	15.00	34.32	59.97
512	3.42	13.91	31.93	56.21
1024	3.42	13.81	31.46	56.14
2048	3.49	13.93	31.39	56.17
4096	3.42	14.04	31.93	56.13
8192	3.40	13.75	31.93	56.39
16384	3.41	13.89	31.45	56.96
32768	3.46	13.71	31.36	56.43
Time#4 (ms)				
Computation on CPU				
1	5.79	28.85	71.57	133.62

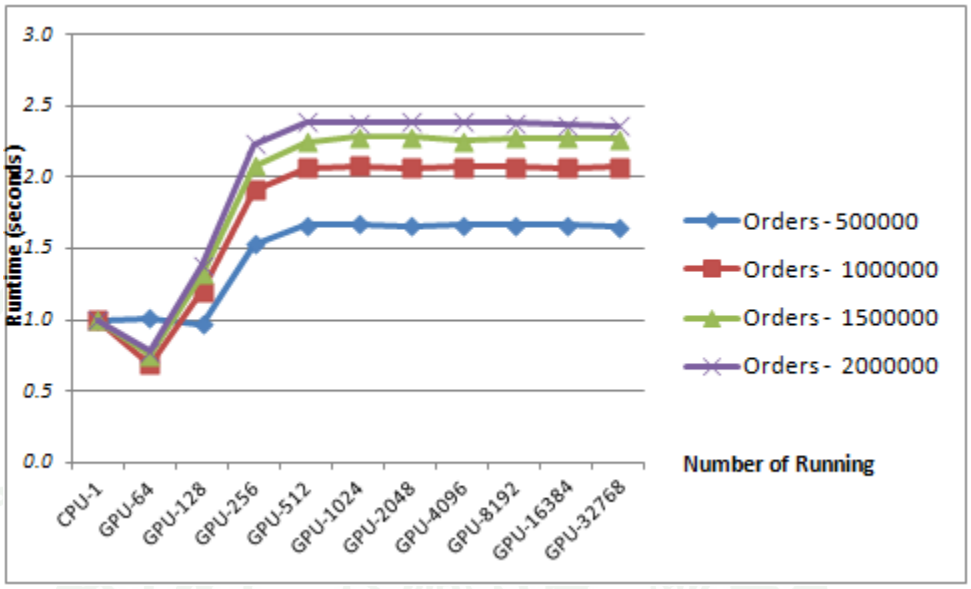
ตารางที่ 4 (ต่อ)

	Orders (n)			
blockSizes/Threads	**500000**	**1000000**	**1500000**	**2000000**
Computation on CPU+GPU				
64	5.64	41.57	94.94	176.28
128	5.94	23.88	54.42	96.85
256	3.75	15.09	34.67	59.94
512	3.42	13.95	32.22	56.13
1024	3.42	13.87	31.38	56.16
2048	3.42	13.94	31.38	56.20
4096	3.42	13.85	31.92	56.14
8192	3.44	13.98	31.30	55.97
16384	3.40	13.98	31.46	56.74
32768	3.46	13.91	31.40	56.99
Time#5 (ms)				
Computation on CPU				
1	5.79	28.90	72.15	134.72
Computation on CPU+GPU				
64	5.64	41.57	94.94	176.28
128	5.94	23.88	54.42	96.85
256	3.75	15.09	34.67	59.94
512	3.42	13.95	32.22	56.13
1024	3.42	13.87	31.38	56.16
2048	3.42	13.94	31.38	56.20
4096	3.42	13.85	31.92	56.14
8192	3.44	13.98	31.30	55.97
16384	3.40	13.98	31.46	56.74
32768	3.46	13.91	31.40	56.99

จากผลทดสอบของตารางที่ 2 เป็นการแสดงเวลาทั้งหมดที่ใช้ในการประมวลผล โดยได้แสดงเวลาที่ใช้ในการประมวลผลตามจำนวนของรายการจาก 500000 1000000 1500000 และ 2000000 ตามลำดับ ในส่วนของการนำหน่วยประมวลผลกราฟิกมาช่วยในการประมวล ได้แสดงเวลาที่ใช้ในการประมวลผลตามจำนวนการทำงานแบบขนานจาก 64 128 256 512 1024 2048 4096 8192 16384 และ 32768 ตามลำดับ จากการแสดงเวลาที่ใช้ในการประมวลผลทั้ง 5 ครั้ง ได้แสดงเวลาการประมวลผลเฉลี่ยดังตารางที่ 3 และอัตราความเร็วที่เพิ่มขึ้นของการประมวลผลดังภาพที่ 10

ตารางที่ 3 ผลทดสอบสมรรถนะการประมวลผลอัลกอริทึมการซื้อขายหลักทรัพย์เฉลี่ย

	Orders (n)			
blockSizes/Threads	**500000**	**1000000**	**1500000**	**2000000**
Average Runtime (ms)				
Computation on CPU				
1	5.70	28.75	71.72	134.13
Computation on CPU+GPU				
64	5.64	41.57	95.05	172.39
128	5.87	23.86	54.37	96.81
256	3.72	15.03	34.43	60.00
512	3.42	13.90	31.92	56.16
1024	3.42	13.80	31.38	56.36
2048	3.44	13.93	31.41	56.19
4096	3.42	13.85	31.80	56.14
8192	3.43	13.87	31.55	56.28
16384	3.42	13.90	31.52	56.64
32768	3.45	13.87	31.63	56.79



ภาพที่ 10 อัตราความเร็วที่เพิ่มขึ้นของการประมวลผลด้วย GPU

จากผลทดสอบการนำประมวลผลกราฟิกมาช่วยในการประมวลผล เพื่อให้เห็นเวลาที่ใช้ในการประมวลผลในแต่ละส่วน จึงแสดงเวลาการถ่ายโอนข้อมูลจากโฮสต์ไปยังหน่วยประมวลผลกราฟิก การประมวลผลคำสั่งจัดเรียงคำสั่งซื้อขาย และการโอนถ่ายข้อมูลจากอุปกรณ์มายังโฮสต์ โดยมีผลการทดสอบทั้งหมด 5 ครั้งตามตาราง 4 และผลทดสอบเฉลี่ยตามตาราง 5 ดังนี้

ตารางที่ 4 ผลทดสอบการประมวลผลด้วยหน่วยประมวลผลกราฟิก

blockSizes/Threads	**Times (ms)**		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
Orders (500000)			
64	0.01068620	0.00005293	5.64023000
128	0.01045300	0.00002408	5.80868000
256	0.01044390	0.00002193	3.68230000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
512	0.01048610	0.00002193	3.42291000
1024	0.01046710	0.00002217	3.39833000
2048	0.01048400	0.00002098	3.42492000
4096	0.01048210	0.00002098	3.40008000
8192	0.01046900	0.00002217	3.42098000
16384	0.01045010	0.00002289	3.43085000
32768	0.01044300	0.00002313	3.44730000
Orders (1000000)			
64	0.02016590	0.00002313	41.59630000
128	0.02014710	0.00002289	23.81730000
256	0.02022100	0.00002313	14.96260000
512	0.02014400	0.00002289	13.82630000
1024	0.02016400	0.00002289	13.70580000
2048	0.02018590	0.00002193	13.80360000
4096	0.02017190	0.00002193	13.70880000
8192	0.02018790	0.00002217	13.82860000
16384	0.02021100	0.00002289	13.78650000
32768	0.02016020	0.00002193	13.91380000
Orders (1500000)			
64	0.02987220	0.00002384	95.17770000
128	0.02984600	0.00002408	54.17510000
256	0.02984500	0.00002408	34.26730000
512	0.02988220	0.00002384	31.58520000
1024	0.02980490	0.00002289	31.38140000
2048	0.02992920	0.00002384	31.26100000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
4096	0.02986500	0.00002480	31.57700000
8192	0.02985910	0.00002503	31.58180000
16384	0.02983590	0.00002408	31.58260000
32768	0.02016020	0.00002193	13.91380000
Orders (2000000)			
64	0.02960400	0.00002718	169.84300000
128	0.04027610	0.00002384	96.68710000
256	0.04024510	0.00002503	60.17250000
512	0.04026990	0.00002503	56.11510000
1024	0.04041100	0.00002408	56.61560000
2048	0.04028610	0.00002408	56.11070000
4096	0.02966020	0.00002599	56.15180000
8192	0.04026790	0.00002503	56.37520000
16384	0.04027300	0.00002503	56.06400000
32768	0.04028300	0.00002503	57.03970000
Time#2 (ms)			
Orders (500000)			
64	0.00809097	0.00005603	5.64013000
128	0.01084800	0.00002193	5.80854000
256	0.01082990	0.00002313	3.68465000
512	0.01086400	0.00002193	3.40239000
1024	0.01088290	0.00002193	3.40128000
2048	0.01086710	0.00002193	3.42056000
4096	0.01089690	0.00002098	3.40142000
8192	0.00795913	0.00002408	3.42210000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
16384	0.01086090	0.00002217	3.43236000
32768	0.01086090	0.00002217	3.43465000
Orders (1000000)			
64	0.01505990	0.00002503	41.59100000
128	0.02091600	0.00002313	23.81980000
256	0.02091000	0.00002313	14.96440000
512	0.02093790	0.00002313	13.84170000
1024	0.02094100	0.00002408	13.70870000
2048	0.02094910	0.00002384	13.96880000
4096	0.02092600	0.00002313	13.75940000
8192	0.01514010	0.00002480	13.80310000
16384	0.02091690	0.00002408	13.82470000
32768	0.02093320	0.00002289	13.85660000
Orders (1500000)			
64	0.02234100	0.00002599	95.18590000
128	0.03106190	0.00002408	54.32570000
256	0.03096010	0.00002289	34.16860000
512	0.03093700	0.00002313	31.59520000
1024	0.03097010	0.00002313	31.22240000
2048	0.03096200	0.00002503	31.58260000
4096	0.03098800	0.00002503	31.57410000
8192	0.02233100	0.00002599	31.57630000
16384	0.03098580	0.00002503	31.58720000
32768	0.03094820	0.00002503	31.98250000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
Orders (2000000)			
64	0.04087210	0.00002599	169.92800000
128	0.04088000	0.00002408	96.68810000
256	0.04087710	0.00002599	59.86260000
512	0.04085490	0.00002503	56.12030000
1024	0.04083900	0.00002384	56.61520000
2048	0.04114320	0.00002480	56.16510000
4096	0.02961800	0.00003195	56.09430000
8192	0.04087590	0.00002599	56.62530000
16384	0.04087500	0.00002503	56.58980000
32768	0.04089590	0.00002503	56.43140000
Time#3 (ms)			
Orders (500000)			
64	0.00796294	0.00005293	5.62619000
128	0.01088000	0.00002193	5.82114000
256	0.01087900	0.00002313	3.71034000
512	0.01082990	0.00002599	3.41241000
1024	0.01083800	0.00002408	3.40831000
2048	0.01097580	0.00002408	3.47868000
4096	0.01085400	0.00002408	3.41234000
8192	0.01086000	0.00002503	3.38871000
16384	0.01084400	0.00002408	3.40287000
32768	0.01103710	0.00002289	3.44911000
Orders (1000000)			
64	0.01488590	0.00002503	41.49630000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
128	0.02094910	0.00002408	23.86610000
256	0.02101110	0.00002193	14.98230000
512	0.02096100	0.00002408	13.89220000
1024	0.02093010	0.00002408	13.79380000
2048	0.02091100	0.00002384	13.91100000
4096	0.02091910	0.00002480	14.01980000
8192	0.02105500	0.00002408	13.72400000
16384	0.02095700	0.00002384	13.86520000
32768	0.02098700	0.00003600	13.68930000
Orders (1500000)			
64	0.02210690	0.00002503	94.92680000
128	0.03114580	0.00002623	54.40090000
256	0.03115610	0.00002384	34.29190000
512	0.03094100	0.00002694	31.89940000
1024	0.03108100	0.00002503	31.42910000
2048	0.03093190	0.00002599	31.35920000
4096	0.03092380	0.00002503	31.89960000
8192	0.03095600	0.00002718	31.89840000
16384	0.03101300	0.00002503	31.42080000
32768	0.02241110	0.00002599	31.33450000
Orders (2000000)			
64	0.02932000	0.00002718	169.50900000
128	0.04129000	0.00002599	96.84680000
256	0.04139590	0.00002599	59.93110000
512	0.04129510	0.00002599	56.16680000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
1024	0.04133700	0.00002718	56.10320000
2048	0.04131700	0.00002790	56.12720000
4096	0.04128190	0.00002599	56.09360000
8192	0.04138180	0.00002599	56.34750000
16384	0.04135700	0.00002599	56.91910000
32768	0.02960210	0.00002599	56.40290000
Time#4 (ms)			
Orders (500000)			
64	0.00802112	0.00005102	5.62721000
128	0.00794482	0.00002313	5.93047000
256	0.01086210	0.00002289	3.73960000
512	0.01090600	0.00002384	3.41333000
1024	0.00794196	0.00002122	3.41270000
2048	0.01100800	0.00002408	3.40752000
4096	0.01085590	0.00002408	3.41240000
8192	0.01088120	0.00002289	3.42885000
16384	0.01138590	0.00002408	3.38865000
32768	0.01086090	0.00002313	3.44476000
Orders (1000000)			
64	0.01506210	0.00002408	41.55140000
128	0.01502920	0.00002384	23.86090000
256	0.02096080	0.00002313	15.07270000
512	0.02096700	0.00002384	13.92670000
1024	0.01507280	0.00002313	13.85750000
2048	0.02263090	0.00002503	13.92060000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
4096	0.02097200	0.00002313	13.83000000
8192	0.02143500	0.00002289	13.95960000
16384	0.02097200	0.00002503	13.95930000
32768	0.02116890	0.00002313	13.89340000
Orders (1500000)			
64	0.02231310	0.00002503	94.91980000
128	0.02290800	0.00002694	54.39570000
256	0.03097200	0.00002503	34.63610000
512	0.03098700	0.00002503	32.18680000
1024	0.02233910	0.00002503	31.35830000
2048	0.03099510	0.00002599	31.34430000
4096	0.03098300	0.00002503	31.88620000
8192	0.03109690	0.00002503	31.26790000
16384	0.03098510	0.00002599	31.42770000
32768	0.03097990	0.00002408	31.36410000
Orders (2000000)			
64	0.02959300	0.00002790	176.24900000
128	0.02968380	0.00002718	96.81790000
256	0.04135490	0.00002718	59.90310000
512	0.02964520	0.00002599	56.09850000
1024	0.02976110	0.00002694	56.13060000
2048	0.04129410	0.00002599	56.15860000
4096	0.04140500	0.00002694	56.09500000
8192	0.04142590	0.00002790	55.92750000
16384	0.04163790	0.00002718	56.70320000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
32768	0.04142620	0.00002694	56.94500000
Time#5 (ms)			
Orders (500000)			
64	0.00808120	0.00005388	5.62667000
128	0.01082610	0.00002289	5.82036000
256	0.01080920	0.00002193	3.74278000
512	0.01101210	0.00002289	3.41302000
1024	0.01083990	0.00002313	3.41380000
2048	0.01092310	0.00002193	3.42145000
4096	0.00783205	0.00002503	3.41162000
8192	0.01081920	0.00002193	3.42314000
16384	0.01081800	0.00002193	3.42641000
32768	0.01083210	0.00002098	3.41398000
Orders (1000000)			
64	0.01487590	0.00002599	41.48960000
128	0.02092310	0.00002289	23.85910000
256	0.02093600	0.00002313	14.97710000
512	0.02093720	0.00002384	13.68790000
1024	0.02108500	0.00002408	14.11840000
2048	0.02099080	0.00002217	13.86760000
4096	0.01491000	0.00002503	13.97570000
8192	0.02096390	0.00002408	13.93900000
16384	0.02093890	0.00002408	13.83430000
32768	0.02094200	0.00002384	13.89380000

ตารางที่ 4 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	**Computation**	**Device -> Host**
Time#1 (ms)			
Orders (1500000)			
64	0.02202010	0.00002694	94.91390000
128	0.03087590	0.00002313	54.26110000
256	0.03091100	0.00002313	34.42590000
512	0.03091120	0.00002384	31.38740000
1024	0.03096200	0.00002384	32.04150000
2048	0.03106210	0.00002384	31.26740000
4096	0.02206590	0.00002503	31.26430000
8192	0.03120900	0.00002790	31.28000000
16384	0.03094600	0.00002503	31.52630000
32768	0.03099700	0.00002384	32.18290000
Orders (2000000)			
64	0.04096200	0.00002480	169.55700000
128	0.04087590	0.00002503	96.80810000
256	0.04090690	0.00002408	60.13980000
512	0.04088500	0.00002503	56.12510000
1024	0.04098890	0.00002503	56.08990000
2048	0.02918480	0.00002813	56.32040000
4096	0.02907200	0.00002813	56.09690000
8192	0.04096290	0.00002503	56.36800000
16384	0.04101010	0.00002480	56.34200000
32768	0.04101200	0.00002503	56.92220000

ตารางที่ 5 ผลทดสอบการประมวลผลด้วยหน่วยประมวลผลกราฟิกเฉลี่ย

blockSizes/Threads	Times (ms)		
	Host -> Device	Computation	Device -> Host
Average Runtime (ms)			
Orders (500000)			
64	0.00856849	0.00005336	5.63208600
128	0.01019038	0.00002279	5.83783800
256	0.01076482	0.00002260	3.71193400
512	0.01081962	0.00002332	3.41281200
1024	0.01019397	0.00002251	3.40688400
2048	0.01085160	0.00002260	3.43062600
4096	0.01018419	0.00002303	3.40757200
8192	0.01019771	0.00002322	3.41675600
16384	0.01087178	0.00002303	3.41622800
32768	0.01080680	0.00002246	3.43796000
Orders (1000000)			
64	0.01600994	0.00002465	41.54492000
128	0.01959290	0.00002337	23.84464000
256	0.02080778	0.00002289	14.99182000
512	0.02078942	0.00002356	13.83496000
1024	0.01963858	0.00002365	13.83684000
2048	0.02113354	0.00002336	13.89432000
4096	0.01957980	0.00002360	13.85874000
8192	0.01975638	0.00002360	13.85086000
16384	0.02079916	0.00002398	13.85400000
32768	0.02083826	0.00002556	13.84938000
Orders (1500000)			
64	0.02373066	0.00002537	95.02482000
128	0.02916752	0.00002489	54.31170000

ตารางที่ 5 (ต่อ)

blockSizes/Threads	Times (ms)		
	Host -> Device	Computation	Device -> Host
Average Runtime (ms)			
256	0.03076884	0.00002379	34.35796000
512	0.03073168	0.00002456	31.73080000
1024	0.02903142	0.00002398	31.48654000
2048	0.03077606	0.00002494	31.36290000
4096	0.02896514	0.00002498	31.64024000
8192	0.02909040	0.00002623	31.52088000
16384	0.03075316	0.00002503	31.50892000
32768	0.02709928	0.00002417	28.15556000
Orders (2000000)			
64	0.03407022	0.00002661	171.01720000
128	0.03860116	0.00002522	96.76960000
256	0.04095598	0.00002565	60.00182000
512	0.03859002	0.00002541	56.12516000
1024	0.03866740	0.00002541	56.31090000
2048	0.03864504	0.00002618	56.17640000
4096	0.03420742	0.00002780	56.10632000
8192	0.04098288	0.00002599	56.32870000
16384	0.04103060	0.00002561	56.52362000
32768	0.03864384	0.00002560	56.74824000

จากผลทดสอบการประมวลผลด้วยหน่วยประมวลผลกราฟิก ได้ทำการแสดงเวลาที่ใช้ในการประมวลผลของแต่ขั้นตอนการประมวลผลบนหน่วยประมวลผลกราฟิกดังนี้ ขั้นตอนถ่ายโอนข้อมูลจากโฮสต์ไปยังอุปกรณ์ ขั้นตอนประมวลผล (ผลลัพธ์ที่ได้จะเก็บไว้ในหน่วยความจำอุปกรณ์) และขั้นตอนโอนถ่ายข้อมูลจากอุปกรณ์มายังโฮสต์ จากการทำงานของทั้ง 3 ขั้นตอน ใน

ขั้นตอนการถ่ายโอนข้อมูลมูลใช้เวลานาน ดังนั้นข้อมูลที่มีการถ่ายโอนมีจำนวนมาก ใช้เวลาในการประมวลผลมากขึ้นด้วย

การนำหน่วยประมวลผลกราฟิกมาช่วยประมวลผลการซื้อขายหลักทรัพย์จากข้อมูลจำลองรายการคำสั่งซื้อคำสั่งขาย โดยมีข้อมูลหุ้น 403 หุ้น ข้อมูลจำลองรายการคำสั่งซื้อคำสั่งขายมีทั้งหมด 2000000 รายการ พร้อมจำลองข้อมูลประเภทการซื้อขาย และราคาเสนอซื้อหรือเสนอขาย ซึ่งข้อมูลจำลองดังกล่าวเป็นข้อมูลที่จำเป็นสำหรับการซื้อขายหลักทรัพย์

ผลทดสอบการการประมวลผล ได้แสดงให้เห็นว่าหน่วยประมวลผลกราฟิกสามารถเร่งความเร็วในการประมวลผล เมื่อเทียบกับการประมวลผลด้วยหน่วยประมวลผลการอย่างเดียว อัตราเร่งของความเร็วที่เพิ่มขึ้นนั้นขึ้นอยู่กลับความสามารถในการทำงานแบบขนาน และการออกแบบการเรียกใช้ทรัพยากร การทดสอบได้ทำการรันทั้งมด 5 ครั้ง ผลการทดสอบทั้ง 5 ครั้ง เห็นว่าเวลาที่ใช้ในการประมวลผลแบบขนานขึ้นอยู่กับข้อมูลที่ทำการทดสอบ และการออกแบบการเรียกใช้ทรัพยากรบนหน่วยประมวลผลกราฟิก กล่าวคือ ข้อมูลจำลองมีจำนวนหุ้น 403 หุ้น จากรายการคำสั่งซื้อขายทั้งหมด 200000 รายการ จากข้อมูลจำลองนี้สามารถทำงานแบบขนานได้สูงสุด 403 งาน เนื่องจากข้อมูลรายการของแต่ละหุ้นมีความมีความสัมพันธ์กัน ดังนั้นรายการของคำสั่งซื้อคำสั่งขายที่เป็นหุ้นเดียวกันจึงไม่สามารถทำการประมวลผลแบบขนานได้

จากข้อมูลจำลองมีจำนวนการทำงานแบบขนานจำกัด ดังนั้นการออกแบบการเรียกใช้ทรัพยากรของหน่วยประมวลผลกราฟิกจึงมีจำกัด จากผลทดสอบได้ทำการออกแบบการเรียกใช้ทรัพยากรของหน่วยประมวลผลกราฟิกออกเป็น 10 ส่วน คือ 64 128 256 512 1024 2048 4096 8192 16384 และ 32768 ตามลำดับ ผลการออกแบบการประมวลผลด้วยหน่วยประมวลผลกราฟิกที่มีจำนวนการทำงานแบบขนานน้อยกว่าจำนวนหุ้น หรือจำนวนการทำงานแบบขนานที่ข้อมูลสามารถประมวลผลได้ในเวลาเดียวกัน ใช้เวลาในการประมวลผลมากกว่าการประมวลผลด้วยหน่วยประมวลผลกลางอย่างเดียว ในทางเดียวกันการออกแบบให้หน่วยประมวลกราฟิกสามารถประมวลผลแบบขนานได้มากกว่ากว่าจำนวนงานที่สามารถประมวลผลแบบขนานได้ ก็ไม่สามารถเร่งความเร็วในการประมวลผลเพิ่มขึ้น ความเร็วที่เพิ่มขึ้นสูงสุดจะอยู่ในช่วงการทำงานแบบขนานที่มีจำนวนงาน และจำนวนการทำงานแบบขนานของหน่วยประมวลผลที่ใกล้เคียงกัน ดังนั้นการเร่งความเร็วในการประมวลผลอัลกอริทึมการซื้อขายหลักทรัพย์จึงขึ้นอยู่จำนวนหุ้นที่สามารถทำงานแบบขนานได้ในเวลาเดียวกัน และการออกแบบการใช้ทรัพยากรของหน่วยประมวลผลกราฟิก

สรุปและข้อเสนอแนะ

งานวิจัยนี้ได้ทำการศึกษาการนำหน่วยประมวลผลกราฟิกมาช่วยในการประมวลผลงานทั่วไปที่นอกเหนือจากงานด้านกราฟิก โดยทำการทดสอบการประมวลผลอัลกอริทึมของการซื้อขายหลักทรัพย์ การทดสอบการประมวลผลนี้ถูกแบ่งการประมวลผลออกเป็น 2 วิธี คือ 1) ประมวลผลด้วยหน่วยประมวลผลกลางอย่างเดียว 2) นำหน่วยประมวลผลมาช่วยในการประมวลผล เพื่อทำการศึกษาสมรรถนะในการประมวลผลของทั้ง 2 วิธี พร้อมศึกษาปัจจัยที่ส่งผลต่อสมรรถนะการประมวลผล

ผลการทดสอบแสดงให้เห็นว่าหน่วยประมวลผลกราฟิกสามารถนำมาช่วยเร่งความเร็วในการประมวลผลได้ ซึ่งอัตราส่วนของความเร็วที่เพิ่มขึ้นในการประมวลผลขึ้นอยู่กับลักษณะของข้อมูล ขีดความสามารถของหน่วยประมวลผลกราฟิก และการออกแบบเรียกใช้ทรัพยากร

จากการทำงานวิจัยนี้ สามารถนำหน่วยประมวลผลกราฟิกไปปรับใช้ในการประมวลผลงานทั่วไปที่นอกเหนือจากงานด้านกราฟิกได้ โดยงานของงานทั่วไปมีการทำงานแบบขนาน ใช้เวลาในการประมวลผลนาน และปริมาณข้อมูลในการถ่ายโอนน้อย นอกจากนี้นักพัฒนาระบบควรมีความรู้ความเข้าใจการทำงานของหน่วยประมลผลกราฟิก เพื่อทำการออกแบบการทำงาน และการเรียกใช้ทรัพยากรที่มีอยู่ให้เกิดประโยชน์สูงสุด

ส่วนของงานที่ควรปรับปรุง คือ ความสามารถของการเร่งความเร็วในการประมวลผลที่เพิ่มขึ้น ข้อมูลที่ใช้ในการทดสอบมีจำนวนของการทำงานแบบขนานที่จำกัด เนื่องจากทำการประมวลผลแบบขนานตามจำนวนหุ้น ซึ่งข้อมูลที่นำมาทดสอบนี้มีจำนวนหุ้นเพียง 403 หุ้นเท่านั้น ดังนั้นการประมวลผลคำสั่งสามารถทำงานพร้อมกัน 403 งานในเวลาเดียวกันเท่านั้น จึงทำให้ไม่สามารถใช้ขีดความสามารถของหน่วยประมวลผลกราฟิกได้เต็มที่ พร้อมทั้งข้อมูลในการถ่ายโอนมีปริมาณมาก จึงส่งผลให้ใช้เวลาในการถ่ายโอนข้อมูลสูง นอกจากนี้ยังอาจปรับอัลกอริทึมให้เรียกใช้ทรัพยากรที่มีอยู่ให้มีประสิทธิภาพมากขึ้น

เอกสารและสิ่งอ้างอิง

R. Halfhill, 2008. "PARALLEL PROCESSING WITH CUDA." Microprocessor Report, www.MPRonline.com.

Nickolls, Buck, Garland, and Skadron, 2008. "Scalable Parallel Programming with CUDA." ACM: 40-53.

Chen, Qin, Xie, Zhao, and Heng ,2009. "AN EFFICIENT SORTING ALGORITHM WITH CUDA." Journal of the Chinese Institute of Engineers: 915-921.

Cascarano, Rolando, Risso, and Sisto, 2010. "iNFAnt: NFA pattern matching on GPGPU devices." ACM: 20-26.

Mohan, 2010. "Faster file matching using GPGPUs." SAAHPC 2010: 13-15.

Wood, 2010. "Faster file matApplications of GPUs in Computational Finance." A thesis submitted for the degree of MSc. Grid Computing.

Dang, 2011. "Modeling multi-factor financial derivatives by a Partial Differential Equation approach with efficient implementation on Graphics Processing Units." A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy Graduate Department of Computer Science University of Toronto.

Grauer-Gray, Killian, Searles, Cavazos, 2013. "Accelerating Financial Applications on the GPU." ACM: 127-136.

ภาคผนวก

ภาคผนวก

การติดตั้ง CUDA

การติดตั้ง CUDA

1. Driver กราฟิกการ์ดที่ทำมารองรับ CUDA จะต้องลงไดร์เวอร์ที่รองรับด้วยเช่นกัน โดยทำการดาวน์โหลดไดร์เวอร์ ของ NVIDIA ตัวล่าสุด จากนั้นทำการติดตั้ง

2. C Compiler CUDA เป็นภาษาที่ใช้ภาษาซีเป็นพื้นฐาน ดังนั้นในเครื่องจะต้องมีคอมไพเลอร์ภาษา C ไว้อยู่ด้วย สำหรับเครื่องที่เป็น Linux มักจะมี gcc มาให้อยู่แล้ว แต่ถ้าไม่มีให้ทำการติดตั้ง gcc ก่อน ส่วนเครื่องที่เป็น Windows ต้องใช้คอมไพลเอร์ภาษาซีของ Microsoft เท่านั้น โดยจะต้องลง Microsoft Visual Studio 2005, 2008 หรือจะเป็น Microsoft Visual C++ Express เป็นต้น

3. CUDA Toolkit เป็นเครื่องมือเอาไว้พัฒนา ประกอบไปด้วย CUDA Compiler, library, debugger และ profiler สามารถทำการดาวน์โหลด Toolkit ได้ที่ CUDA Downloads

4. GPU Computing SDK เป็นตัวรวมตัวอย่างโค้ดพื้นฐานของ CUDA ในส่วนนี้จะลงหรือไม่ลงก็ได้ เอาไว้แสดงโค้ดตัวอย่างของการเขียนโปรแกรมด้วย CUDA

เพื่อเป็นการทดสอบว่าติดตั้งทั้งหมดเรียบร้อยแล้ว ทำการทดลองการทำงานของเครื่องมือก่อนอื่นให้ลองเปิด Command Prompt (เข้าที่ Run แล้วพิมพ์ cmd) ขึ้นมาสำหรับ Windows หรือเปิด Terminal สำหรับเครื่องที่เป็น Linux แล้วทดสอบว่าคอมไพเลอร์ของ CUDA ใช้งานได้ด้วยการพิมพ์ nvcc --version เพื่อแสดงข้อมูลเวอร์ชันของคอมไพเลอร์ CUDA

ประวัติการศึกษาและการทำงาน

ชื่อ	นางสาวเกษรินทร์ รุ่งเรือง
เกิดวันที่	17 เมษายน 2527
สถานที่เกิด	อำเภอบรบือ จังหวัดมหาสารคาม
ประวัติการศึกษา	วท.บ. (วิทยาการคอมพิวเตอร์) มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ตำแหน่งปัจจุบัน	นักวิเคราะห์ระบบ
สถานที่ทำงานปัจจุบัน	บริษัท เอไอเอ