ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

ปริญญา

วิศวกรรมคอมพิวเตอร์ วิศวกรรมคอมพิวเตอร์

สาขา **ภาควิชา**

เรื่อง การสร้างแบบจำลองสถานการณ์ไม่ต่อเนื่องให้เหมาะสมโดยใช้กลุ่มประมวลผลกราฟิก

Discrete – Event Simulation Optimizer on a GPU Cluster

นามผู้วิจัย นางสาวกีรติญา ศรีมูล

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(ผู้ช่วยศาสตราจารย์จุฑา พิชิตลำเค็ญ, Ph.D.)

หัวหน้าภาควิชา .....

(ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

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

วิทยานิพนธ์

เรื่อง

การสร้างแบบจำลองสถานการณ์ไม่ต่อเนื่องให้เหมาะสมโดยใช้กลุ่มประมวลผลกราฟิก

Discrete – Event Simulation Optimizer on a GPU Cluster

โดย

นางสาวกีรติญา ศรีมูล

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)
พ.ศ. 2555

กีรติญา ศรีมูล 2555: การสร้างแบบจำลองสถานการณ์ไม่ต่อเนื่องให้เหมาะสมโดยใช้กลุ่มประมวลผลกราฟิก ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D. 109 หน้า

วิทยานิพนธ์นี้เสนอแนวคิดการออกแบบการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการคำนวณหาผลลัพธ์ให้กับแบบจำลองสถานการณ์ไม่ต่อเนื่อง แบบจำลองสถานการณ์การจัดการธุรกิจ อาทิเช่น ปัญหาการจัดการระบบคลังสินค้า, ปัญหาการจัดเส้นทางยานพาหนะการรับ/ส่งสินค้า เป็นต้น มีกระบวนคิดที่ซับซ้อนในการคำนวณหาคำตอบที่ดีที่สุดให้กับแบบจำลองจึงใช้เวลาการคำนวณจำนวนมากเมื่อมีการคำนวณปัญหาที่มีขนาดใหญ่

ดังนั้นในงานวิจัยนี้ทำการทดสอบการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการจำลองสถานการณ์ระบบคลังสินค้า จากผลการทดสอบสามารถเพิ่มสมรรถนะความเร็วในการประมวลผลได้ถึง 4 เท่า ผู้วิจัยทำจึงศึกษาปัจจัยการเพิ่มสมรรถนะความเร็วในการประมวลผลบนหน่วยประมวลผลกราฟิกจากการผลทดสอบดังกล่าว เพื่อพัฒนาการโปรแกรมแบบขนานบนกลุ่มประมวลผลกราฟิกกับการคำนวณหาผลลัพธ์ให้แบบสถานการณ์การจัดเส้นทางยานพาหนะขนาดต่างๆ และจากผลการทดสอบพบว่า สามารถเร่งความเร็วในการประมวลผลได้ถึง 44 เท่า

/ /

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Geeratiya Srimool 2012: Discrete – Event Simulation Optimizer on a GPU Cluster. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Assistant Professor Putchong Uthayopas, Ph.D. 109 pages.

This thesis presents the concept of parallel programming on GPU cluster design for calculating the solution for discrete-event simulation. The simulation of business management such as, the inventory problem, the transportation problem. This simulation has the complex methodology to compute the best solution. As the consequent, it takes very long computation time when there is the large problem size.

So, this study experiment the parallel computing on GPU for solving inventory problem simulation. From the experiment, we can increase the speedup 4 times. So, we study the factors that affect the increasing speedup for parallel on GPU from the result. In order to develop parallel programming on GPU cluster for calculating the solution for transportation problem simulation. From the observation, we can increase the speedup 44 times.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลงได้ ข้าพเจ้าขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ภุชงค์ อุทโยภาศ ประธานกรรมการที่ปรึกษา ผู้ให้แนวทางการวิจัย และข้อเสนอแนะรวมถึงวิถีทางในการดำเนินชีวิตหลักปฏิบัติการเป็นนักวิจัยที่ดีมีศีลธรรม และผู้ช่วยศาสตราจารย์ ดร.จุฑา พิชิตลำเค็ญ ที่ให้คำแนะนำชี้แนะถึงแนวทาง และองค์ความรู้ที่เกี่ยวข้องกับงานวิจัย

ข้าพเจ้าขอขอบคุณเพื่อนๆ นิสิตปริญญาโทจากหลายสถาบัน สมาชิกกลุ่ม HPCNC ที่ให้คำปรึกษา และมอบกำลังใจในการศึกษางานวิจัยนี้ อีกทั้งเจ้าหน้าที่ธุรการภาควิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตบางเขน ทุกท่าน สำหรับให้ความช่วยเหลือด้านการประสานงาน และดำเนินการเอกสารต่างๆ ขอบคุณงานวิจัยไร้พรมแดนที่ทำให้ข้าพเจ้าได้เปิดโลกทัศน์ศึกษาผลงานวิจัยของท่านอื่นๆ สุดท้ายนี้ขอบคุณความห่วงใยที่ครอบครัวของข้าพเจ้ามีให้กันเสมอมา

กีรติญา ศรีมูล

มีนาคม 2555

สารบัญ

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญตาราง (ต่อ)

สารบัญตาราง (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

ภาพผนวกที่ **หน้า**

การสร้างแบบจำลองสถานการณ์ไม่ต่อเนื่องให้เหมาะสมโดยใช้กลุ่มประมวลผลกราฟิก

Discrete – Event Simulation Optimizer on a GPU Cluster

คำนำ

การบริหารจัดการระบบธุรกิจมีความสำคัญต่อการดำเนินงานต่างๆภายในองค์กร แต่ละองค์กรมีความซับซ้อนในการตัดสินใจการดำเนินงานที่ไม่เหมือนกัน ขึ้นกับขนาดขององค์กร และปัจจัยต่างๆที่ส่งผลต่อเป้าหมายของแต่ละองค์กร การตัดสินใจโดยอาศัยความชำนาญอาจเกิดความเสี่ยงต่อผลกำไรของธุรกิจที่มีขนาดใหญ่ เนื่องจากการดำเนินการตัดสินใจนั้นมีมากกว่าสองทางเลือก ดังนั้นจึงมีการสร้างโมเดลทางธุรกิจ เพื่อศึกษาข้อมูลเชิงปริมาณที่สามารถเปรียบเทียบเชิงตัวเลขช่วยให้ผู้ประกอบการมองเห็นแนวทางที่ดีที่สุดของการตัดสินใจที่ส่งผลต่อกำไรสูงสุดขององค์กรได้ การสร้างโมเดลนั้นเป็นการนำข้อมูลเชิงสถิติมาประมวลผลโดยใช้สมการคณิตศาสตร์คำนวณผลกำไรจากทางเลือกในการตัดสินใจที่เป็นไปได้ ซึ่งกระบวนการแปลงข้อมูลของปัญหาทางธุรกิจให้เป็นสมการทางคณิตศาสตร์จึงมีความสำคัญอย่างยิ่ง ดังนั้นจึงประยุกต์นำคอมพิวเตอร์จำลองสถานการณ์ของระบบธุรกิจ โดยทำการสุ่มตัวเลขขึ้นมาตามข้อมูลสถิติเพื่อคาดคะเนความน่าจะเป็นของสิ่งที่จะเกิดขึ้นกับระบบ ข้อมูลที่ได้จากการจำลองสถานการณ์ของระบบจะนำไปคำนวณหาผลกำไรตามสมการคณิตศาสตร์ใช้ประกอบการพิจารณา การตัดสินใจหลายๆทางเลือกของผู้ประกอบการได้

เพื่อการวิเคราะห์แนวทางการตัดสินใจของผู้บริหาร คอมพิวเตอร์จึงถูกใช้ในการคำนวณสร้างแบบจำลองสถานการณ์ของการบริหารจัดการอุตสาหกรรมในระบบโลจิสติกส์ (Logistic Management) เช่น การจำลองระบบคลังสินค้า (Inventory problem) คำนวณการตัดสินใจการรับจำนวนสินค้าเข้ามากักตุน และขายไป, การจำลองเส้นทางการขนส่งของยานพาหนะ (Transportation problem) ที่ต้องเดินทางไปยังจุดรับ/ส่งสินค้าหลายจุดโดยมีเงื่อนไขของเวลาคำนวณหาการจำนวนยานพาหนะที่น้อยที่สุดที่ให้บริหารทุกการร้องขอ และใช้ระยะเวลาเดินทางที่สั้นที่สุด เป็นต้น

การประมวลผลการโปรแกรมแบบลำดับสำหรับการจำลองสถานการณ์ของระบบงานที่มีขนาดใหญ่นั้น ใช้พลังการประมวลผลที่สูงขึ้นตามความซับซ้อน และเงื่อนไขของปัจจัยที่มากขึ้นของระบบงาน ซึ่งทำให้ใช้เวลาคำนวณการหาผลลัพธ์จำนวนมาก ดังนั้นเพื่อเพิ่มประสิทธิภาพเวลาการคำนวณ จึงนำแนวคิดการประมวลผลแบบขนานมาทำการปรับประยุกต์การคำนวณแบบจำลองสถานการณ์ โดยคำนึงถึงการใช้ประโยชน์ตามความสามารถของทรัพยากรบนเครื่องคอมพิวเตอร์นั้นๆ ซึ่งในวิทยานิพนธ์นี้เล็งเห็นความสามารถการประมวลผลทางภาพกราฟิกบนหน้าจอคอมพิวเตอร์ของหน่วยประมวลผลกราฟิก(Graphic Processing Unit: GPU) หรือจีพียู เนื่องจากสถาปัตยกรรมของหน่วยประมวลผลกราฟิก ประกอบด้วย หน่วยประมวลผล (Core) หรือคอร์จำนวนมาก สามารถรองรับการคำนวณแบบขนาน จึงมีแนวคิดออกแบบการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิก สำหรับการจำลองสถานการณ์

งานวิจัยนี้เสนอการออกแบบการคำนวณแบบขนานสำหรับการค้นหาคำตอบที่เหมาะสมที่สุดของการจำลองสถานการณ์ไม่ต่อเนื่องบนระบบจีพียูคลัสเตอร์ (GPU Cluster) เพื่อศึกษาขีดจำกัดการเพิ่มสมรรถนะความเร็วการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์

วัตถุประสงค์

เสนอแนวคิดการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์ สำหรับการจำลองสถานการณ์ไม่ต่อเนื่อง เพื่อเร่งสมรรถนะการคำนวณให้สูงขึ้น และศึกษาปัจจัยที่มีผลกระทบต่อการเร่งสมรรถนะการประมวลผลบนหน่วยประมวลผลกราฟิก

ขอบเขตการวิจัย

1. เสนอแนวคิดการแก้ปัญหาการจำลองสถานการณ์ไม่ต่อเนื่อง ให้รวดเร็วโดยใช้เทคโนโลยีจีพีจีพียู และคลัสเตอร์

2. พัฒนาอัลกอริทึมที่ใช้หาผลลัพธ์ของปัญหาแบบจำลองสถานการณ์ไม่ต่อเนื่อง ให้สามารถประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก และทำงานร่วมกับโปรแกรมไมโครซอฟท์เอ็กเซลได้

3. ทดสอบประสิทธิภาพ และประเมินผล

การตรวจเอกสาร

ความรู้พื้นฐาน

งานวิจัยนี้ออกแบบอัลกอริทึมการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก เพื่อนำไปปรับประยุกต์ใช้ศึกษาวิธีการเร่งสมรรถนะความเร็วการคำนวณหาผลลัพธ์ของแบบจำลองสถานการณ์ต่างๆในภาคธุรกิจ เช่น ปัญหาการสั่งสินค้าเข้ามากักตุนในระบบคลังสินค้า, ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด เป็นต้น โดยศึกษาแนวคิด และความรู้พื้นฐานดังต่อไปนี้

การสร้างแบบจำลองสถานการณ์

ศึกษาวิธีการสร้างแบบจำลองสถานการณ์ โดยการนำเงื่อนไขข้อจำกัดของปัญหาในทางปฏิบัติมาทำการวิเคราะห์ทางคณิตศาสตร์ และสรุปผลออกมาเป็นตัวเลขเพื่อใช้ในการตัดสินใจ วิธีการดังกล่าวเรียกว่า “การวิจัยดำเนินการ (Operations Research: OR)” (Hamdy and Taha, 2003) ถูกนำมาประยุกต์ใช้ในงานบริหารจัดการ เพื่อให้ผู้บริหารที่มีอำนาจในการตัดสินใจสามารถมองเห็นความแตกต่างของทางเลือกทั้งหมดที่นำไปสู่ผลประโยชน์สูงสุดขององค์กร ซึ่งการสร้างแบบจำลองสถานการณ์ ต้องมีองค์ประกอบที่สำคัญของ การวิจัยดำเนินการ ดังนี้

1. การระบุปัญหา, วัตถุประสงค์ และข้อจำกัดที่ไม่สามารถหลีกเลี่ยงได้

2. การพัฒนาแบบจำลองคณิตศาสตร์ เพื่อให้ได้ผลลัพธ์ของคำตอบที่ดีที่สุดออกมา

3. การกำหนดตัวแปรที่ต้องใช้ในการตัดสินใจ

การจำลองสถานการณ์ไม่ต่อเนื่อง (Discrete event simulation) เป็นการสร้างแบบจำลองทางคณิตศาสตร์ เพื่อจำลองระบบงานต่างๆ ที่มีการเปลี่ยนแปลงสถานะของระบบ ณ จุดเวลาต่างๆ ที่อยู่ในช่วงเวลาที่เราสนใจอย่างไม่ต่อเนื่อง ซึ่งในงานวิจัยนี้ศึกษาปัจจัยการเร่งสมรรถนะความเร็วการประมวลผลบนหน่วยประมวลผลกราฟิก จากการจำลองสถานการณ์ระบบคลังสินค้า และจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด

การจำลองสถานการณ์ของระบบคลังสินค้า (Silver *et al*., 1998) (Petruzzi *et al.*, 1999) เมื่อมีเหตุการณ์เกิดขึ้นกับระบบคลังสินค้า ข้อมูลที่ได้จากการดำเนินงานของระบบจะถูกจัดเก็บรวบรวมเป็นค่าข้อมูลสังเกตทางสถิติ เพื่อใช้วิเคราะห์ประกอบการตัดสินใจได้ ดังเช่น บริษัทที่มีระบบคลังสินค้าขนาดใหญ่ ต้องมีการตัดสินใจเกี่ยวกับการรับจำนวนสินค้าเข้ามากักตุน และขายไป ถ้าช่วงเวลาหนึ่งบริษัทมีการกักสินค้าไว้มากอาจส่งผลกระทบต่อผลกำไรที่บริษัทควรจะได้รับ แต่เนื่องบริษัทไม่อาจทราบได้ถึงจำนวนลูกค้าที่เข้ามาซื้อสินค้าล่วงหน้าได้ ทำให้บริษัทต้องมีการวางแผนการบริหารจัดการระบบคลังสินค้าอย่างรอบครอบ จึงมีการประยุกต์ใช้เครื่องคอมพิวเตอร์สร้างแบบจำลองสถานการณ์ของระบบคลังสินค้า เพื่อทำการคาดคะเนเหตุการณ์ที่ส่งผลกระทบกับระบบคลังสินค้าล่วงหน้า และหาผลลัพธ์ที่เหมาะสมที่สุดให้กับแบบจำลองของระบบ ซึ่งปัญหาคลังสินค้าที่มีขนาดใหญ่ย่อมมีความซับซ้อนของเงื่อนไขจำนวนมาก ส่งผลต่อเวลาที่ใช้คำนวณหาผลลัพธ์ให้กับแบบจำลองสถานการณ์ของระบบคลังสินค้านั้นมีจำนวนมาก

การจำลองสถานการณ์การจัดเส้นทางยานพาหนะ (Savelsbergh and Sol, 1995) เป็นส่วนสำคัญในการจัดการระบบโลจิสติกส์ เป็นการวางแนวทางการเคลื่อนย้าย, การจัดเก็บ และกระจายสินค้า เพื่อจัดส่งสินค้าจากต้นทางไปยังผู้บริโภคปลายทางได้อย่างทันเวลาโดยใช้ต้นทุนต่ำสุด การขยายตัวของเศรษฐกิจที่รวดเร็วทำให้การบริหารจัดการขนส่งในระบบโลจิสติกส์มีความซับซ้อนมากขึ้น ซึ่งการวางแผนการจัดเส้นทางการขนส่งสินค้าของยานพาหนะโดยอาศัยความชำนาญอาจก่อให้เกิดความเสี่ยงที่ส่งผลกระทบต่อกำไรที่ควรจะได้รับ จึงประยุกต์ใช้เครื่องคอมพิวเตอร์จำลองเส้นทางการเดินรถ เพื่อวิเคราะห์ และจัดเส้นทางการเดินรถทั้งหมดที่เป็นไปได้ โดยอยู่ในเงื่อนไขของเวลา และค่าใช้จ่ายต่ำสุด

ปัญหาการจัดเส้นทางยานพาหนะ (Pickup and Delivery problem with time windows: PDPTW) คือ ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด ซึ่งมีสำคัญในการบริหารจัดการระบบโลจิสติกส์ วัตถุประสงค์การแก้ปัญหา PDPTW คือ การคำนวณหาจำนวนการใช้ยานพาหนะที่น้อยที่สุดที่สามารถให้บริการทุกจุดรับ/ส่งสินค้าที่มีการร้องขอ และแสดงเส้นทางการเดินรถทุกคันที่ใช้รับ/ส่งสินค้าในระบบ ซึ่งคำตอบที่ได้ต้องอยู่ในเงื่อนไขจำนวนมาก เช่น ช่วงเวลาเปิด/ปิดทำการในแต่ละจุดรับ/ส่งสินค้า, ประมาณสินค้าที่ต้องการโหลดขึ้นยานพาหนะในแต่ละจุดรับ/ส่งสินค้าที่ไม่เท่ากัน, ความสามารถในการบรรทุกสินค้าของแต่ละยานพาหนะ และปริมาณเชื้อเพลิงที่จำกัดที่ของยานพาหนะที่สามารถใช้ได้ตลอดเส้นทางการเดินรถ เป็นต้น

ที่ผ่านมามีงานวิจัยที่เสนออัลกอริทึมการค้นหาคำตอบที่ดีที่สุดสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะ ดังเช่น ปี 2001 กลุ่มวิจัยของ Li and Lim เสนอการค้นหาคำตอบด้วยวิธีเมตาฮิวริสติก (Metaheuristic Algorithm) โดยทำการทดสอบกับปัญหาการจัดเส้นทางยาพาหนะที่มีจุดรับ/ส่งหลายจุด ซึ่งคำตอบที่ได้ต้องอยู่ในฟังก์ชั่นวัตถุประสงค์ 2 ข้อ คือ 1)จำนวนยานพาหนะน้อยที่สุด และ2)ระยะทางของยานพาหนะแต่ละคันต้องสั้นที่สุด งานวิจัยนี้แสดงคำตอบที่ดีที่สุดในกรณี 106 จุดรับ/ส่งสินค้า, ปี 2002 กลุ่มวิจัย Lao and Liang เสนอวิธีสำหรับการแก้ปัญหาการจัดเส้นทางยาพาหนะที่มีจุดรับ/ส่งหลายจุด โดยการประยุกต์วิธีการฮิวริสติก (Heuristic Algorithm) และวิธีการแทรก (Insertion Algorithm) สำหรับการคำนวณหาคำตอบเริ่มต้น จากนั้นทำการปรับปรุงค่าคำตอบด้วยวิธีการทาบู (Taboo Search Algorithm: TS)

อัลกอริทึมที่ใช้เพื่อคำนวณหาคำตอบที่ดีที่สุดนั้น ย่อมต้องการพลังในการคำนวณที่สูง เนื่องจากมีกระบวนการค้นหา และเลือกคำตอบที่ซับซ้อนมากขึ้น ดังนั้นในงานวิจัยนี้จึงศึกษาผลงานวิจัยอื่นๆที่มีการออกแบบการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกมาคำนวณงานที่มีลักษณะใกล้เคียงกับการค้นหาคำตอบของการจำลองสถานการณ์การจัดเส้นทางยานพาหนะ ดังเช่น ปี 2008 กลุ่มวิจัย Adam, Wladyslaw และMaciej เสนอวิธีการปรับประยุกต์อัลกอริทึมการค้นหาแบบทาบูบนหน่วยประมวลผลกราฟิก สำหรับการแก้ปัญหาคนเดินขายของ (The Traveling Salesman Problem หรือ TSP) ด้วยวิธีการทาบูซึ่งเป็นวิธีการค้นหาคำตอบแบบเมตาฮิวริสติก โดยคำตอบที่ได้จากวิธีการทาบูนี้สามารถนำไปใช้ได้จริงในระบบอุตสาหกรรมโลจิสติกส์ เนื่องจากการค้นหาแบบทาบูมีวิธีการปรับปรุงคำตอบให้ดีขึ้น โดยการค้นหาคำตอบข้างเคียงจากคำตอบที่ดีที่สุด ณ ขณะนั้น เมื่อคำตอบข้างเคียงให้ค่าผลลัพธ์ที่ดีกว่าคำตอบเดิมจะทำการปรับปรุงคำตอบข้างเคียงนั้นให้เป็นคำตอบที่ดีที่สุด ซึ่งเงื่อนไขการค้นหาคำตอบข้างเคียงจะไม่ค้นหากลับไปยังคำตอบเดิม เป็นวิธีการค้นหาแบบมีข้อห้าม หรือวิธีการค้นหาทาบู จากผลการทดสอบสามารถเร่งความเร็วได้ถึง 16 เท่า และในปี 2011 กลุ่มวิจัย Shigeyoshi เสนอวิธีการพัฒนาอัลกอริทึมแบบอาณาจักรมด (Ant colony optimization หรือ ACO) สำหรับการแก้ปัญหาการแก้สมการกำลังสอง (Quadratic Assignment Problems หรือ QAPS) โดยการปรับประยุกต์อัลกอริทึมการค้นหาแบบทาบู และอัลกอริทึมแบบอาณาจักรมดบนหน่วยประมวลผลกราฟิก อัลกอริทึมแบบอาณาจักรมดเป็นอัลกอริทึมเชิงสุ่มใช้โมเดลCombinatorial Optimization (CO) สำหรับการเลือกเส้นทางที่ให้ค่าที่ดีที่สุด ซึ่งสามารถนำไปปรับประยุกต์กับการแก้ปัญหาคนเดินขายของได้

นอกจากนี้ยังมีผลงานวิจัยอื่นๆที่ออกแบบการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก เพื่อทำการคำนวณแบบจำลองสถานการณ์อื่นๆ ดังเช่น ในปี 2006 งานวิจัยของ Kalyan ทำการศึกษาอัลกอริทึมที่สามารถค้นหาคำตอบของปัญหาการจำลองสถานการณ์ไม่ต่อเนื่อง โดยทำการปรับประยุกต์อัลกอริทึมให้มีการทำงานแบบขนานบนหน่วยประมวลผลกราฟิก เพื่อศึกษาประสิทธิการการเร่งสมรรถนะเทียบกับการคำนวณแบบลำดับโดยหน่วยประมวลผลกลาง จากผลการทดสอบพบว่าการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิกนั้นเหมาะสมกับปัญหาขนาดใหญ่ กลุ่มวิจัยของ Kalyan จึงมีแนวคิดที่จะพัฒนาการประมวลผลแบบ Hybrid และปรับแก้อัลกอริทึมให้สามารถประมวลผลแบบขนานอย่างมีประสิทธิภาพบนหน่วยประมวลผลกราฟิก เพื่อเร่งสมรรถนะการประมวลผลให้เร็วขึ้นกว่าเดิม, ปี 2008 งานวิจัยของ Hong และ Linda ทำการจำลองสถานการณ์การตอบสนองของโมเลกุลกลุ่มเล็กๆในระบบชีวะโดยใช้อัลกอริทึม Gillespies's Stochastic Simulation (SSA) ซึ่งใช้เวลาประมวลผลที่ยาวนานเมื่อคำนวณกับระบบชีววิทยาขนาดใหญ่ กลุ่มวิจัยจึงมีแนวคิดออกแบบการคำนวณแบบขนานบนระบบคลัสเตอร์ (Parallel Computing on Cluster) เพื่อเร่งเวลาในการประมวลผลให้เร็วขึ้น ซึ่งการเขียนโปรแกรมแบบบน FPGA เป็นเรื่องที่ยุ่งยากสำหรับการพัฒนาการออกแบบการโปรแกรมแบบขนาน และราคาที่สูงของ FPGA จึงทำให้กลุ่มวิจัยสนใจพัฒนาการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกขนานแทน เนื่องจากการโปรแกรมบนหน่วยประมวลผลกราฟิกใช้ CUDA ซึ่งเป็นเทคโนโลยีการโปรแกรมแบบขนานที่ถูกพัฒนาต่อยอดจากภาษา C สามารถเรียนรู้ง่าย และยังมีราคาที่ถูกกว่า FPGA จากผลการทดลองแสดงให้เห็นถึงสมรรถนะความเร็วจากการประมวลผลบนหน่วยประมวลผลกราฟิกเพิ่มขึ้น 20 เท่า เทียบกับการโปรมแกรมแบบลำดับซึ่งใช้เวลาการคำนวณถึง 6 เดือน และปี 2008 Mark ศึกษาค้นคว้างานวิจัยการจำลองกลศาสตร์ของไหล ซึ่งได้ออกแบบอัลกอริทึม Spatial data structure on particles (SPH) ให้มีการประมวลผลบนหน่วยประมวลผลกราฟิก เพื่อคำนวณตำแหน่งอนุภาค, ความหนาแน่น, การชน และแรงที่ทำให้ความเร็วของอนุภาคเปลี่ยนไป จากผลการทดลองสามารถอธิบายหลักการเพิ่มประสิทธิภาพในการประมวลผลบนหน่วยประมวลผลกราฟิก โดยเน้นการจัดการเทรด (Thread) บนหน่วยประมวลผลกราฟิก

การประมวลผลแบบขนาน

แนวโน้มการประมวลผลแบบขนาน (Parallel Computing) (Kumar *et al*., 1994) ได้รับความนิยมมาก เป็นการใช้เครื่องคอมพิวเตอร์ต่อกันตั้งแต่ 2 เครื่องขึ้นไป หรือเครื่องคอมพิวเตอร์ที่มีจำนวนโปรเซสเซอร์ตั้งแต่ 2 โปรเซสเซอร์ขึ้นไป ช่วยกันคำนวณงานที่มีขนาดใหญ่ ด้วยการแบ่งจำนวนงานไปประมวลผลยังโปรเซสเซอร์หลายๆตัว เพื่อช่วยลดเวลาการคำนวณให้เร็วขึ้นกว่าการคำนวณบนโปรเซสเซอร์เดียว ซึ่งสถาปัตยกรรมการประมวลผลแบบขนานของโปรเซสเซอร์ แบ่งออกเป็น 4 ประเภท คือ

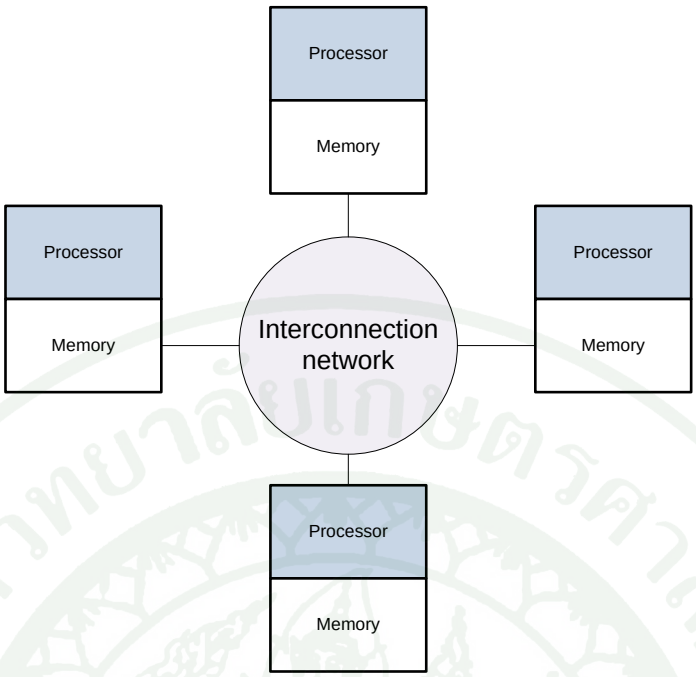
1. Single Instruction stream, Single Data stream หรือ SISD คือ การประมวลผลคำสั่งการทำงานเดียวกัน ซึ่งใช้ข้อมูลชุดเดียวกัน

2. Single Instruction stream, Multiple Data stream หรือ SIMD คือ การประมวลผลคำสั่งการทำงานเดียวกัน ซึ่งใช้ข้อมูลคนละชุด

3. Multiple Instruction stream, Single Data stream หรือ MISD คือ การประมวลผลคำสั่งการทำงานคนละชุดคำสั่ง แต่ใช้ข้อมูลชุดเดียวกัน

4. Multiple Instruction stream, Multiple Data stream หรือ MIMD คือ การประมวลผลคำสั่งการทำงานคนละชุดคำสั่ง และใช้ข้อมูลคนละชุด

Message Passing Interface หรือ MPI (Barney, 2007) เป็นมาตรฐานการพัฒนาการโปรแกรมแบบขนานแบบส่งข้อความผ่านเครือข่ายเน็ตเวิร์คภายในกลุ่มของเครื่องคอมพิวเตอร์ซึ่งทำงานเสมือนคอมพิวเตอร์ 1 เครื่อง เรียกว่า “คลัสเตอร์คอมพิวเตอร์ (Computer Cluster)” (Baker, 2000) โครงสร้างของคลัสเตอร์คอมพิวเตอร์ ประกอบด้วย กลุ่มเครื่องคอมพิวเตอร์ส่วนบุคคล, ระบบสื่อสารข้อมูลระหว่างเครื่องคอมพิวเตอร์ความเร็วสูง และซอฟต์แวร์จัดการประสานการทำงานระหว่างเครื่องคอมพิวเตอร์ในระบบคลัสเตอร์ ซึ่งการทำงานของแต่ละเครื่องคอมพิวเตอร์ส่วนบุคคลนั้นเป็นอิสระต่อกัน แสดงดังภาพที่ 1



ภาพที่ 1 โครงสร้างระบบคลัสเตอร์

การทดสอบประสิทธิภาพการคำนวณแบบขนาน ได้จากการคำนวณ Speedup (Quinn, 2003) คือ การคำนวณอัตราส่วนระหว่างเวลาที่ใช้ประมวลผลของโปรแกรมแบบลำดับ และเวลาที่ใช้ในการโปรแกรมแบบขนาน ดังสมการที่ 1 Speedup เป็นข้อมูลเชิงตัวเลขสามารถนำไปวิเคราะห์เพื่อทำการปรับปรุงการออกแบบการโปรแกรมแบบขนานได้

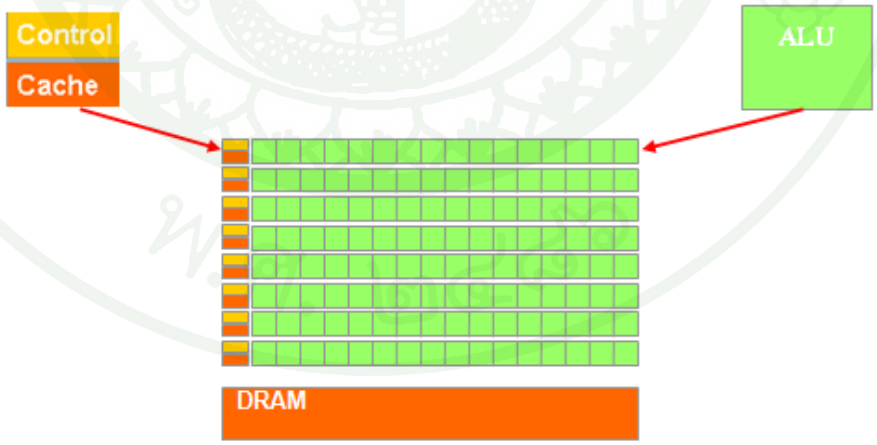
$$Speedup = \frac{T_{Sequential}}{T_{Parallel}} \quad (1)$$

$T_{Sequential}$ คือ เวลารวมการประมวลผลของโปรแกรมแบบลำดับ

$T_{Parallel}$ คือ เวลารวมการประมวลผลของโปรแกรมแบบขนาน

การประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก

สาเหตุที่หน่วยประมวลผลกราฟิกสามารถเร่งความเร็วในการประมวลผลนั้น เริ่มมาจากการศึกษากฎของมัวร์ (Moore's Law) (Bill Dally, 2009) ซึ่งอธิบายถึง "จำนวนทรานซิสเตอร์ที่บรรจุอยู่บนชิพจะมีจำนวนเพิ่มขึ้นเป็นเท่าตัวในทุกๆ2ปี" ทำให้หน่วยประมวลผลกลาง (Central Processing Unit) มีความเร็วในการประมวลผลเพิ่มสูงขึ้น จึงเกิดการพัฒนาเพื่อเพิ่มสมรรถนะในการประมวลผลให้เร็วขึ้นมากกว่าเดิมโดยการนำหน่วยประมวลผลกลางตั้งแต่ 2 ตัวขึ้นไปมาทำงานร่วมกัน ที่เรียกว่า " มัลติคอร์ (Multicore) " เช่นเดียวกับหน่วยประมวลผลกราฟิก (Gray, 2009) ซึ่งเป็นเทคโนโลยีมัลติคอร์บนหน่วยประมวลผลกราฟิก มีจำนวนคอร์มากเมื่อเทียบกับหน่วยประมวลผลกลาง จึงมีแนวคิดการนำหน่วยประมวลผลกราฟิกมาประมวลผลนอกเหนือจากงานประมวลผลทางด้านกราฟิกเพียงอย่างเดียว เช่น คำนวณแก้สมการอนุพันธ์, การสร้างแบบจำลอง เป็นต้น ทำให้เกิดเทคโนโลยีใหม่ที่มีชื่อว่า "General-purpose computing on graphics units Technology หรือ GPGPU" ซึ่งมีการประมวลผลแบบขนานในรูปแบบ SIMD (Single Instruction Multiple Data) (Enhua and Youquan, 2008) คือ ข้อมูลที่ใช้ประมวลผลบนแบบขนานบนหน่วยประมวลผลกราฟิกนั้น มีลักษณะกลุ่มข้อมูลคนละชุดกันที่จะต้องประมวลผลโดยใช้คำสั่งการทำงานเดียวกัน ผู้ใช้สามารถเขียนโปรแกรมควบคุมการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกได้ เพื่อเพิ่มสมรรถนะความเร็วในการการประมวลผลได้ แสดงดังภาพที่ 2

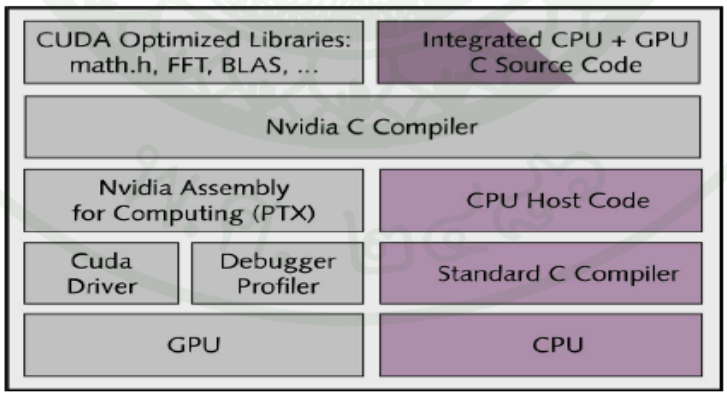


ภาพที่ 2 สถาปัตยกรรมของหน่วยประมวลผลกราฟิก

ที่มา: NVIDIA CUDA C Programming Guide (2010)

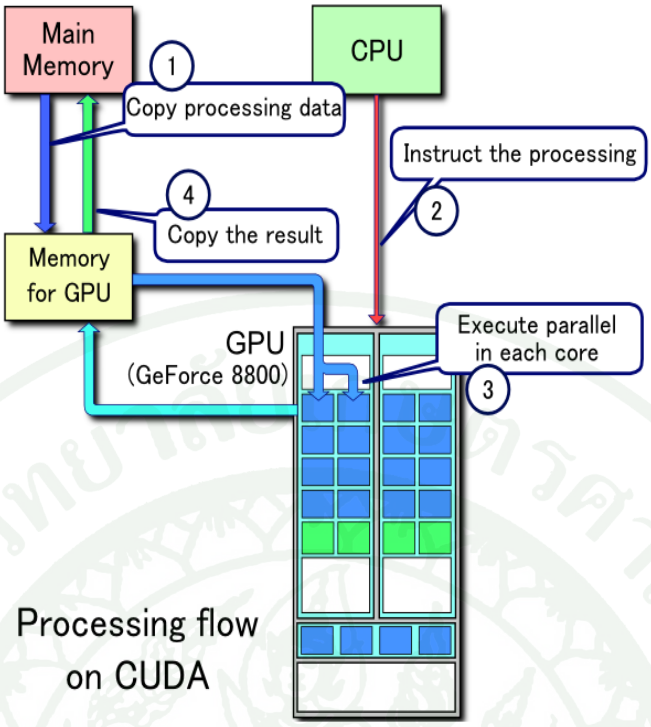
ในอดีตการอนุญาตให้ผู้ใช้สามารถพัฒนาโปรแกรมบนหน่วยประมวลผลกราฟิกนั้นเป็นเรื่องยาก บริษัท Nvidia จึงได้พัฒนา SDK ที่เรียกว่า CUDA (Compute Unified Device Architecture) (Tom, 2008) ซึ่งเป็นสถาปัตยกรรมที่สร้างขึ้นตามแนวคิดเทคโนโลยี GPGPU โดยประกอบด้วยหน่วยประมวลผลกราฟิก,โปรแกรมมิ่งโมเดล และตัวคอมไพเลอร์ เพื่อที่จะสามารถเขียนโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกได้ ซึ่งหน่วยประมวลผลกราฟิกที่สามารถรองรับการทำงานของ CUDA คือ ผลิตภัณฑ์ของบริษัทNVIDIA รุ่น GEFORE SERIES 8 ขึ้นไปเท่านั้น และภาษาโปรแกรมที่ใช้ในการเขียนโปรแกรมบนหน่วยประมวลผลกราฟิกร่วมกับ CUDA คือ C, C++, FORTRAN ,MATLAB และJAVA เป็นต้น

การทำงานของ CUDA (Pharr and Fernando, 2005) เป็นการทำงานร่วมกันระหว่างหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิกจึงแบ่งการทำงานออกเป็น 2ส่วนโดยผู้ใช้สามารถเขียนส่วนการประมวลผลบนหน่วยประมวลผลกลาง และ หน่วยประมวลผลกราฟิกได้ในงานเดียวกัน โดยคอมไพล์ตัวแปลภาษา Nvidia C จะจัดการแยกชุดคำสั่งออกเป็น 2ส่วน คือ1) Nvidia Assembly (PTX) จัดการชุดคำสั่งของหน่วยประมวลผลกราฟิก ซึ่งชุดคำสั่งนี้จะเรียกใช้ CUDA Driver และ Debugger Profiler เพื่อให้สามารถประมวลผลบนหน่วยประมวลผลกราฟิกได้ และ 2) ชุดคำสั่งของหน่วยประมวลผลกลาง ใช้ตัวแปลภาษา C เป็นตัวจัดการ เพื่อให้สามารถประมวลผลบนหน่วยประมวลผลกลางได้ดังภาพที่ 3 ซึ่งขั้นตอนการทำงานร่วมกันระหว่างหน่วยประมวลผลกลางและหน่วยประมวลผลกราฟิกแสดงดังภาพที่ 4



ภาพที่ 3 การทำงานของ CUDA

ที่มา: Tom (2008)



ภาพที่ 4 ขั้นตอนการทำงานร่วมกันระหว่าง CPU และ GPU

ที่มา: NVIDIA (2009)

ขั้นตอนการทำงานร่วมกันระหว่าง CPU และGPU

1. ทำการคัดลอกข้อมูลจากหน่วยความจำหลัก ไปยังหน่วยความจำบนหน่วยประมวลผลกราฟิก ผ่านทางบัส PCI-Express

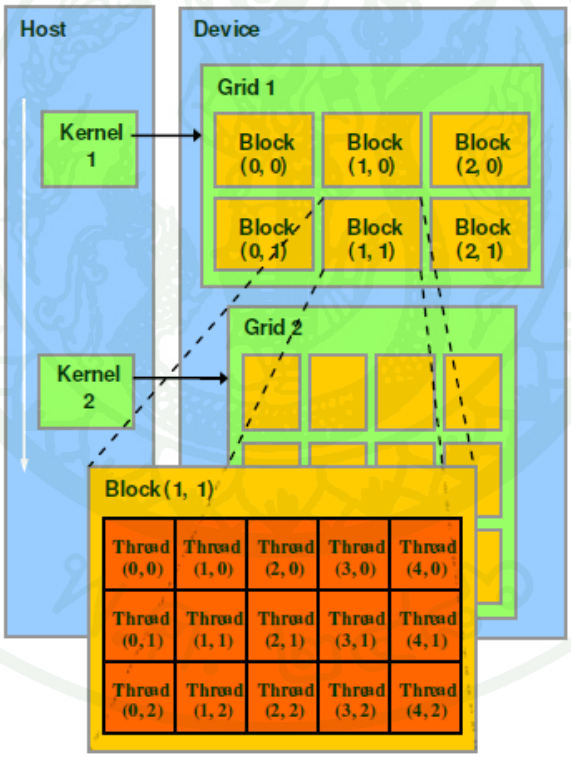
2. หน่วยประมวลผลกลางส่งคำสั่งไปยังหน่วยประมวลผลกราฟิก

3. เมื่อหน่วยประมวลผลกราฟิกได้รับคำสั่งแล้ว จึงประมวลผลแบบขนานในแต่ละคอร์

4. เมื่อการประมวลผลเสร็จสิ้นแล้ว ทำการคัดลอกผลลัพธ์จากหน่วยความจำบนหน่วยประมวลผลกราฟิกไปยังหน่วยความจำหลัก

สถาปัตยกรรมหน่วยความจำบนหน่วยประมวลผลกราฟิก หรืออุปกรณ์ (Device) ประกอบด้วย หน่วยความจำแบบโกลบอล (Global Memory), หน่วยความจำแบบค่าคงที่ (Constant Memory) และหน่วยความจำแบบแทคเจอร์ (Texture Memory) ซึ่งหน่วยประมวลผลกลาง หรือโฮสต์ (Host) สามารถอ่าน/เขียนข้อมูลผ่านหน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิกได้ หน่วยความจำแบบโกลบอลเป็นหน่วยความจำที่สำคัญใช้สำหรับการเชื่อมต่อข้อมูลระหว่างหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิก

ลักษณะการประมวลผลบนหน่วยประมวลผลกราฟิกนั้น เนื่องจากข้อมูลอินพุตแต่ละตัวนั้นมีรูปแบบเหมือนกัน และเป็นอิสระต่อกัน จึงสามารถทำการประมวลผลแบบขนานได้ มีเคอร์เนล (Kernel) เป็นส่วนเรียกการประมวลผลบนหน่วยประมวลผลกราฟิกซึ่งประกอบด้วยอาร์เรย์ของบล็อก (Blocks) ที่เรียกว่า “กริด (Grid)” แสดงดังภาพที่ 5

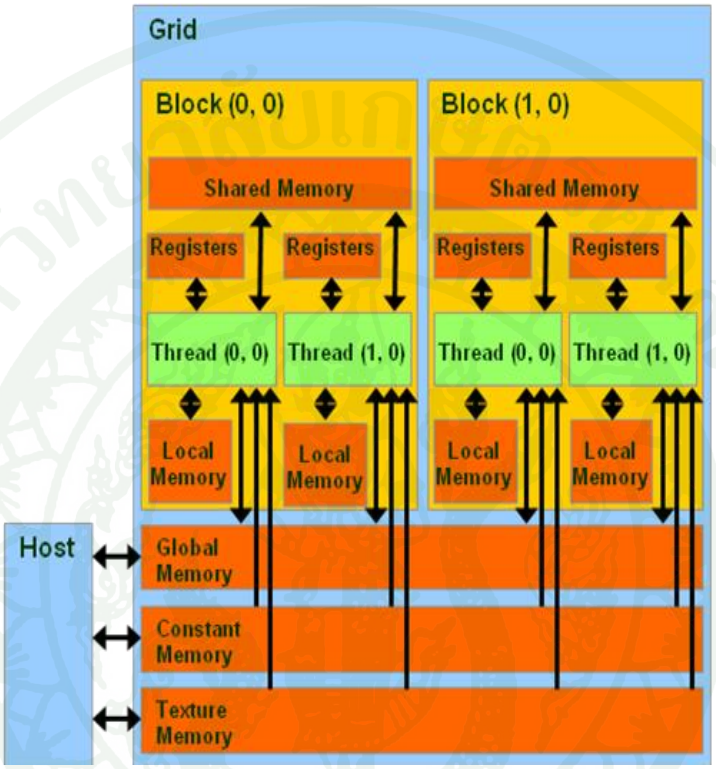


ภาพที่ 5 การเรียกการทำงานบนหน่วยประมวลผลกราฟิก

ที่มา: NVIDIA CUDA C Programming Guide (2010)

แต่ละบล็อกบนกริดนั้นประกอบด้วยเทรดจำนวนมาก แต่ละจีพียูเทรดสามารถอ่าน/เขียนข้อมูลลงรีจิสเตอร์ (Register), หน่วยความจำแบบโลคอล (Local Memory) และหน่วยความจำแบบ

โกลบอล บนหน่วยประมวลผลกราฟิกได้โดยตรง แต่ไม่สามารถเขียนข้อมูลลงหน่วยความจำแบบค่าคงที่ และหน่วยความจำแบบแทคเจอร์บนหน่วยประมวลผลกราฟิกได้ จีพียูเทรดที่อยู่ในบล็อกเดียวกันนั้นสามารถแลกเปลี่ยนข้อมูลโดยผ่านหน่วยความจำแบบแชร์ (Shared Memory) บนหน่วยประมวลผลกราฟิกได้ แสดงดังภาพที่ 6



ภาพที่ 6 สถาปัตยกรรมหน่วยความจำบนหน่วยประมวลผลกราฟิก

ที่มา: NVIDIA CUDA C Programming Guide (2010)

การประยุกต์ใช้เทคโนโลยี GPGPU กับงานที่ต้องการพลังการคำนวณสูง การเขียนโปรแกรมบนหน่วยประมวลผลกราฟิกให้สามารถประมวลผลได้อย่างมีประสิทธิภาพนั้น ควรคำนึงถึงการจัดการใช้หน่วยความจำบนหน่วยประมวลผลกราฟิก เนื่องจากหน่วยความจำแบบโกลบอล และหน่วยความจำแบบโลคอลเป็นหน่วยความจำประเภทพลวัต ทำให้ใช้เวลามากในการเข้าถึงข้อมูลซึ่งมีผลต่อการเร่งสมรรถนะในการประมวลผลเมื่อเทียบกับการเข้าถึงข้อมูลผ่านหน่วยความจำแบบแชร์บนหน่วยประมวลผลกราฟิก

การทำงานแบบเทรดบนหน่วยประมวลผลกลาง และหน่วยประมวลผลกราฟิกนั้น มีความแตกต่างกันอย่างมาก เทรดบนหน่วยประมวลผลกราฟิกจะมีขนาดเล็กกว่าเทรดบนหน่วยประมวลผลกลางมาก ทำให้การสร้างเทรดบนหน่วยประมวลผลกราฟิกใช้จำนวนรอบในการสร้างที่น้อย และถูกจัดการได้ง่ายกว่าเทรดบนหน่วยประมวลผลกลางที่ใช้หลายพันรอบในการสร้าง และจัดการ ดังนั้นการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกจึงควรคำนึงถึงการจัดการเทรด, ขนาดของหน่วยความจำบนประมวลผลกราฟิก และการจัดการใช้หน่วยความจำบนหน่วยประมวลผลกราฟิกของแต่ละเทรดด้วย ซึ่งแนวคิดการเขียนโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก สามารถศึกษาได้จากภาคผนวก ก.

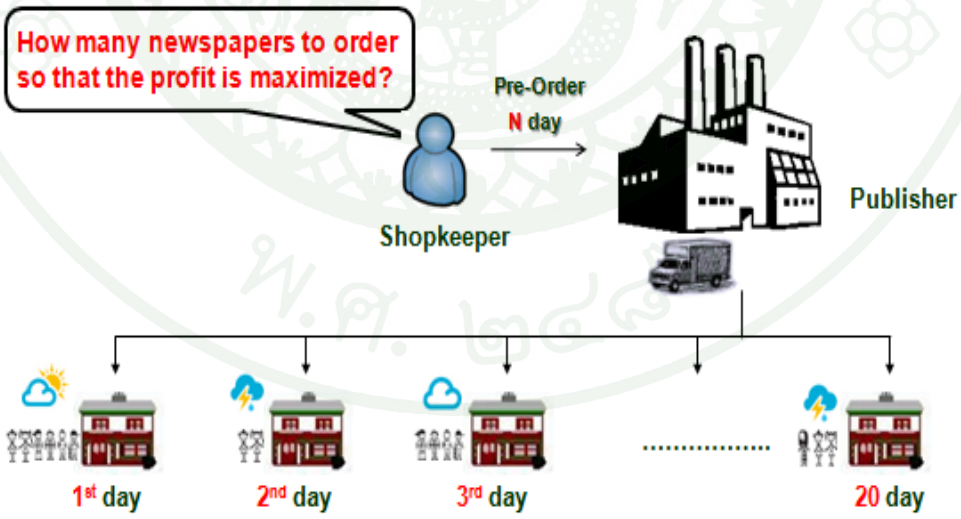
จากการศึกษาสถาปัตยกรรม และการทำงานบนหน่วยประมวลผลกราฟิก ทำให้ในงานวิจัยนี้มีแนวคิดพัฒนาอัลกอริทึมสำหรับการแก้ปัญหาการจำลองสถานการณ์ไม่ต่อเนื่องให้สามารถทำงานแบบขนานบนหน่วยประมวลกราฟิก และออกแบบการคำนวณแบบขนานตามแนวคิดของ Message Passing บนระบบจีพียูคลัสเตอร์ (GPU Cluster) เพื่อเพิ่มสมรรถนะการเร่งความเร็วการประมวลผลอย่างเต็มประสิทธิภาพ

อุปกรณ์และวิธีการ

แบบจำลองสถานการณ์ของระบบคลังสินค้า

ออกแบบการคำนวณการหาผลลัพธ์สำหรับแบบจำลองสถานการณ์ของระบบคลังสินค้า โดยให้สามารถประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกได้ เพื่อศึกษาวิธีการเร่งสมรรถนะความเร็วการประมวลผลบนหน่วยประมวลผลกราฟิก ได้ทำการทดสอบกับเครื่องคอมพิวเตอร์ชนิดพกพา ซึ่งประกอบด้วย หน่วยประมวลผลกลางรุ่น Core 2 Duo ซึ่งมีอัตราการทำงานด้วยความถี่ 2.26 GHz, หน่วยความจำหลัก 3 GByte และใช้หน่วยประมวลผลกราฟิกรุ่น Nvidia GeForce 9400 M ซึ่งประกอบด้วย 16 คอร์ แต่ละคอร์มีอัตราการทำงานด้วยความถี่ 1.10 GHz และมีหน่วยความจำแบบโกลบอล เท่ากับ 256 MByte

ในงานวิจัยนี้เลือกศึกษาปัญหา News Dealer ซึ่งเป็นหนึ่งในปัญหาระบบคลังสินค้า เป็นปัญหาการตัดสินใจการสั่งซื้อจำนวนหนังสือพิมพ์ล่วงหน้า 1 เดือนจากโรงพิมพ์เพื่อมาขายต่อให้กับผู้ซื้อรายย่อย ซึ่งการสั่งซื้อจำนวนหนังสือพิมพ์ล่วงหน้ามีส่งผลต่อการคำนวณหาผลกำไรขาดทุน แสดงดังภาพที่ 7



ภาพที่ 7 ปัญหา News Dealer

ปัญหาในลักษณะนี้เหมาะแก่การสร้างแบบจำลองสถานการณ์โดยการสุ่มความต้องการหนังสือพิมพ์ของผู้ซื้อรายย่อยแต่ละวันใน 1 เดือนข้างหน้าจากโมเดลสถิติ เพื่อคำนวณหาจำนวนสั่งซื้อหนังสือพิมพ์ล่วงหน้าแล้วให้ค่าเฉลี่ยผลกำไรสูงสุด

รูปแบบอินพุตของปัญหา News Dealer แสดงดังภาพที่ 8 โดยมีรายละเอียดดังนี้

Type of Newsday หมายถึง สถานการณ์ของยอดขายใน1วัน แบ่งออกเป็น ยอดขายดี (Good), ยอดขายปกติ (Fair) และยอดขายแย่ (Poor)

Probability for Type of Newsday หมายถึง โอกาสความน่าเป็นการเกิดสถานการณ์ของยอดขายใน 1 วัน

Purchase Price หมายถึง ราคาซื้อ/ฉบับ ที่สั่งซื้อหนังสือพิมพ์ล่วงหน้าจากโรงพิมพ์ (ราคาต้นทุนของหนังสือพิมพ์ 1 ฉบับ)

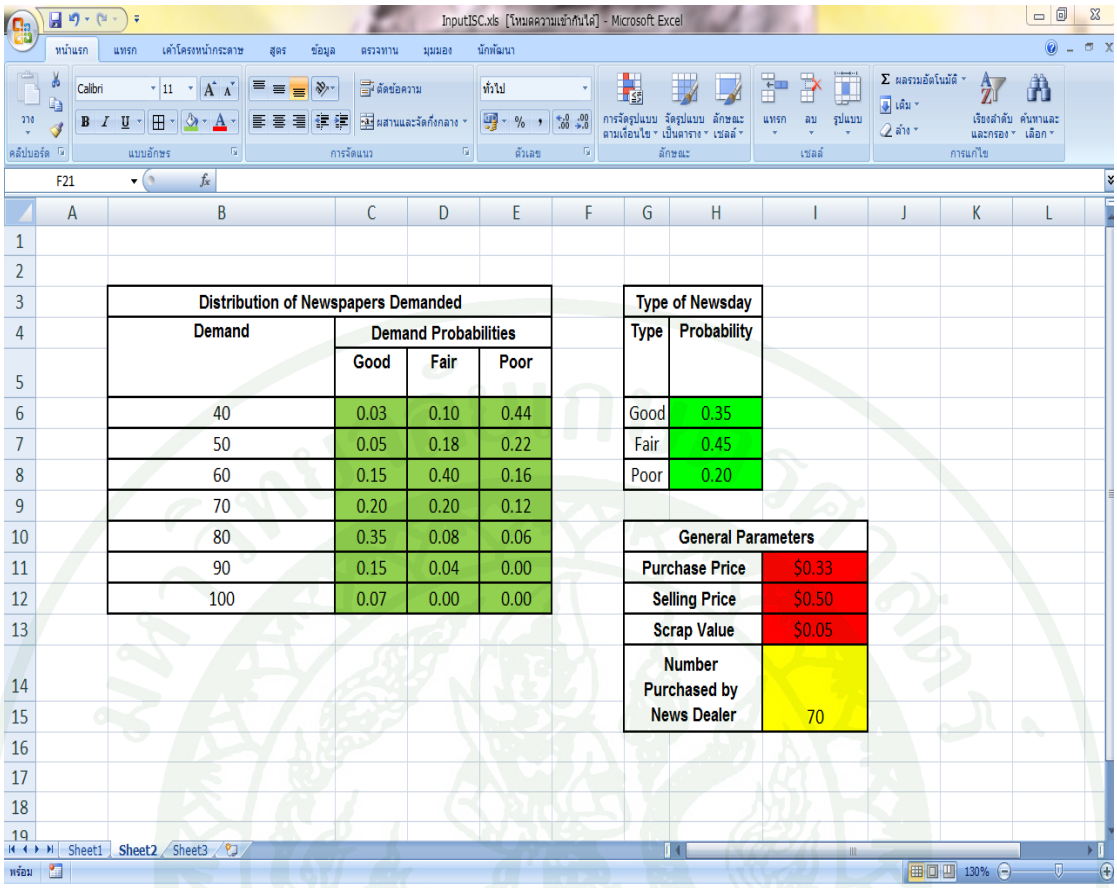
Selling Price หมายถึง ราคาขาย/ฉบับ ซึ่งเป็นราคาที่นำไปขายต่อให้ผู้ซื้อรายย่อย

Scrap Value หมายถึง ราคาขายคืน/ฉบับ ซึ่งเป็นราคาที่สั่งหนังสือพิมพ์มาเกินความต้องการของผู้ซื้อรายย่อย จึงต้องทำการขายคืนให้โรงพิมพ์ในราคาที่ขายต่ำกว่าราคาต้นทุนที่รับมา

Number Purchased by News Dealer หมายถึง จำนวนสั่งซื้อหนังสือพิมพ์ล่วงหน้าจากโรงพิมพ์ ซึ่งเป็นส่วนแสดงผลลัพธ์ที่ได้จากการจำลองสถานการณ์

Distribution of Newspaper Demand หมายถึง ตารางแจกแจงโอกาสความน่าเป็นของจำนวนความต้องการของผู้ซื้อรายย่อย (Demand) ซึ่งสัมพันธ์กับการเกิดสถานการณ์ของยอดขายใน 1 วัน

Demand หมายถึง จำนวนความต้องการของผู้ซื้อรายย่อยภายใน1วัน



Distribution of Newspapers Demanded			
Demand	Demand Probabilities		
	Good	Fair	Poor
40	0.03	0.10	0.44
50	0.05	0.18	0.22
60	0.15	0.40	0.16
70	0.20	0.20	0.12
80	0.35	0.08	0.06
90	0.15	0.04	0.00
100	0.07	0.00	0.00

Type of Newsday	
Type	Probability
Good	0.35
Fair	0.45
Poor	0.20

General Parameters	
Purchase Price	$0.33
Selling Price	$0.50
Scrap Value	$0.05
Number Purchased by News Dealer	70

ภาพที่ 8 ตัวอย่างอินพุตของปัญหา News Dealer

วัตถุประสงค์การแก้ปัญหา News Dealer

1. คำนวณหาจำนวนการสั่งหนังสือพิมพ์ล่วงหน้า 1 เดือน ซึ่งให้ค่ากำไรสูงสุด

เงื่อนไขบังคับของการแก้ปัญหา News Dealer

1. จำนวนการสั่งหนังสือพิมพ์ล่วงหน้าอยู่ในช่วงของจำนวนความต้องการของผู้ซื้อรายย่อย

2. จำนวนการสั่งหนังสือพิมพ์ล่วงหน้า 1 เดือน ต้องมีค่ามากกว่า 0

การหาคำตอบที่ดีที่สุดของแบบจำลอง เริ่มต้นจากการหาผลกำไรเฉลี่ยของแต่ละคำตอบที่เป็นไปได้ จากนั้นเลือกคำตอบที่ดีที่สุดเป็นคำตอบให้แก่แบบจำลองโดยการคำนวณหาผลกำไร

เฉลี่ยของแต่ละคำตอบที่เป็นไปได้นั้นเป็นการคำนวณหาผลกำไรรวมที่ได้จากการจำลองสถานการณ์ตามจำนวนวันที่ต้องมีการสั่งหนังสือพิมพ์ล่วงหน้า ซึ่งผลกำไรต่อ 1 วันที่ทำการจำลองสถานการณ์นั้นเกิดจากการสุ่มตัวเลข 2ค่าเพื่อจำลองสถานการณ์ “ยอดขาย” และ “จำนวนความต้องการหนังสือพิมพ์ของลูกค้า”นำมาคำนวณกับค่าคำตอบที่เป็นไปได้ คือ “จำนวนสั่งหนังสือพิมพ์ล่วงหน้า” ตามโมเดลการคำนวณหาผลกำไร รายละเอียดการหาคำตอบให้แก่แบบจำลองสถานการณ์ปัญหา News Dealer แสดงดังภาคผนวก ข.

จากขั้นตอนการคำนวณการหาผลลัพธ์ให้กับแบบจำลองสถานการณ์ปัญหา News Dealer สามารถเลือกคำตอบที่เหมาะสมจากการสุ่มตัวเลขโมเดลค่าสถิติเพื่อสร้างสถานการณ์ขึ้นมา เมื่อเขียนโปรแกรมแบบลำดับจะแสดงดังภาพที่ 9

```
Num          # Number of  Possible Solution
 for k ← 1 to Num
{
   Number _ Purchased = Demand[k];
   Total _ Profit = 0;
    for Count ← 1 to Iteration
   {
       for i ← 1 to Day
      {
         Type _ Day = Sim _ Type _ Day();
         Random _ demand = Sim _ Demand(Type _ Day);
         Daily _ Profit = Cal _ Profit(Number _ Purchaesd, Random _ demand,.....);
         Total _ Profit += Daily _ Profit;
      }
   }
   Avg[k] = Total _ Profit / Iteration
}
SelectTheBestSolution();
```

ภาพที่ 9 การโปรแกรมแบบลำดับ สำหรับการจำลองสถานการณ์ของปัญหา News Dealer

เพื่อคำนวณหาคำตอบที่ดีที่สุด จึงมีการกำหนดจำนวนรอบการคำนวณตั้งแต่ 1,000 ถึง 10,000,000 รอบ ให้แก่ตัวแปร Iteration ซึ่งผลการทดสอบเวลาที่ใช้ประมวลผลการโปรแกรมแบบลำดับสำหรับแบบจำลองสถานการณ์ของปัญหา News Dealer แสดงดังตารางที่ 1

ตารางที่ 1 เวลา (วินาที) ในการประมวลผลการโปรแกรมแบบลำดับการจำลองสถานการณ์ของปัญหา News Dealer

	จำนวนพันรอบการคำนวณ (Iterations)				
	1	10	100	1,000	10,000
เวลารวม(วินาที)	0.0214	0.195	1.913	19.111	190.683

จากผลการทดสอบพบว่า ระยะเวลาที่ใช้ในคำนวณมีจำนวนมากขึ้นตามจำนวนรอบการคำนวณที่เพิ่มขึ้นในการหาผลลัพธ์ให้กับการจำลองสถานการณ์ของปัญหา News Dealer ดังนั้นในงานวิจัยจึงได้ปรับอัลกอริทึมให้สามารถประมวลผลบนหน่วยประมวลผลกราฟิก เพื่อเร่งสมรรถนะความเร็วในการประมวลผล และศึกษาปัจจัยการเร่งสมรรถนะการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิก โดยออกแบบให้แต่ละจีพียูเทรดที่มีการทำงานแบบขนานบนหน่วยประมวลผลกราฟิกทำการคำนวณตามขั้นตอนการแก้ปัญหา News Dealer เพื่อคำนวณหาผลกำไร และกำหนดให้ไม่มีการรับ/ส่งข้อมูลระหว่างจีพียูเทรดเพื่อป้องกันการเกิด ปัญหา Data Hazard ดังนั้นเมื่อแต่ละจีพียูเทรดทำการคำนวณเสร็จสิ้นจะทำการเขียนผลลัพธ์ลงหน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิก ซีพียูเทรดทำการคัดลอกข้อมูลจาก หน่วยความจำบนหน่วยประมวลผลกราฟิกไปยังหน่วยความจำหลัก จากนั้นทำการคำนวณหาผลกำไรเฉลี่ย และเลือกคำตอบที่เหมาะสมที่สุดให้กับแบบจำลองสถานการณ์ของปัญหา News Dealer อัลกอริทึมการโปรแกรมแบบขนานสำหรับการแก้ปัญหา News Dealer แสดงได้ดังภาพที่ 10

```
__ global __ void Answer(int dNumber _ Purchased,.....)
{
      int idx = blockIdx.x * blockDim.x + threadIdx.x;

      # Parallel Computing On GPU

      Type _ Day = Sim _ Type _ Day();
      Random _ demand = Sim _ Demand(Type _ Day);
      Daily _ Profit[idx] = Cal _ Profit(Number _ Purchaesd, Random _ demand,.....);

}
int main(void)
{
   Num            # Number of  Possible Solution
    for k ← 1 to Num
   {
      Number _ Purchased = Demand[k];
      .....
      Answer <<< dim Grid, dim Block >>> (Number _ Purchased,.....);
      Avg[k] = Cal _ Total _ Profit();
   }
   SelectTheBestSolution();
   return 0;
}
```

ภาพที่ 10 อัลกอริทึมการโปรแกรมแบบขนานสำหรับการแก้ปัญหา News Dealer

จากการโปรแกรมแบบขนานดังภาพที่ 10 เมื่อซีพียูเทรดดำเนินการประมวลผลมาถึงฟังก์ชั่น Answer <<< dimGrid, dimBlock>>> ( ); ซึ่งเป็นคำสั่งการเรียกเคอร์เนล เพื่อแสดงถึงจุดเริ่มต้นการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิก การทำงานแต่ละจีพียูเทรดจะคำนวณตามขั้นตอนการหาผลกำไรของการจำลองสถานการณ์ที่ได้ระบุไว้ในฟังก์ชัน __global__ เมื่อจีพียูเทรดทั้งหมดประมวลผลเสร็จสิ้นแล้วเป็นหน้าที่ของซีพียูเทรดจะประมวลผลในส่วนของการคำนวณเลือกคำตอบที่ดีที่สุดต่อไป

ในงานวิจัยนี้ได้ทำการศึกษา 3 ปัจจัยหลักที่ส่งผลต่อการเร่งสมรรถนะความเร็วการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก ดังนี้

1. การออกแบบการคำนวณของจีพียูเทรด

1.1 กำหนดให้แต่ละจีพียูเทรดทำการคำนวณผลกำไรในแต่ละวันที่ทำการจำลองสถานการณ์ และกำหนดให้จำนวนจีพียูเทรด/1 บล็อก เท่ากับจำนวน N วันที่ทำการจำลองสถานการณ์ การคำนวณ 1 บล็อกเป็นการคำนวณผลกำไรรวมเท่ากับ 1 รอบการคำนวณ (Iteration) ดังภาพที่ 11 เป็นตัวอย่างการคำนวณผลกำไร 20 วันที่ทำการจำลองสถานการณ์ ดังนั้น 1 บล็อกจึงมีการคำนวณแบบขนานของจีพียูเทรดทั้งสิ้น 20 เทรด เป็นต้น

$Block_i$

Thread0	Thread1	Thread2	……..	Thread17	Thread18	Thread19

ภาพที่ 11 ออกแบบการคำนวณ 1 จีพียูเทรดคำนวณผลกำไร 1 วัน

1.2 กำหนดให้แต่ละจีพียูเทรดทำการคำนวณผลกำไรรวมทั้งหมดจากจำนวน N วันที่ทำการจำลองสถานการณ์ ซึ่งการคำนวณ 1 จีพียูเทรดเท่ากับการคำนวณผลกำไรรวม 1 รอบการคำนวณ และกำหนดให้จำนวนเทรดจีพียูเทรด/1 บล็อก เท่ากับจำนวน N เพื่อใช้เปรียบเทียบสมรรถนะการประมวลผลกับการออกแบบการคำนวณของจีพียูเทรดในรูปแบบที่ 1.1 ภาพที่ 12 เป็นตัวอย่างการคำนวณผลกำไร 20 วันที่ทำการจำลองสถานการณ์ และ 1 บล็อกมีการคำนวณแบบขนานของจีพียูเทรดทั้งสิ้น 20 เทรด แต่ละจีพียูเทรดเป็นการคำนวณผลกำไรรวม 1 รอบการคำนวณ ดังนั้น 1 บล็อกจึงเป็นการคำนวณผลกำไรรวม 20 รอบการคำนวณ เป็นต้น

$Block_i$

Thread0	Thread1	Thread2	……..	Thread17	Thread18	Thread19

ภาพที่ 12 ออกแบบการคำนวณ 1 จีพียูเทรดคำนวณผลกำไรรวม N วัน

2. การสุ่มข้อมูล

การหาคำตอบของแบบจำลองสถานการณ์ปัญหา News Dealer มีขั้นตอนของการสุ่มตัวเลขจำนวนมากเพื่อจำลองสถานการณ์ “ยอดขายในแต่ละวัน” และ “จำนวนความต้องการหนังสือพิมพ์ของลูกค้ารายย่อย” ใช้สำหรับการคำนวณหาผลกำไรให้แบบจำลองสถานการณ์ ซึ่งขณะทำการทดสอบนั้นยังไม่มีชุดคำสั่งการสุ่มตัวเลขบนหน่วยประมวลผลกราฟิก ดังนั้นจึงแบ่งเป็น 2 กรณีศึกษา ดังนี้

2.2 ทำการสุ่มข้อมูลโดยหน่วยประมวลผลกลาง จากนั้นคัดลอกข้อมูลไปยังหน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิก

2.3 ทำการสุ่ม Seed Time โดยหน่วยประมวลผลกลาง จากนั้นคัดลอกข้อมูลไปยังหน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิก แล้วทำการสุ่มข้อมูลด้วยอัลกอริทึมของ Steve and Keith (1998) บนหน่วยประมวลผลกราฟิก ซึ่งกำหนดการใช้การสุ่ม 1 Seed Time/ 1 เทรด

3. การเข้าใช้หน่วยความจำบนหน่วยประมวลผลกราฟิกของแต่ละเทรด

เพื่อศึกษาเวลาประมวลผลที่เทรดใช้เข้าถึงข้อมูลบริเวณหน่วยความจำประเภทต่างๆ บนหน่วยประมวลผลกราฟิก จึงแบ่งเป็น 2 กรณี ดังนี้

3.1 กำหนดให้เทรดบนหน่วยประมวลผลกราฟิกเข้าถึงข้อมูลผ่านเฉพาะหน่วยความจำแบบโกลบอลเท่านั้น

3.2 กำหนดให้เทรดบนหน่วยประมวลผลกราฟิกเข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอล และอนุญาตให้แต่ละเทรดที่มีการทำงานอยู่ภายในในบล็อกเดียวกันเข้าถึงหน่วยความจำแบบแชร์ได้

ศึกษา 3 ปัจจัยหลักที่ส่งผลต่อการเร่งสมรรถนะความเร็วการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับแบบจำลองสถานการณ์ปัญหา News Dealer ในงานวิจัยนี้สามารถแบ่งการทดสอบเป็น 8 กรณี แสดงดังตารางที่ 2

ตารางที่ 2 การทดสอบ 8 กรณีที่ศึกษาจาก 3 ปัจจัยการเร่งสมรรถนะการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการจำลองสถานการณ์ปัญหา News Dealer

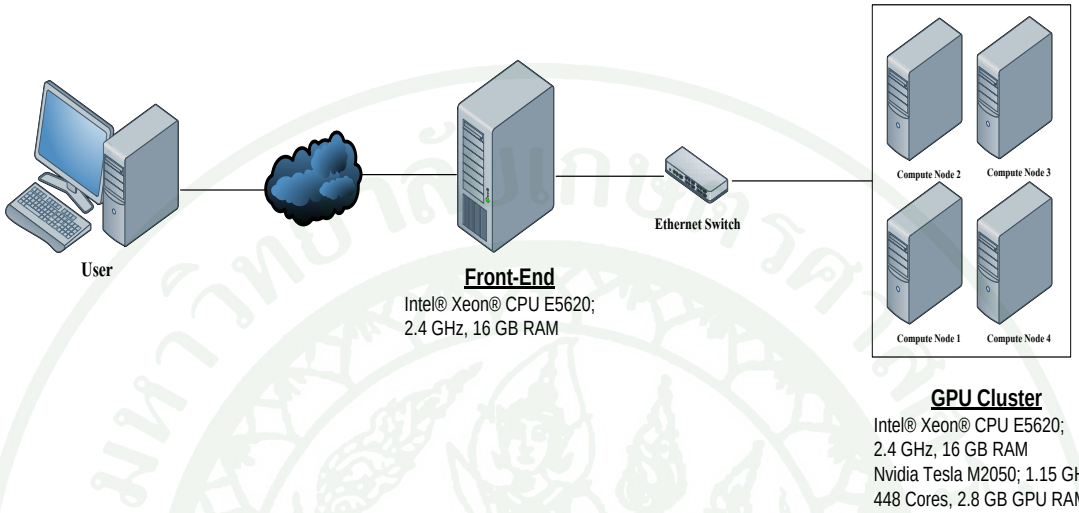
กรณีที่	1 จีพียูเทรดคำนวณผลกำไร	การสุ่มข้อมูล	การเข้าถึงข้อมูลบนหน่วยความจำของจีพียู
1	1 วัน	ซีพียู	หน่วยความจำโกลบอลเท่านั้น
2	N วัน (1 รอบการคำนวณ)	ซีพียู	หน่วยความจำโกลบอลเท่านั้น
3	1 วัน	จีพียู	หน่วยความจำโกลบอลเท่านั้น
4	N วัน (1 รอบการคำนวณ)	จีพียู	หน่วยความจำโกลบอลเท่านั้น
5	1 วัน	ซีพียู	หน่วยความจำโกลบอล และแชร์
6	N วัน (1 รอบการคำนวณ)	ซีพียู	หน่วยความจำโกลบอล และแชร์
7	1 วัน	จีพียู	หน่วยความจำโกลบอล และแชร์
8	N วัน (1 รอบการคำนวณ)	จีพียู	หน่วยความจำโกลบอล และแชร์

ในการทดสอบนี้เพื่อป้องกันการเกิดปัญหา Memory Leak ได้กำหนดจำนวนบล็อกมากที่สุดที่สามารถประมวลผลได้ใน 1 เคอร์เนลบนหน่วยประมวลผลกราฟิก คือ 10,000 บล็อก เนื่องจากขนาดของหน่วยความจำบนหน่วยประมวลผลกราฟิกยังคงเป็นปัจจัยหลัก

แบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด

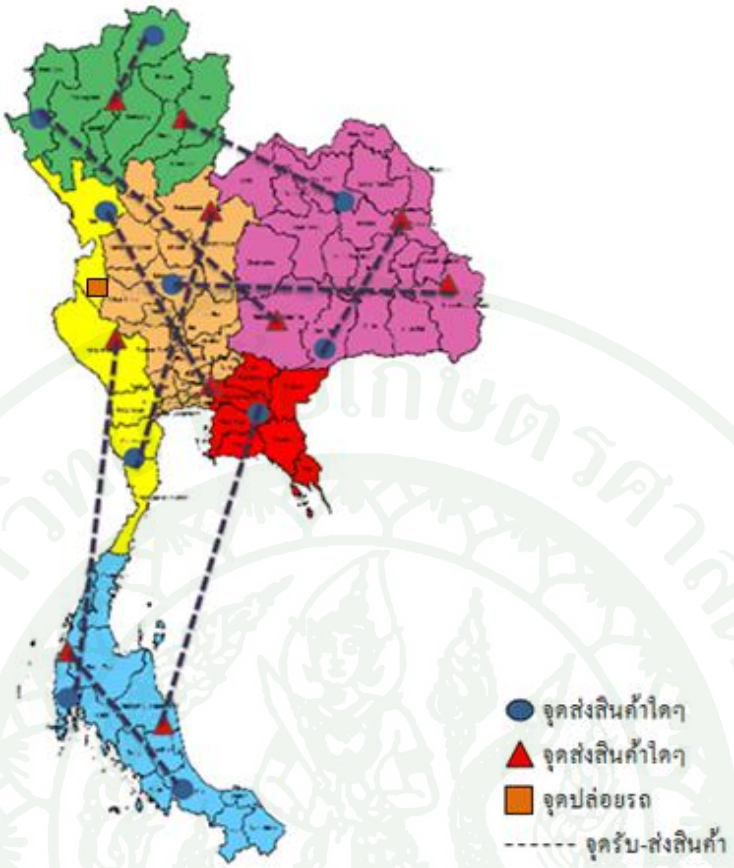
ออกแบบการคำนวณการหาผลลัพธ์ของแบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด โดยพัฒนาอัลกอริทึมให้สามารถประมวลผลแบบขนานบนจีพียูคลัสเตอร์เพื่อศึกษาปัจจัยการเร่งสมรรถนะความเร็วประมวลผลบนจีพียูคลัสเตอร์ ทดสอบกับระบบจีพียูคลัสเตอร์ ของภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ ซึ่งประกอบด้วยเครื่องเซิร์ฟเวอร์จำนวน 8 เครื่อง แต่ละเครื่องเซิร์ฟเวอร์ประกอบด้วย หน่วยประมวลกลางรุ่น Intel(R) Xeon(R) CPU E5620 มีอัตราการทำงานด้วยความถี่ 2.40 GHz, หน่วยความจำหลัก 16 GByte, หน่วยประมวลผลกราฟิกรุ่น Nvidia Tesla M2050 จำนวน 2 ใบ การ์ดแต่ละใบมีจำนวน 448 คอร์ มีอัตราการทำงานด้วยความถี่ 1.15 GHz และมีหน่วยความจำแบบโกลบอลเท่ากับ 2.8 GByte

การทดสอบนี้กำหนดการใช้จำนวนเครื่องเซิร์ฟเวอร์มากสุดเพียง 4 เครื่องเท่านั้น และกำหนดให้มีการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก 1 การ์ด/1เครื่องเซิร์ฟเวอร์ แสดงดังภาพที่ 13



ภาพที่ 13 โครงสร้างการเชื่อมต่อระหว่างส่วนประกอบต่างๆ ในระบบ

ในวิทยานิพนธ์นี้เลือกศึกษาปัญหา PDPTW ซึ่งเป็นปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด แสดงดังภาพที่ 14 เนื่องจากการคำนวณหาคำตอบต้องอยู่ภายใต้เงื่อนไขจำนวนมาก เช่น เวลาเปิด/ปิดทำการของแต่ละจุดรับ/ส่ง สินค้าที่แตกต่างกัน, ปริมาณสินค้าที่ต้องทำการบรรทุกขึ้นยานพาหนะในแต่ละจุดรับ/ส่ง สินค้าที่ไม่เท่ากัน, ความสามารถในการบรรทุกของยานพาหนะ และข้อจำกัดการใช้เชื้อเพลิงของยานพาหนะ เป็นต้น จึงมีขั้นตอนการคำนวณที่ซับซ้อนเพื่อให้ได้คำตอบที่เหมาะสมที่สุดสำหรับแบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด



ภาพที่ 14 ปัญหาการหาเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด

รูปแบบอินพุตของปัญหาการจัดเส้นทางการเดินรถที่มีจุดรับ/ส่งหลายจุด แสดงดังภาพที่ 15 โดยมีรายละเอียดดังนี้

Number of Customers หมายถึง จำนวนร้านค้าทั้งหมด (ไม่รวมจุดปล่อยรถ 1 จุด)

Number of Vehicles หมายถึง จำนวนรถบรรทุกทั้งหมดที่มีอยู่ในระบบ

Capacity หมายถึง จำนวนความจุของรถบรรทุกสินค้า

Speed หมายถึง อัตราความเร็วของรถบรรทุกสินค้า

Customer หมายถึง หมายเลขลำดับของร้านค้า เงื่อนไขดังนี้ ถ้ามีค่า $= 0$ หมายถึง จุดปล่อยรถ (Depot) และถ้ามีค่า $\neq 0$ หมายถึง ร้านค้า

X Coordination หมายถึง ตำแหน่งที่ตั้งของร้านค้าแนวแกน X

Y Coordination หมายถึง ตำแหน่งที่ตั้งของร้านค้าแนวแกน Y

Demand/Load หมายถึง ความต้องการในการบรรทุกสินค้า (Pickup) หรือส่งสินค้า (Delivery) มีเงื่อนไขดังนี้ ถ้ามีค่า $\rangle 0$ หมายถึง ร้านค้านี้เป็นจุดบรรทุกสินค้า, ถ้ามีค่า $\langle 0$ หมายถึง ร้านค้านี้เป็นจุดจัดส่งสินค้า และถ้ามีค่า $= 0$ หมายถึง จุดปล่อยรถ

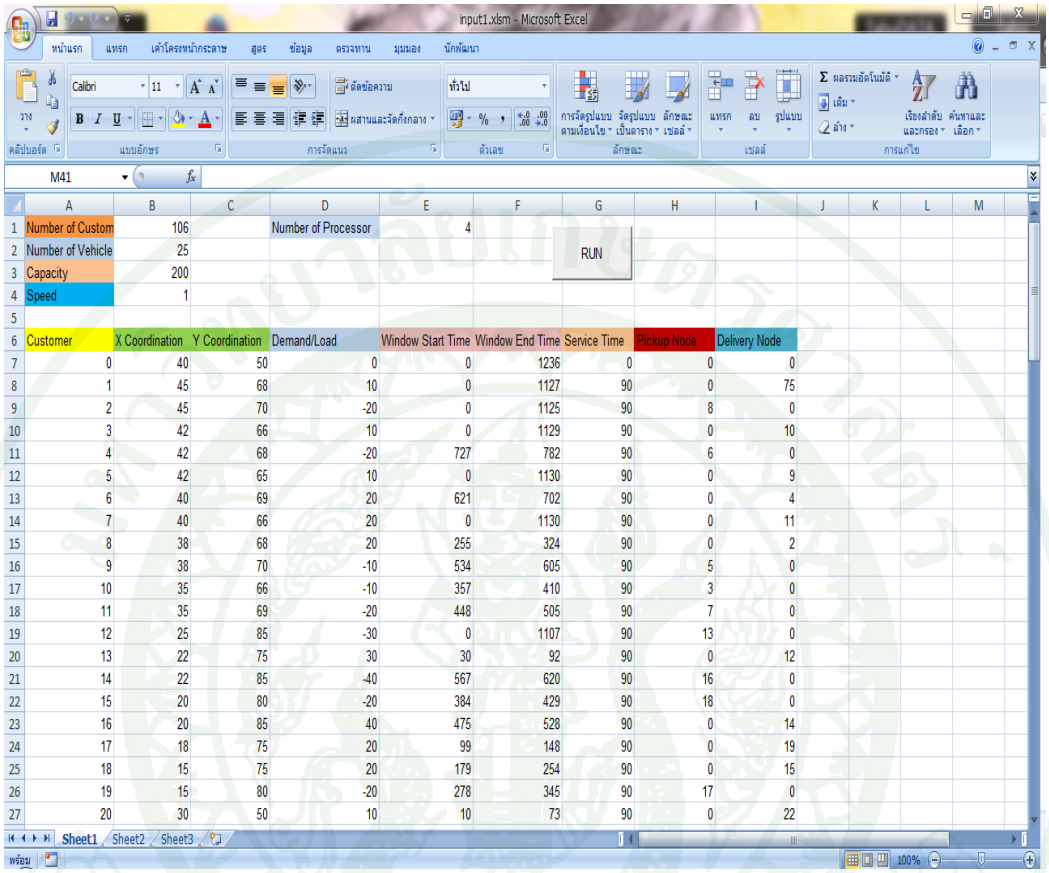
Window Start Time หมายถึง เวลาเปิดทำการรับ/ส่งสินค้าของร้านค้า

Window End Time หมายถึง เวลาปิดทำการรับ/ส่งสินค้าของร้านค้า

Service Time หมายถึง ระยะเวลาการบริการ ณ ร้านค้านั้นๆ (การรับ/ส่งสินค้า)

Pickup Node หมายถึง จุดรับสินค้า มีเงื่อนไขดังนี้ ถ้ามีค่า $\neq 0$ หมายถึง หมายเลขของร้านค้าที่ต้องรอรับสินค้ามาส่ง และถ้ามีค่า $= 0$ หมายถึง ไม่ต้องรับสินค้าจากร้านค้าใดๆ

Delivery Node หมายถึง จุดส่งสินค้า มีเงื่อนไขดังนี้ ถ้ามีค่า $\neq 0$ หมายถึง หมายเลขของร้านค้าที่ต้องบรรทุกสินค้าไปส่ง และถ้ามีค่า $= 0$ หมายถึง ไม่ต้องบรรทุกสินค้าไปส่งร้านค้าใดๆ



	A	B	C	D	E	F	G	H	I
1	Number of Custom	106		Number of Processor	4		RUN		
2	Number of Vehicle	25							
3	Capacity	200							
4	Speed	1							
5									
6	Customer	X Coordination	Y Coordination	Demand/Load	Window Start Time	Window End Time	Service Time	Pickup Node	Delivery Node
7	0	40	50	0	0	1236	0	0	0
8	1	45	68	10	0	1127	90	0	75
9	2	45	70	-20	0	1125	90	8	0
10	3	42	66	10	0	1129	90	0	10
11	4	42	68	-20	727	782	90	6	0
12	5	42	65	10	0	1130	90	0	9
13	6	40	69	20	621	702	90	0	4
14	7	40	66	20	0	1130	90	0	11
15	8	38	68	20	255	324	90	0	2
16	9	38	70	-10	534	605	90	5	0
17	10	35	66	-10	357	410	90	3	0
18	11	35	69	-20	448	505	90	7	0
19	12	25	85	-30	0	1107	90	13	0
20	13	22	75	30	30	92	90	0	12
21	14	22	85	-40	567	620	90	16	0
22	15	20	80	-20	384	429	90	18	0
23	16	20	85	40	475	528	90	0	14
24	17	18	75	20	99	148	90	0	19
25	18	15	75	20	179	254	90	0	15
26	19	15	80	-20	278	345	90	17	0
27	20	30	50	10	10	73	90	0	22

ภาพที่ 15 ตัวอย่างอินพุตของปัญหาการจัดเส้นทางการเดินรถที่มีจุดรับ/ส่งสินค้า 106 จุด

วัตถุประสงค์การแก้ปัญหา PDPTW

1. คำนวณหาจำนวนการใช้รถบรรทุกสินค้าน้อยที่สุดที่สามารถให้บริการแก่ลูกค้า (Customer) ครบทั้งหมด และใช้ระยะเวลาในการเดินทางที่อยู่ภายใต้เงื่อนไขของเวลา

2. ระยะเวลาที่ใช้ในการเดินทางของรถบรรทุกสินค้าต้องเป็นระยะเวลาที่น้อยที่สุด

เงื่อนไขบังคับของการแก้ปัญหา PDPTW

1. รถบรรทุกสินค้าทุกคันที่มีอยู่ในระบบ มีขีดความสามารถการบรรทุกสินค้าที่เท่ากัน

2. รถบรรทุกแต่ละคันเริ่มต้น และสิ้นสุดการให้บริการ ณ จุดปล่อยรถที่ 0

3. รถบรรทุกสินค้าแต่ละคันจะต้องออกตัว และกลับมายังจุดปล่อยรถเพียงแค่ครั้งเดียวเท่านั้น

4. เมื่อเริ่มต้นการให้บริการ รถบรรทุกจะต้องเดินทางออกจากจุดปล่อยรถที่ 0 ไปยังร้านค้าที่มีการร้องขอการบริการบรรทุกสินค้าไปส่ง ณ ร้านค้าอื่นๆเสมอ

5. แต่ละจุดรับ/ส่งสินค้าที่มีการร้องขอการบริการ จะมีเพียงแค่รถบรรทุกคันเดียวเท่านั้นที่สามารถให้บริการได้

6. ถ้ารถบรรทุกสินค้าเดินทางมาถึงก่อนเวลาเปิดทำการของร้านค้าลำดับที่ i รถบรรทุกสินค้าจะต้องทำการรอจนถึงช่วงเวลาเปิดทำการ

7. ระยะเวลาเดินทางตลอดเส้นทางรวมกับเวลาที่ให้บริการในแต่ละร้านค้าลำดับที่ i ใดๆ ไม่สามารถเกินเวลาปิดทำการของร้านค้าลำดับที่ i ได้

จากการศึกษางานวิจัยของกลุ่ม Antoine (2001), Hoong (2001) และ Quan (2006) ในงานวิจัยนี้ทำการปรับประยุกต์วิธีการค้นหาคำตอบเพื่อให้ค้นหาวิธีที่เหมาะสมกับการคำนวณหาผลลัพธ์สำหรับแบบจำลองสถานการณ์ปัญหา PDPTW

วิธีการคำนวณหาคำตอบสำหรับแบบจำลองสถานการณ์ปัญหา PDPTW แสดงดังภาคผนวก ค. ซึ่งประกอบด้วย 2 ขั้นตอน คือ 1) คำนวณหาคำตอบเริ่มต้น ด้วยกระบวนการ Insertion Heuristic และ 2) ทำการค้นหาคำตอบใกล้เคียง (Neighborhood Search) จากบริเวณคำตอบที่ได้จากขั้นตอนที่ 1 ด้วยกระบวนการค้นหาแบบทาบู จากนั้นทำการปรับปรุงคำตอบปัจจุบันเมื่อได้คำตอบข้างเคียงที่ดีกว่า

การเลือกคำตอบที่เหมาะสมที่สุดสำหรับแบบจำลองสถานการณ์ปัญหา PDPTW ต้องทำการเลือกจากชุดคำตอบที่เป็นไปได้ทั้งหมด ซึ่งได้มาจากการค้นหาคำตอบเริ่มต้นในทุกๆกรณีที่เป็นไปได้ทั้งหมดก่อนนำไปปรับปรุงค่าคำตอบให้ดีขึ้นด้วยวิธีการทาบู

ชุดข้อมูลที่มีผลต่อค่าคำตอบเริ่มต้น คือ ชุดข้อมูล A ที่มีการจัดเรียงช่วงเวลาเปิดทำการของ “จุดบรรทุกสินค้า”ทั้งหมด การคำนวณหาคำตอบเริ่มต้นด้วยกระบวนการ Insertion Heuristic จะทำการเลือกตำแหน่งชุดข้อมูลที่จัดเรียงช่วงเวลาเปิดทำการของ “จุดบรรทุกสินค้า” ออกมาพิจารณาเป็นลำดับ ซึ่งการเลือกชุดข้อมูลที่มีจัดเรียงช่วงเวลาเปิดทำการเร็วไม่ได้การันตีถึงการใช้ระยะเวลารวมที่สั้นที่สุด เนื่องจาก “จุดบรรทุกสินค้า” ที่มีเวลาเปิดทำการเร็วอาจมีระยะทางที่อยู่ไกลจากตำแหน่งที่เรากำลังพิจารณา เป็นต้น ดังนั้นจึงทำการสลับคู่ตำแหน่งที่เป็นไปได้ของชุดข้อมูล A ที่มีการจัดเรียงช่วงเวลาเปิดทำการของ“จุดบรรทุกสินค้า”ทั้งหมดก่อนการคำนวณหาคำตอบเริ่มต้นในแต่ละครั้งที่ทำการสลับตำแหน่ง แสดงดังภาพที่ 16ก – 16ข

1	**3**	**5**	**7**	**31**	**37**	**50**	**52**	**82**	**83**	**95**	**...**	**68**	**อาร์เรย์ A**
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		$[\frac{n}{2}]$	

16ก. การจัดเรียงข้อมูลในอาร์เรย์ A

<u>สลับครั้งที่ 1</u>: ตำแหน่ง 1, 2

3	**1**	**5**	**7**	**31**	**37**	**50**	**52**	**82**	**83**	**95**	**...**	**68**	**อาร์เรย์ A**
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		$[\frac{n}{2}]$	

<u>สลับครั้งที่ 2</u>: ตำแหน่ง 1, 3

5	**3**	**1**	**7**	**31**	**37**	**50**	**52**	**82**	**83**	**95**	**...**	**68**	**อาร์เรย์ A**
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		$[\frac{n}{2}]$	

<u>สลับครั้งที่ 3</u>: ตำแหน่ง 1, 4

7	**3**	**5**	**1**	**31**	**37**	**50**	**52**	**82**	**83**	**95**	**...**	**68**	**อาร์เรย์ A**
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		$[\frac{n}{2}]$	

: : : : : : : : : : : : :

ภาพที่ 16 การสลับตำแหน่งข้อมูลในอาร์เรย์ A ก่อนการคำตอบหาคำตอบเริ่มต้น

<u>สลับครั้งสุดท้าย</u> : ตำแหน่ง $\frac{n}{2}-1, \frac{n}{2}$

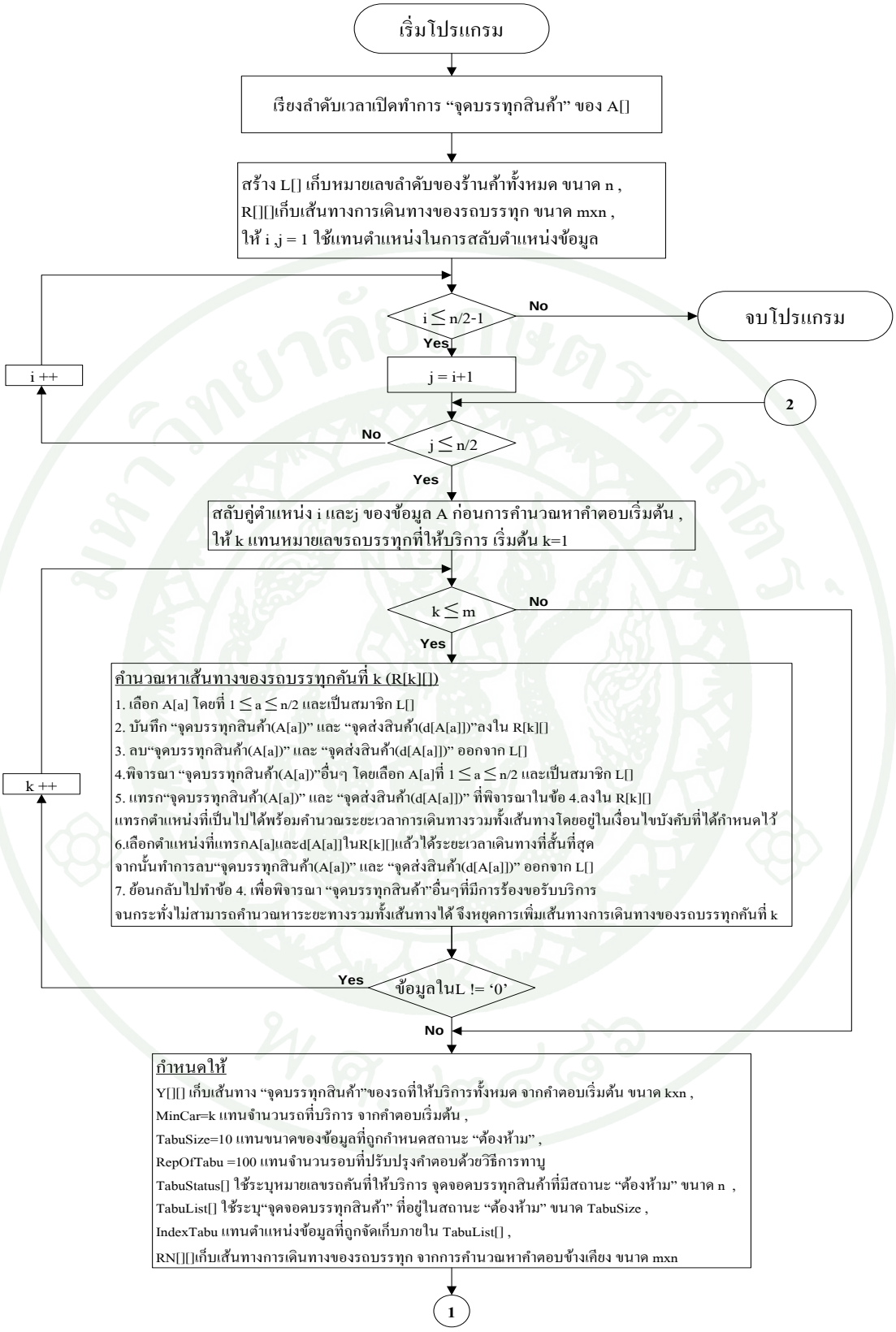
1	**3**	**5**	**7**	**31**	**37**	**50**	**52**	**82**	**83**	**95**	**...**	**68**	**23**	**อาร์เรย์** A
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		$[\frac{n}{2}-1]$	$[\frac{n}{2}]$	

16ข. การสลับคู่ตำแหน่งข้อมูลที่เป็นไปได้ในอาร์เรย์ A

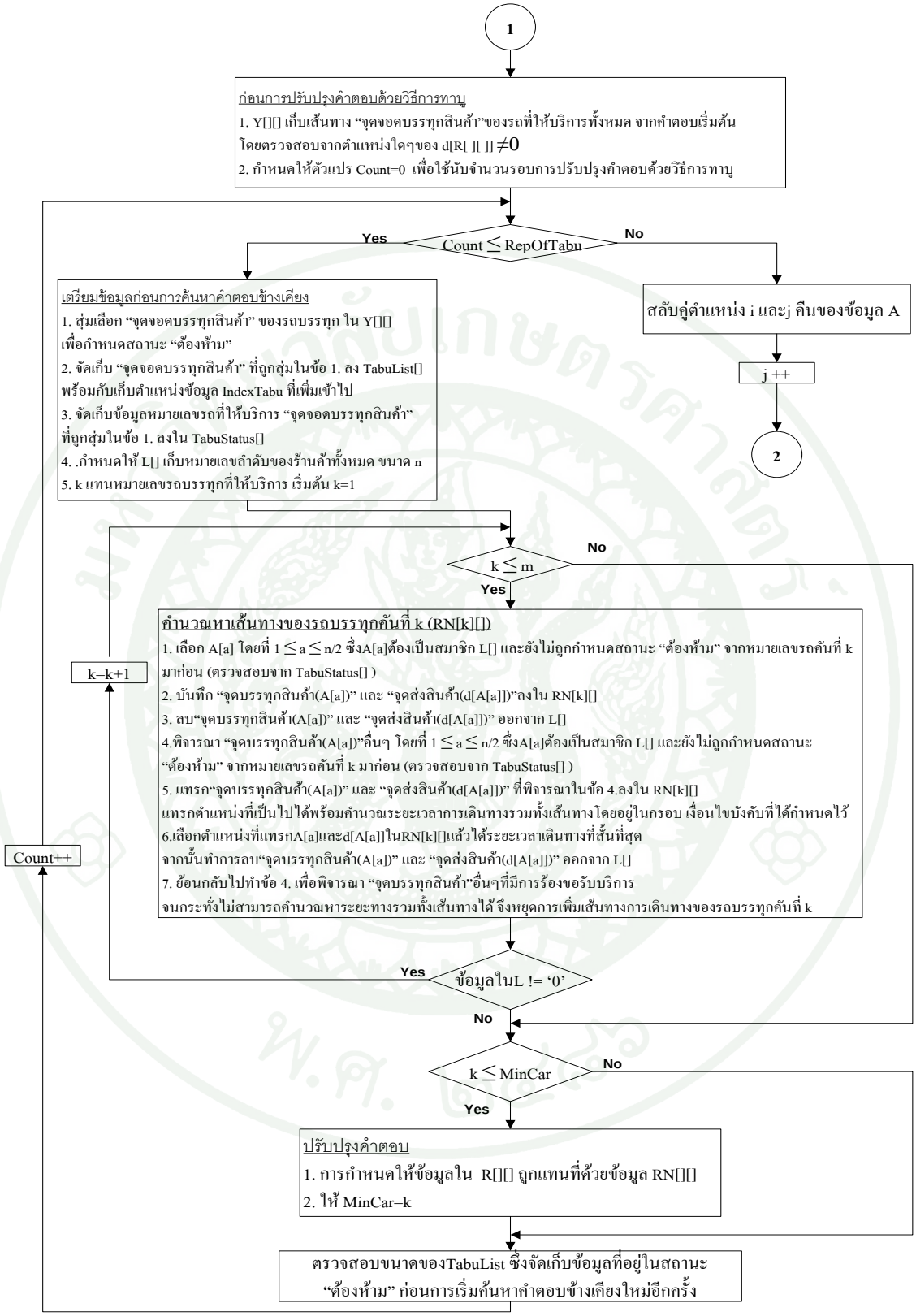
ภาพที่ 16 (ต่อ)

จากการสลับคู่ตำแหน่งที่เป็นไปได้ของชุดข้อมูล A ทำให้ได้กลุ่มชุดคำตอบทั้งสิ้น $\left[\frac{\frac{n}{2}\times(\frac{n}{2}-1)}{2}\right]$ จำนวน เลือกชุดคำตอบที่ให้ค่าฟังก์ชั่นวัตถุประสงค์ดีที่สุดเพื่อใช้เป็นคำตอบที่เหมาะสมที่สุดสำหรับแบบจำลองสถานการณ์ปัญหา PDPTW

หลักในการคำนวณหาคำตอบสำหรับแบบจำลองสถานการณ์ปัญหา PDPTW ประกอบด้วย 2 ขั้นตอน คือ ขั้นตอนการค้นหาคำตอบเริ่มต้นด้วยวิธี Insertion Heuristic และขั้นตอนการปรับปรุงค่าคำตอบด้วยการค้นหาคำตอบข้างเคียงจากวิธีการทาบู ซึ่งทั้ง 2 ขั้นตอนการคำนวณดังกล่าวสามารถเขียนโปรแกรมแบบลำดับแสดงดังภาพที่ 17 และ18



ภาพที่ 17 ขั้นตอนการทำงานของโปรแกรมการแก้ปัญหา PDPTW



ภาพที่ 18 ขั้นตอนการทำงานของโปรแกรมการแก้ปัญหา PDPTW (ต่อ)

งานวิจัยนี้ได้ทำทดสอบการโปรแกรมแบบลำดับกับแบบจำลองสถานการณ์ปัญหาการจัดเส้นทางยานพาหนะที่มีจำนวนจุดรับ/ส่งสินค้าขนาดต่างๆ โดยทำการดาวน์โหลดตัวอย่างปัญหา PDPTW จาก Li & Lim Benchmark (2010) เพื่อทำการวัดเวลาที่ใช้ประมวลผล แสดงดังตารางที่ 3

ตารางที่ 3 เวลา (วินาที) ในการประมวลผลการโปรแกรมแบบลำดับ การจำลองสถานการณ์ของปัญหาPDPTW ขนาด 106, 212 และ 422 จุดรับ/ส่งสินค้า

Problem Sizes (locations)	**Execution time(Seconds)**			**Total Execution Time (Seconds)**
	Read Input	**Compute**	**Write Output**	
106	0.0011	762.00	0.0011	762.00
212	0.0014	16645.40	0.0014	16645.40
422	0.0017	263030.00	0.0017	263030.00

จากการทดสอบการคำนวณการหาคำตอบที่เหมาะสมให้กับแบบจำลองสถานการณ์ PDPTW ขนาด 106, 212 และ422 พบว่าระยะเวลาที่ใช้ในการประมวลผลมีจำนวนมากขึ้นตามจำนวนรอบการคำนวณชุดคำตอบที่เป็นไปได้ของปัญหาขนาด n โดยที่จำนวนชุดคำตอบที่เป็นไปได้คำนวณจาก $\left\lceil \frac{\frac{n}{2} \times (\frac{n}{2} - 1)}{2} \right\rceil$

ดังนั้นในงานวิจัยนี้จึงมีแนวคิดออกแบบการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด ซึ่งทดสอบแนวคิดนี้ตามโครงสร้างระบบคลัสเตอร์ประกอบด้วย เครื่องคอมพิวเตอร์หลายๆ เครื่องที่มีการทำงานประสานกันผ่าน MPI โดยมีเครื่องคอมพิวเตอร์ 1 เครื่องที่ทำหน้าที่จัดสรรกระจายงานในระบบเรียกว่า “Front-End” และเพื่อเร่งสมรรถนะความเร็วในการประมวลผลของระบบคลัสเตอร์ ผู้วิจัยจึงประยุกต์นำหน่วยประมวลผลกราฟิกซึ่งมีการคำนวณแบบขนานมาประมวลผลร่วมกับหน่วยประมวลผลกลางในแต่ละเครื่องคอมพิวเตอร์ที่มีอยู่ในระบบ เรียกว่า “Compute-Node”

การคำนวณการหาคำตอบให้กับแบบจำลองสถานการณ์ปัญหา PDPTW มีแนวคิดเช่นเดียวกับการโปรแกรมแบบลำดับ แต่งานวิจัยนี้พัฒนาอัลกอริทึมให้เหมาะสมกับการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์ ซึ่งมีขั้นตอนการทำงานดังนี้

1. กำหนด S คือ จำนวน Compute-Node ที่ใช้ประมวลผลในระบบคลัสเตอร์ และ Worker_ID คือ หมายเลขประจำ Compute-Node แต่ละเครื่อง

2. Front-End ทำการกระจายข้อมูลอินพุต, ขนาดของทาบู (Tabu List Size), จำนวนรอบในการปรับปรุงคำตอบด้วยวิธีการทาบู(Iteration), จำนวนเทรดภายในบล็อก(Thread Size)ที่ประมวลผลบนหน่วยประมวลกราฟิก และจำนวน Compute-Node ที่ใช้ในการประมวลผล โดยผ่าน Message Passing Interface (MPI)

3. ก่อนคำนวณหาคำตอบเริ่มต้นให้แต่ละ Compute-Node ทำการกำหนดพารามิเตอร์ต่างๆ จากข้อมูลอินพุต และคำนวณหาจำนวนคำตอบที่เป็นไปได้ทุกกรณีของปัญหาขนาด n โดยกำหนดให้ตัวแปร *Job* คือ อาร์เรย์1มิติ จัดเก็บการสลับคู่ตำแหน่งที่เป็นไปได้ของชุดข้อมูล A ซึ่งเป็นชุดข้อมูลที่มีการจัดเรียงช่วงเวลาเปิดทำการของ“จุดบรรทุกสินค้า”ทั้งหมด ใช้สำหรับการหาคำตอบเริ่มต้น แสดงดังภาพที่ 19ก – 19ข

1	3	5	7	31	37	50	52	82	83	95	...	68	**อาร์เรย์** A
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		$[\frac{n}{2}]$	

19ก. การจัดเรียงข้อมูลในอาร์เรย์ A

อาร์เรย์ *Job*

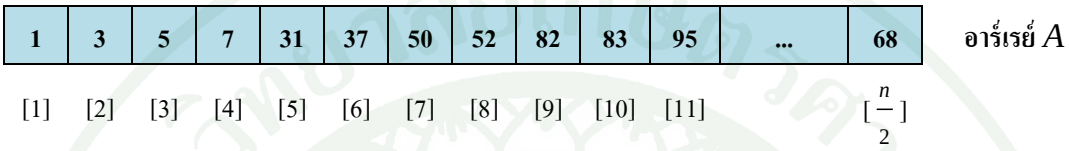
1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11	1,12	...	$\frac{n}{2}-1, \frac{n}{2}$
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		[P]

19ข. ข้อมูลในอาร์เรย์ *Job*

ภาพที่ 19 อาร์เรย์ *Job* จัดเก็บการสลับคู่ตำแหน่งที่เป็นไปได้ทั้งหมดของอาร์เรย์ A

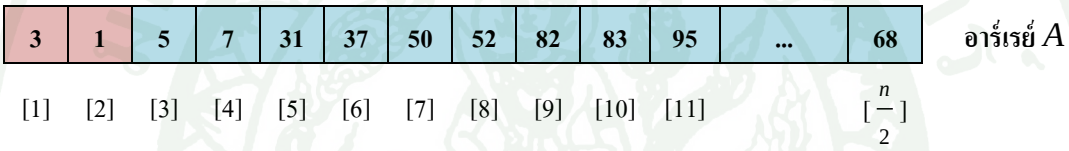
จากการสลับคู่ตำแหน่งที่เป็นไปได้ของชุดข้อมูล A ทำให้ได้กลุ่มชุดคำตอบทั้งสิ้น P จำนวน ซึ่งคือจำนวน P งานที่ต้องทำการประมวลผลเช่นเดียวกัน

4. กำหนดจำนวนเทรดให้เท่ากับ P งานที่ต้องทำการประมวลผล แต่ละเทรดใช้ข้อมูลในอาร์เรย์ *Job* สลับตำแหน่งข้อมูลในอาร์เรย์ A ก่อนการคำนวณหาคำตอบเริ่มต้น และปรับปรุงคำตอบด้วยวิธีการทาบู แสดงดังภาพที่ 20ก – 20ข

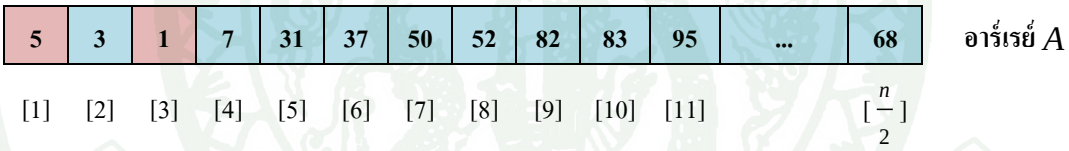


20ก. การจัดเรียงข้อมูลในอาร์เรย์ A

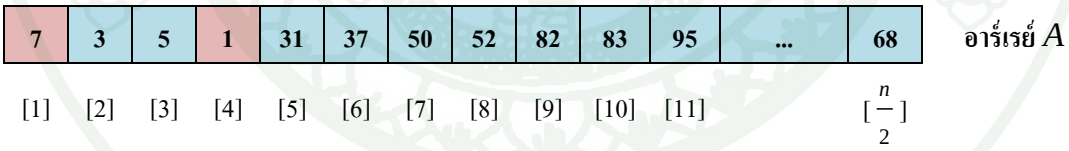
Thread_ID 1: ใช้ข้อมูล *Job*[1] ทำการสลับตำแหน่งข้อมูลในอาร์เรย์ A



Thread_ID 2: ใช้ข้อมูล *Job*[2] ทำการสลับตำแหน่งข้อมูลในอาร์เรย์ A

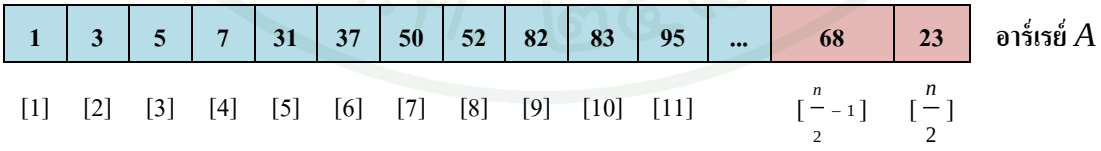


Thread_ID 3: ใช้ข้อมูล *Job*[3] ทำการสลับตำแหน่งข้อมูลในอาร์เรย์ A



: : : : : : : : : : : : :

Thread_ID P **:** ใช้ข้อมูล *Job*[P] ทำการสลับตำแหน่งข้อมูลในอาร์เรย์ A



20ข. ใช้ข้อมูลในอาร์เรย์ *Job* สลับตำแหน่งข้อมูลในอาร์เรย์ A

ภาพที่ 20 ข้อมูล A ที่แต่ละเทรดใช้ประมวลผล

5. ทำการแบ่งจำนวนงานที่แต่ละ Compute-Node ต้องทำการคำนวณดังสมการที่ 2 ซึ่งสามารถแสดง Thread_ID ที่ทำงานบนแต่ละ Compute-Node ดังตารางที่ 4

$$J = \frac{P}{S} \tag{2}$$

J คือ จำนวนงาน ที่แต่ละ Compute-Node ต้องประมวลผล

P คือ จำนวนงาน ของปัญหาขนาด n

S คือ จำนวน Compute-Node ที่ใช้ประมวลผลในระบบคลัสเตอร์

ตารางที่ 4 การกำหนดหมายเลข Thread_ID ที่ประมวลผลแต่ละ Compute-Node

Worker_ID	Start Thread_ID	End Thread_ID
1	1	J
2	$J+1$	$2*J$
3	$2*J+1$	$3*J$
…	…	…

6. แต่ละเทรดทำงานเป็นอิสระต่อกัน ทำให้แต่ละ Compute-Node ต้องจัดสรรการใช้หน่วยความจำบนหน่วยประมวลผลกราฟิกเอง โดยการจองพื้นที่ในส่วนหน่วยความจำโกลบอลบนหน่วยประมวลผลกราฟิกเอง เพื่อใช้คัดลอกข้อมูลจากหน่วยความจำหลักที่แต่ละเทรดจำเป็นต้องใช้ในระหว่างการคำนวณบนหน่วยประมวลผลกราฟิก

7. ก่อนเริ่มทำการคำนวณแบบขนานให้กำหนดจำนวนบล็อกที่ใช้ประมวลผลบนหน่วยประมวลผลกราฟิก ซึ่งแต่ละ Compute-Node สามารถคำนวณหาจำนวนบล็อกที่ใช้ประมวลผล ณ เวลาหนึ่งๆ ดังสมการที่ 3

$$nBlocks = \frac{J}{T} \tag{3}$$

$nBlocks$ คือ จำนวนบล็อก ที่แต่ละ Compute-Node ต้องประมวลผล

J คือ จำนวนงานที่แต่ละ Compute-Node ต้องประมวลผล

T คือ จำนวนเทรดภายในบล็อก ที่ประมวลผลบนหน่วยประมวลผลกราฟิก

8. เมื่อ Compute-Node ทำการเรียกฟังก์ชั่น__global__ ทุกเทรดบน Compute-Node จะเริ่มต้นประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก ซึ่งแต่ละเทรดนั้นจะคำนวณหาผลลัพธ์โดยการค้นหาคำตอบเริ่มต้นด้วยวิธี Insertion Heuristic และการปรับปรุงค่าคำตอบด้วยการค้นหาคำตอบข้างเคียงจากวิธีการทาบู (ดังภาคผนวก ค.)

9. เมื่อแต่ละเทรดเสร็จสิ้นการคำนวณบนหน่วยประมวลผลกราฟิก จะทำการคัดลอกผลลัพธ์ของแต่เทรดจากหน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิกไปยังหน่วยความจำหลัก

10. แต่ละ Compute-Node เลือกผลลัพธ์ที่ให้ค่าฟังก์ชั่นวัตถุประสงค์ดีที่สุด ใช้เป็นคำตอบที่เหมาะสมที่สุดสำหรับแบบจำลองสถานการณ์ปัญหา PDPTW ส่งกลับไปยัง Front-End เพื่อให้ Front-End ทำการคัดลอกข้อมูลลงเอาท์พุตไฟล์ต่อไป

จากการออกแบบขั้นตอนการทำงานแบบขนานบนระบบจีพียูคลัสเตอร์ทั้ง10ข้อ สามารถแสดงโครงสร้างการทำงานดังภาพที่ 21 และเพื่อทดสอบปัจจัยที่มีผลต่อการเร่งสมรรถนะการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการจำลองสถานการณ์ปัญหา PDPTW ขนาดต่างๆ จึงทำการศึกษา 3 ประเด็นหลักดังนี้

1. จำนวน Compute-Node

จำนวน Compute-Node ที่ใช้ประมวลผลในระบบจีพียูคลัสเตอร์มีความสัมพันธ์กับขนาดปัญหาที่ทำการคำนวณซึ่งมีผลต่อการเร่งสมรรถนะการประมวลผล ดังนั้นการทดสอบปัจจัยนี้จึงกำหนดการเพิ่มจำนวน Compute-Node ที่ใช้คำนวณสูงสุดถึง 4 Compute-Node

2. จำนวนเทรดภายในบล็อก

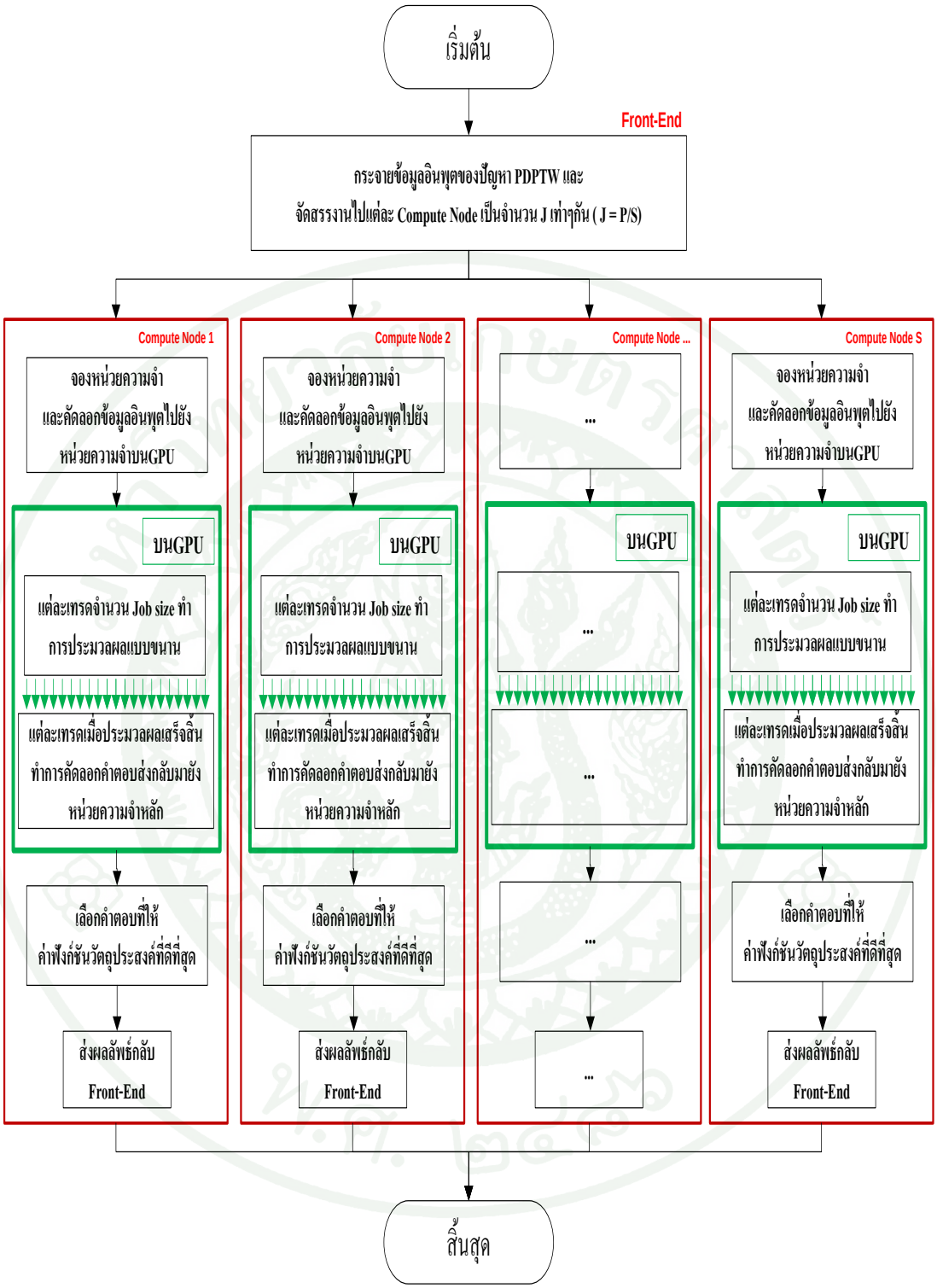
การกำหนดจำนวนเทรด/1บล็อกมากไป อาจส่งผลกระทบต่อการเร่งสมรรถนะการคำนวณกับปัญหาที่มีขนาดใหญ่ เนื่องจากต้องใช้ระยะเวลารอการประมวลผลทุกเทรดภายในบล็อกเสร็จสิ้น จึงจะสามารถคำนวณบล็อกถัดไป ดังนั้นการทดสอบปัจจัยนี้จึงกำหนดจำนวนเทรดภายในบล็อก ดังนี้ 32, 64 และ128 เทรด/บล็อก

3. การเข้าใช้หน่วยความจำบนหน่วยประมวลผลกราฟิกของแต่ละเทรด

เพื่อศึกษาเวลาประมวลผลที่เทรดใช้เข้าถึงข้อมูลบริเวณหน่วยความจำประเภทต่างๆ บนหน่วยประมวลผลกราฟิกจึงแบ่งเป็น 2 กรณี ดังนี้

3.1 กำหนดให้เทรดบนหน่วยประมวลผลกราฟิกเข้าถึงข้อมูลผ่านเฉพาะหน่วยความจำแบบโกลบอลเท่านั้น

3.2 กำหนดให้เทรดบนหน่วยประมวลผลกราฟิกเข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอล และอนุญาตให้แต่ละเทรดที่มีการทำงานอยู่ภายในในบล็อกเดียวกันเข้าถึงหน่วยความจำแบบแชร์ได้



ภาพที่ 21 โครงสร้างการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์

ผลการทดลอง

แบบจำลองสถานการณ์ของระบบคลังสินค้า

ผลการทดสอบเวลารวมที่ใช้ประมวลผลตามการออกแบบการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิกทั้ง 8 กรณีสำหรับการจำลองสถานการณ์ปัญหา News Dealer พบว่า ระยะเวลาที่ใช้ในการคำนวณการโปรแกรมแบบขนานเสร็จสิ้นเร็วกว่าการคำนวณการโปรแกรมแบบลำดับ เมื่อมีการคำนวณปัญหา News Dealer ที่มีขนาดใหญ่แสดงดังตารางที่ 3 – 10

ตารางที่ 5 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 1 (1 จีพียูเทรดคำนวณผลกำไรต่อ 1 วัน, สุ่มข้อมูลบนซีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การดึงข้อมูลโดย CPU	
1	0.0004	0.0115	0.3904	0.0055	0.4078
10	0.0030	0.1116	0.0928	0.0487	0.2561
100	0.0281	0.9409	0.2276	0.4685	1.6650
1,000	0.2769	8.9886	1.7072	4.6491	15.6218
10,000	2.7776	89.6740	16.7385	46.5868	155.7770

ตารางที่ 6 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 2 (1 จีพียูเทรดคำนวณผลกำไรรวม N วัน, สุ่มข้อมูลบนซีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูล โดย CPU	
1	9e-0.5	0.0181	0.0523	0.0048	0.0753
10	0.0004	0.1458	0.0713	0.0478	0.2653
100	0.0030	1.2265	0.1544	0.5838	1.9677
1,000	0.0297	11.7359	1.0117	5.8790	18.6563
10,000	0.2962	115.9720	9.6648	58.8213	184.7540

ตารางที่ 7 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 3 (1 จีพียูเทรดคำนวณผลกำไรต่อ 1 วัน, สุ่มข้อมูลบนจีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูล โดย CPU	
1	0.0004	0.0123	0.0494	0.0025	0.0645
10	0.0028	0.1188	0.0614	0.0249	0.2079
100	0.0283	0.9995	0.1976	0.2425	1.4679
1,000	0.2809	9.3953	1.3795	2.42782	13.4835
10,000	2.8190	93.5350	13.3596	24.1284	133.8420

ตารางที่ 8 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก สำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 4 (1 จีพียูเทรดคำนวณผลกำไรรวม N วัน, สุ่มข้อมูลบนจีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูลโดย CPU	
1	8.1e-0.5	0.0192	0.0523	0.0002	0.0717
10	0.0004	0.1541	0.0647	0.0012	0.2203
100	0.0030	1.2857	0.0873	0.0113	1.3873
1,000	0.0299	12.0336	0.2032	0.1130	12.3798
10,000	0.2948	119.4960	1.2610	1.1167	122.1680

ตารางที่ 9 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 5 (1 จีพียูเทรดคำนวณผลกำไรต่อ 1 วัน, สุ่มข้อมูลบนซีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และ แชร์บนจีพียู)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูลโดย CPU	
1	0.0004	0.0105	0.0498	0.0047	0.0654
10	0.0027	0.1004	0.0692	0.0482	0.2205
100	0.0275	0.8360	0.2248	0.4675	1.5557
1,000	0.2716	7.8398	1.6118	4.6483	14.3715
10,000	2.7012	77.8610	15.6405	46.4547	142.6570

ตารางที่ 10 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 6 (1 จีพียูเทรดคำนวณผลกำไรรวม N วัน, สุ่มข้อมูลบนซีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และแชร์บนจีพียู)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมการประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูล โดย CPU	
1	9.5e-0.5	0.0069	0.0530	0.0048	0.0648
10	0.0004	0.0574	0.0728	0.0481	0.1787
100	0.0031	0.5185	0.2177	0.5857	1.3249
1,000	0.0298	5.0381	1.0922	5.8962	12.0564
10,000	0.2956	50.7527	9.4963	58.6639	119.2090

ตารางที่ 11 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 7 (1 จีพียูเทรดคำนวณผลกำไรต่อ 1 วัน, สุ่มข้อมูลบนจีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และ แชร์บนจีพียู)

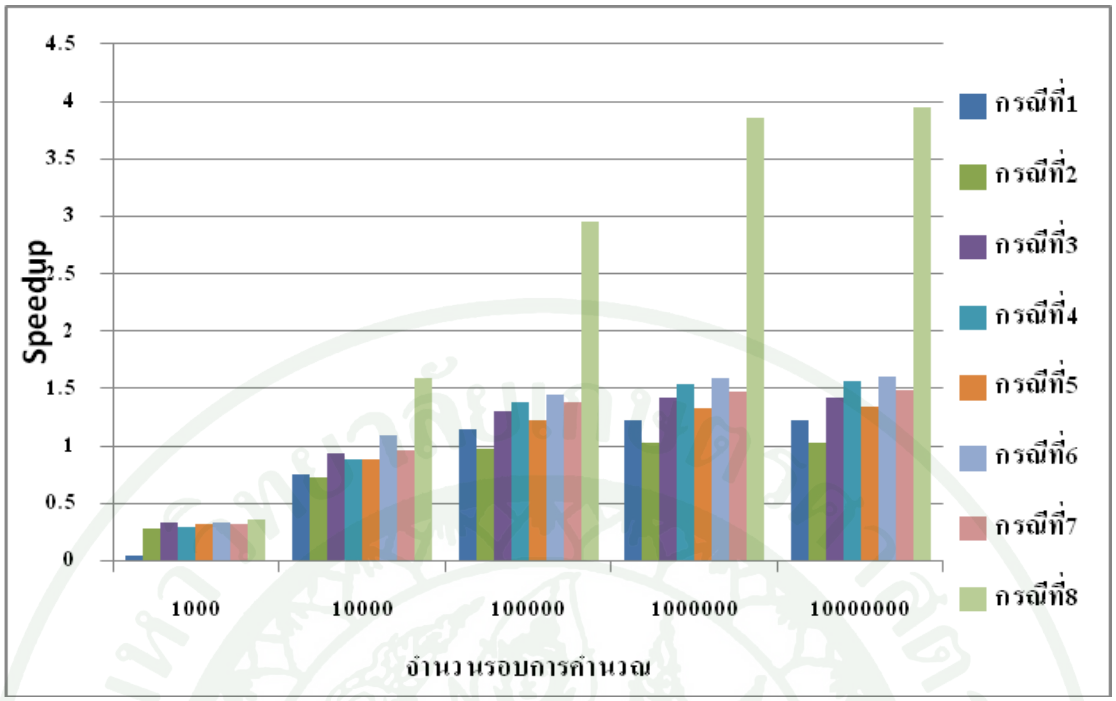
จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูล โดย CPU	
1	0.0004	0.0119	0.0508	0.0025	0.0655
10	0.0028	0.1141	0.0616	0.0249	0.2034
100	0.0286	0.9310	0.1850	0.2425	1.3872
1,000	0.2899	8.8120	1.3899	2.4323	12.9241
10,000	2.8958	87.4212	13.5100	24.1532	127.9800

ตารางที่ 12 เวลา (วินาที) ในการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer ที่มีการจัดการเทรดภายในบล็อก ดังกรณีที่ 8 (1 จีพียูเทรดคำนวณผลกำไรรวม N วัน, สุ่มข้อมูลบนจีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และแชร์บนจีพียู)

จำนวนพันรอบการคำนวณ (Iterations)	เวลา (วินาที)				เวลารวมประมวลผลแบบขนาน (วินาที)
	ประมวลผลบน CPU	ประมวลผลบน GPU	คัดลอกข้อมูลระหว่าง Main Memory ไป GPU Memory	การสุ่มข้อมูล โดย CPU	
1	8E-005	0.0077	0.0521	0.0001	0.0600
10	0.0004	0.0638	0.0566	0.0012	0.1219
100	0.0030	0.5508	0.0835	0.0114	0.6487
1,000	0.0297	4.6392	0.1705	0.1125	4.9520
10,000	0.2952	45.4824	1.3880	1.1174	48.2830

การทดสอบประสิทธิภาพการออกแบบการคำนวณแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการจำลองสถานการณ์ปัญหา News Dealer นี้เพื่อศึกษา 3 ปัจจัยการเร่งสมรรถนะความเร็วการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก ในงานวิจัยนี้ออกแบบการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกด้วยการกำหนดการคำนวณของจีพียูเทรด, การสุ่มตัวเลข และการเข้าถึงข้อมูลแต่ละจีพียูเทรดบนหน่วยความจำของหน่วยประมวลผลกราฟิกที่แตกต่างกัน โดยทำการทดสอบบนเครื่องคอมพิวเตอร์ชนิดพกพา 1 เครื่องที่ประกอบด้วย หน่วยประมวลผลกราฟิก รุ่น 9400 M มีจำนวน 2 มัลติคอร์ แต่ละมัลติคอร์ประกอบด้วย 8 คอร์

จากการศึกษา 3 ปัจจัยหลักที่ส่งผลต่อการเร่งสมรรถนะความเร็วการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับแบบจำลองสถานการณ์ปัญหา News Dealer ทั้ง 8 กรณี ดังตารางที่ 2 สามารถวิเคราะห์ประสิทธิภาพการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก ได้จากการคำนวณ Speedup ที่เพิ่มขึ้นแสดงดังภาพที่ 22



ภาพที่ 22 กราฟ Speedup สำหรับการทดสอบประสิทธิภาพการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกของปัญหา News Dealer

ผลการทดสอบประสิทธิภาพของการออกแบบการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer สามารถวิเคราะห์ได้ดังนี้

1. จากผลการทดสอบการประมวลผลแต่ละรอบการคำนวณการโปรแกรมแบบลำดับดังตารางที่ 1 และการโปรแกรมแบบขนานบนหน่วยประมาณผลกราฟิกทั้ง 8 กรณีดังตารางที่ 5 -12 การประมวลผลที่ใช้เวลาเร็วที่สุดในแต่ละรอบการคำนวณแสดงได้ดังนี้

1.1 กรณีที่ดีที่สุดของรอบการคำนวณ 1,000 รอบ คือ การประมวลผลโปรแกรมแบบลำดับ

1.2 กรณีที่ดีที่สุดของรอบการคำนวณ 10,000 ถึง 10,000,000 รอบ คือ การประมวลผลโปรแกรมแบบขนานกรณีที่ 8 (การออกแบบ 1 จีพียูเทรด คำนวณหาผลกำไรรวมรวมของ N วันที่จำลองสถานการณ์, สุ่มข้อมูลบนจีพียู และแต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอล และแชร์บนจีพียู)

ผลการทดสอบแสดงให้เห็นถึงปัญหาที่มีขนาดใหญ่เหมาะสมกับการประยุกต์นำหน่วยประมวลผลกราฟิกมาทำการโปรแกรมแบบขนานเพื่อช่วยเร่งความเร็วในการประมวลผล

2. การประเมินประสิทธิภาพของโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกด้วยกราฟ Speedup แสดงให้เห็นถึง 3 ปัจจัยหลักการที่ส่งผลต่อการเร่งสมรรถนะความเร็วการประมวลผลดังนี้

2.1 วิเคราะห์เวลารวมการทำงานของทุกเทรดที่ใช้ประมวลผลบนจีพียู โดยจำนวนบล็อกที่ประมวลผลที่การออกแบบให้ 1 จีพียูเทรดมีการคำนวณผลกำไรต่อ 1 วัน ดังกรณี {1,3,5 และ 6} มีจำนวนบล็อกที่ต้องคำนวณมากกว่าการออกแบบให้ 1 จีพียูเทรดมีการคำนวณผลกำไรรวม N วันดังกรณี {2,4,6 และ 8} จากการวิเคราะห์เปรียบเทียบการทดสอบกรณีที่ {1, 2}, {3, 4}, {5, 6} และ{7, 8} พบว่า จำนวนรอบการคำนวณตั้งแต่ 10,000 ถึง 10,000,000 กรณีที่ 1 สามารถเร่งสมรรถนะได้ดีกว่ากรณีที่ 2 เนื่องจากการคำนวณแต่ละจีพียูเทรดในกรณีที่ 1 มีการคำนวณที่ไม่ซับซ้อน ซึ่งคำนวณเพียงผลลัพธ์ 1 วัน จึงใช้เวลาการคำนวณ 1 บล็อกที่เร็วกว่ากรณีที่ 2 ที่ 1 จีพียูเทรดคำนวณผลลัพธ์กำไรรวมทั้งหมด N วัน แต่เมื่อมีการปรับปรุงการสุ่มข้อมูล และการเข้าถึงข้อมูลบนหน่วยความจำจีพียู ผลการทดสอบพบว่า การออกแบบให้ 1 จีพียูเทรดมีการคำนวณผลกำไรรวม N วัน จะสามารถเร่งสมรรถนะได้ดีมากกว่าการออกแบบให้ 1 จีพียูเทรดมีการคำนวณผลกำไรต่อ 1 วัน ดังกรณีที่ {3, 4}, {5, 6} และ{7, 8}

2.2 การจำลองสถานการณ์สำหรับปัญหา News Dealer จำเป็นต้องมีกระบวนการสุ่มตัวเลขขึ้นมาเพื่อจำลองสถานการณ์ยอดขาย และจำนวนความต้องการของลูกค้า ซึ่งใช้คำนวณหาผลกำไร แต่เนื่องจากขณะที่ทำการทดสอบนั้นยังไม่มีคำสั่งการโปรแกรมที่รองรับกระบวนการสุ่มตัวเลขบนหน่วยประมวลผลกราฟิกจึงแบ่งแนวคิดวิธีการสุ่มตัวเลขขึ้นมา 2 วิธี คือ

2.2.1 หน่วยประมวลผลกลางทำการสุ่มตัวเลขขึ้นมา จากนั้นทำการคัดลอกข้อมูลจากหน่วยความจำหลักไปยังหน่วยความจำแบบโกลบอลบนจีพียู ซึ่งการจำลองสถานการณ์ 1วันใช้ตัวเลขสุ่ม 2 ค่า (จำลองสถานการณ์ยอดขาย และจำนวนความต้องการของลูกค้า) ดังนั้นถ้าทำการทดสอบการจำลองสถานการณ์ 20วัน 10,000,000รอบคำนวณจะต้องทำการคัดลอกข้อมูลไปยังหน่วยความจำแบบโกลบอลบนจีพียูทั้งหมดจำนวน 2 *20*10,000,000 ค่า ซึ่งคิดเป็น 1.6 กิกกะไบต์ เป็นต้น

2.2.2 ปรับประยุกต์อัลกอริทึมในการสุ่มตัวเลขบนหน่วยประมวลผลกราฟิก โดยเริ่มต้นด้วยการให้หน่วยประมวลผลกลางสุ่มค่า Seed Time เพื่อใช้ในการสุ่มตัวเลขบนจีพียู จากนั้นทำการคัดลอกข้อมูลจากหน่วยความจำหลักไปยังหน่วยความจำแบบโกลบอลบนจีพียู ซึ่งการทำงานของแต่ละเทรดนั้นจะทำการสุ่มตัวเลขขึ้นมาเองตามอัลกอริทึมที่ได้ทำการปรับประยุกต์ไปบนจีพียู โดยออกแบบให้การคำนวณต่อ 1 จีพียูเทรดใช้ 1 Seed Time ดังนั้นถ้าแบ่งจำนวนเทรดที่คำนวณภายใน 1 บล็อกบนจีพียูเท่ากับ 20 เทรด แสดงว่า มีการใช้ข้อมูล Seed Time จำนวน 20 ค่าเช่นเดียวกัน

ปัจจัยการสุ่มข้อมูลนั้นมีผลต่อเวลาที่ใช้คัดลอกข้อมูลระหว่างหน่วยความจำหลักกับหน่วยความจำของจีพียู จากการทดสอบกรณีที่ {1, 3}, {2, 4}, {5, 7} และ{6, 8} ตามแนวคิดทั้ง 2 ของการสุ่มตัวเลข พบว่าการปรับอัลกอริทึมการสุ่มตัวเลขบนหน่วยประมวลผลกราฟิกใช้เวลาการประมวลผลที่เร็วกว่าเนื่องจาก 1) จำนวนครั้งการเข้าถึงข้อมูลตัวเลขสุ่มบริเวณหน่วยความจำแบบ โกลบอลบนจีพียูของแต่ละเทรดนั้นมีผลต่อสมรรถนะ ดังนั้นแนวคิดแรกที่ให้หน่วยประมวลผลกลางสุ่มข้อมูลแล้วทำการคัดลอกไปยังหน่วยความจำแบบโกลบอลบนจีพียู ทำให้แต่ละเทรดจำเป็นต้องเข้าถึงตัวเลขสุ่มบริเวณหน่วยความจำแบบโกลบอล 2 ครั้ง เพื่อนำค่าสุ่มไปเข้ากระบวนการจำลองสถานการณ์ยอดขาย และจำนวนความต้องการของลูกค้าใช้คำนวณหาผลกำไร เมื่อเทียบกับแนวคิดที่ 2 แต่ละเทรดเข้าถึงข้อมูลตัวเลขสุ่มบริเวณหน่วยความจำแบบโกลบอล เพียงแค่ 1 ครั้ง เพื่อนำค่า Seed Time ไปคำนวณการสุ่มตัวเลขบนจีพียูซึ่งเวลารวมที่ใช้ประมวลบนจีพียูของแนวคิดที่ 2 จึงเร็วกว่าแนวคิดที่1, 2) จำนวนข้อมูลตัวเลขสุ่มที่น้อยทำให้การคัดลอกข้อมูลจากหน่วยความหลักไปหน่วยความจำของจีพียูใช้เวลาที่สั้นตามไปด้วย และ3) เมื่อมีการประมวลผลด้วยจำนวนรอบการคำนวณ หรือจำนวนวันที่จำลองสถานการณ์ที่มากขึ้น ทำให้การสุ่มตัวเลขตามแนวคิดแรกต้องทำการสุ่มเป็นจำนวนมาก ซึ่งเวลาที่ใช้ประมวลผลบนซีพียูสำหรับการสุ่มส่งผลต่อการเร่งสมรรถนะความเร็วในการประมวลผลด้วย

2.3 การเข้าถึงข้อมูลของหน่วยความจำบนหน่วยประมวลผลกราฟิกมีผลต่อการเร่งสมรรถนะ จากการวิเคราะห์เปรียบเทียบการทดสอบกรณีที่ {1, 5}, {2, 6}, {3, 7} และ{4, 8} พบว่าการออกแบบให้เทรดสามารถเข้าถึงข้อมูลได้เฉพาะหน่วยความจำแบบโกลบอลบนจีพียูจะใช้เวลาประมวลผลบนจีพียูที่นานกว่าการออกแบบให้เทรดสามารถเข้าถึงข้อมูลบริเวณหน่วยความจำแบบโกลบอล และแชร์บนจีพียู เนื่องจากรอบของเวลา (Time period) ที่ใช้เข้าถึงข้อมูลผ่านหน่วยความจำแบบแชร์บนจีพียูสั้นกว่ารอบของเวลาที่ใช้เข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอลบนจีพียู แต่หน่วยความจำแบบแชร์บนจีพียูมีข้อจำกัดในเรื่องของขนาด และการเข้าใช้ข้อมูล

ร่วมกันเฉพาะเทรดที่อยู่ภายในบล็อกเดียวกันเท่านั้นที่สามารถใช้ข้อมูลผ่านหน่วยความจำแบบแชร์ได้ ดังนั้นการออกแบบเลือกบางส่วนของข้อมูลที่แต่ละเทรดภายในบล็อกจำเป็นต้องมีการเข้าถึงข้อมูลบ่อยๆนำไปไว้บริเวณหน่วยความจำแบบแชร์บนจีพียูจึงมีความสำคัญ ดังเช่นการทดสอบกรณีที่ 5, 6, 7 และ8 ที่ออกแบบให้เทรดแรกของแต่ละบล็อกทำการคัดลอกข้อมูลอินพุตบางส่วน (ค่าความน่าเป็นของจำนวนความต้องการของผู้ซื้อรายย่อย) จากหน่วยความจำแบบโกลบอลไปยังหน่วยความจำแบบแชร์ จากนั้นทุกเทรดภายในบล็อกจึงสามารถเข้าใช้ข้อมูลนี้ได้ ซึ่งใช้เวลาประมวลผลที่เร็วกว่าการให้ทุกเทรดเข้าถึงข้อมูลอินพุตเฉพาะบริเวณหน่วยความจำแบบโกลบอลเท่านั้นดังการทดสอบกรณีที่ 1, 2, 3 และ4 เนื่องจากข้อมูลอินพุตเป็นข้อมูลที่ทุกเทรดจำเป็นต้องใช้ในการจำลองสถานการณ์ขึ้นมาเพื่อคำนวณหาผลกำไร ทำให้แต่ละเทรดต้องเสียเวลาเข้าถึงหน่วยความจำบ่อยๆ ซึ่งถ้าออกแบบให้เทรดเข้าใช้ได้เพียงเฉพาะหน่วยความจำแบบโกลบอลบนจีพียูจะส่งผลต่อการเร่งสมรรถนะการประมวลผลแบบขนานได้แสดงดังภาพที่ 22 เป็นต้น

3. ผลทดสอบการออกแบบการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกสำหรับการแก้ปัญหา News Dealer บนเครื่องคอมพิวเตอร์ 1 เครื่อง ด้วยเทคนิคการเพิ่มประสิทธิภาพการประมวลผลบนหน่วยประมวลผลกราฟิก คือ 1) การออกแบบการคำนวณของจีพียูเทรด, 2) ปรับประยุกต์อัลกอริทึมในการสุ่มตัวเลขบนหน่วยประมวลผลกราฟิก และ3) การออกแบบให้เทรดที่อยู่ภายในบล็อกเดียวกัน มีการเข้าถึงข้อมูลบริเวณหน่วยความจำแบบแชร์บนจีพียู ทั้ง 3 เทคนิคสามารถเร่งสมรรถนะความเร็วได้ถึงประมาณ 4 เท่าเมื่อเปรียบเทียบกับการโปรแกรมแบบลำดับจากการกำหนดรอบการคำนวณเท่ากับ 10,000,000 รอบ ดังตารางที่ 12

จากผลการทดสอบทั้ง 8 กรณี ของการออกแบบการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก สำหรับการจำลองสถานการณ์ปัญหา News Dealer ทำให้ในงานวิจัยนี้มีแนวคิดที่จะพัฒนาอัลกอริทึมบนระบบจีพียูคลัสเตอร์ และปรับประยุกต์วิธีการเพิ่มประสิทธิภาพการประมวลผลกราฟิก กับการคำนวณหาคำตอบที่ต้องการพลังการคำนวณสูง ดังเช่น การคำนวณแบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด

แบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด

การทดสอบประสิทธิภาพการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาแบบจำลองสถานการณ์ PDPTW ขนาด 106, 212 และ422 นี้ ได้ออกแบบการทดสอบบนเครื่อง Compute-Node จำนวน 4 Compute-Node แต่ละ Compute-Node มีการประยุกต์การคำนวณการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกซึ่งหน่วยประมวลผลกราฟิกที่ใช้บนแต่ละ Compute-Node คือ Nvidia Tesla M2050 ประกอบด้วย 14 มัลติคอร์ แต่ละมัลติคอร์มี 32 คอร์

ศึกษา 3 ปัจจัยการเร่งสมรรถนะความเร็วการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์ จึงทำการทดสอบกับแบบจำลองสถานการณ์ปัญหา PDPTW ขนาดต่างๆ โดยออกแบบการทดสอบประสิทธิภาพเป็น 8 ส่วน คือ

1. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 1 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้เฉพาะบริเวณหน่วยความจำแบบโกลบอลบนจีพียูเท่านั้น

2. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 2 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้เฉพาะบริเวณหน่วยความจำแบบโกลบอลบนจีพียู เท่านั้น

3. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 3 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้เฉพาะบริเวณหน่วยความจำแบบโกลบอลบนจีพียู เท่านั้น

4. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 4 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้เฉพาะบริเวณหน่วยความจำแบบโกลบอลบนจีพียู เท่านั้น

5. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 1 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้ทั้งบริเวณหน่วยความจำแบบโกลบอล และแชร์ บนจีพียู

6. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 2 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้ทั้งบริเวณหน่วยความจำแบบโกลบอล และแชร์ บนจีพียู

7. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 3 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้ทั้งบริเวณหน่วยความจำแบบโกลบอล และแชร์ บนจีพียู

8. ทดสอบประสิทธิภาพการประมวลผลการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 บนเครื่อง Compute-Node 4 เครื่อง ด้วยการแบ่งจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียู ด้วยจำนวนที่ต่างกัน คือ 32, 64, 128 และกำหนดให้แต่ละเทรดเข้าถึงข้อมูลได้ทั้งบริเวณหน่วยความจำแบบโกลบอล และแชร์ บนจีพียู

ผลการทดสอบพบว่าเวลารวมที่ใช้ประมวลผลตามการออกแบบการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการจำลองสถานการณ์ปัญหา PDPTW ขนาด 106, 212 และ422 ใช้เวลาที่สั้นลงกว่าการคำนวณแบบลำดับแสดงดังตารางที่ 13 – 20 และสามารถวิเคราะห์ประสิทธิภาพการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิกได้จากการคำนวณ Speedup ที่เพิ่มขึ้นแสดงดังภาพที่ 23 – 25

ตารางที่ 13 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 1 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64, 128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น

Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	142.04	150.17	135.21
212	1380.60	1517.01	1731.14
422	21314.93	38775.73	37867.82

ตารางที่ 14 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 2 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64, 128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น

Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	135.95	139.14	135.67
212	838.64	853.08	862.96
422	12061.03	18637.36	18629.25

ตารางที่ 15 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 3 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64, 128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น

Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	128.71	126.90	135.99
212	792.53	819.77	854.73
422	8417.74	13810.23	13932.70

ตารางที่ 16 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 4 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64, 128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอลบนจีพียูเท่านั้น

Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	124.66	127.48	136.08
212	767.84	768.10	854.67
422	6062.87	8207.91	8557.12

ตารางที่ 17 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 1 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64, 128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และแชร์บนจีพียู

Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	171.75	115.26	121.48
212	2352.15	1459.71	2003.22
422	34302.41	19991.13	32893.87

ตารางที่ 18 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 2 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64, 128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และแชร์บนจีพียู

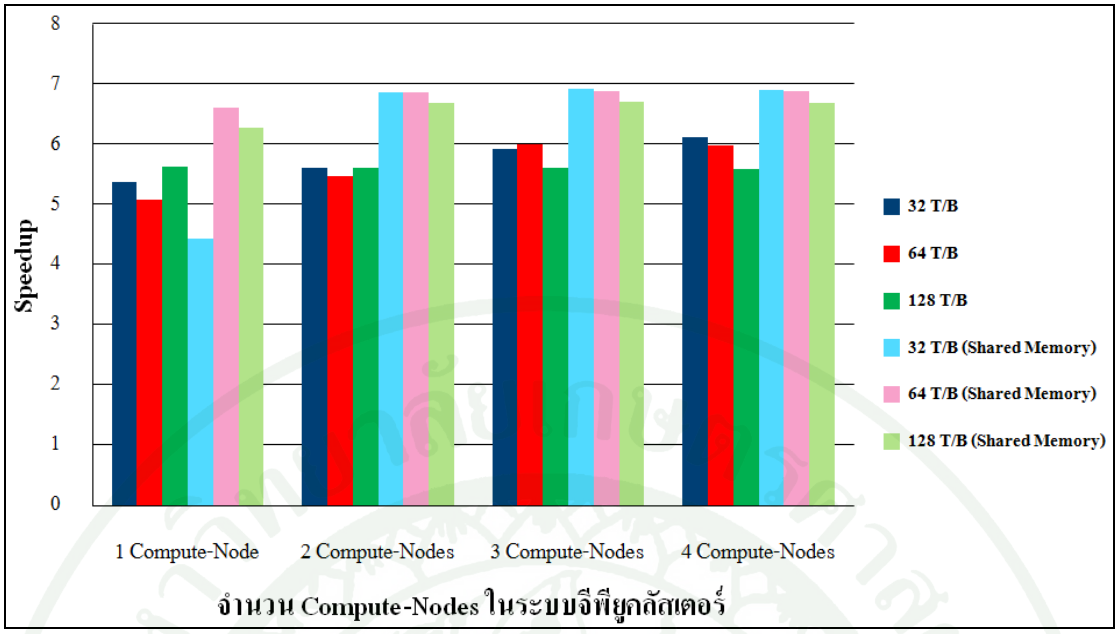
Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	110.94	110.98	113.90
212	1453.43	987.64	649.64
422	18252.56	11041.95	16972.06

ตารางที่ 19 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 3 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64,128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และแชร์บนจีพียู

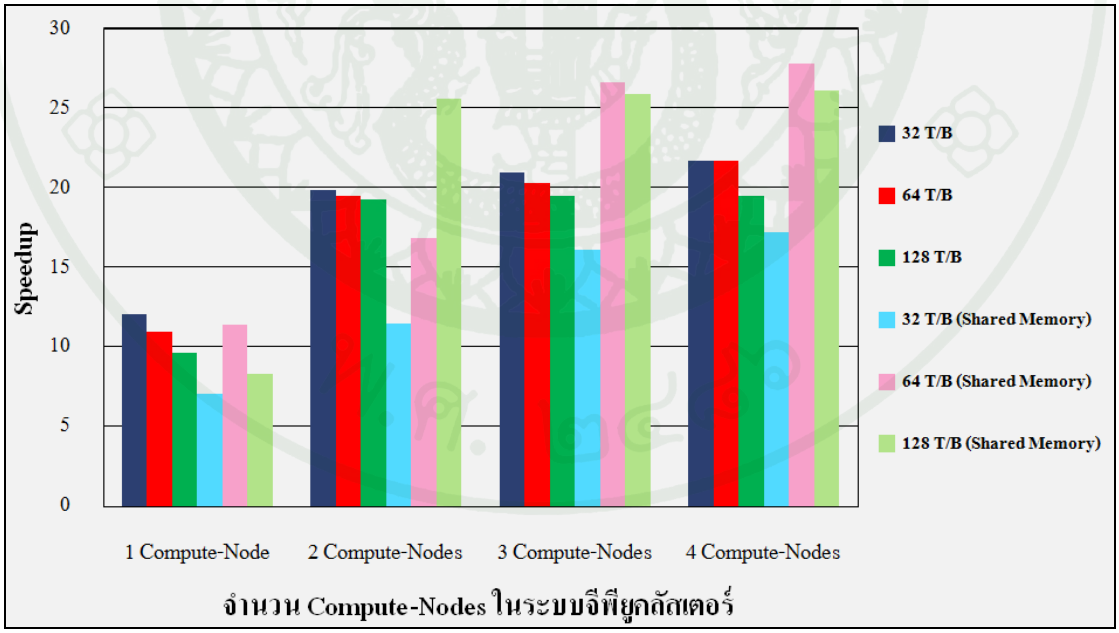
Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	110.09	110.53	113.66
212	1031.31	625.46	643.47
422	12300.72	6850.99	12094.65

ตารางที่ 20 เวลา (วินาที) ในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหา PDPTW บนเครื่อง Compute-Node จำนวน 4 เครื่อง ซึ่งแต่ละ Compute-Node มีการกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32, 64,128 และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำโกลบอล และแชร์บนจีพียู

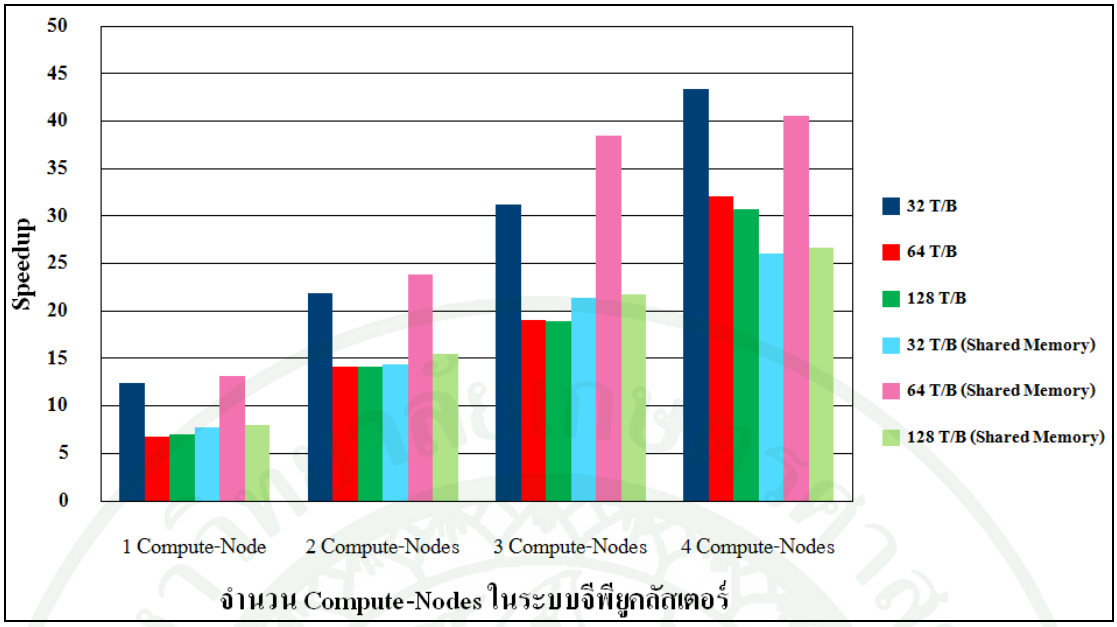
Problem Sizes	**Number of threads within block on GPU**		
(locations)	***32***	***64***	***128***
106	110.47	110.75	113.88
212	967.74	598.62	636.97
422	10084.05	6491.41	9886.76



ภาพที่ 23 กราฟ Speedup แสดงการทดสอบประสิทธิภาพการประมวลผลแบบขนานบนระบบจีพียูคลัส เตอร์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจำนวน 106 จุดรับ/สินค้า



ภาพที่ 24 กราฟ Speedup แสดงการทดสอบประสิทธิภาพการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจำนวน 212 จุดรับ/สินค้า



ภาพที่ 25 กราฟ Speedup แสดงการทดสอบประสิทธิภาพการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจำนวน 422 จุดรับ/สินค้า

จากผลการทดสอบประสิทธิภาพการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจำนวน 106, 212 และ422 จุดรับ/ส่งสินค้า สามารถคำนวณอัตราเร่งสมรรถนะ Speedup ระหว่างเวลาที่ใช้ประมวลผลของโปรแกรมแบบลำดับ และเวลาที่ใช้ในการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์ ดังสมการที่ 4 และวิเคราห์ผลการทดสอบดังนี้

$$Speedup = \frac{T_{Sequential}}{(T_{X1} + T_{X2}) + T_{Compute\text{-}Node}} \quad (4)$$

$T_{Sequential}$ คือ เวลาที่ใช้ประมวลผลของการโปรแกรมแบบลำดับ

T_{X1} คือ ระยะเวลาที่เครื่อง Front-End กระจายงานไปเครื่อง Compute-Node

T_{X2} คือ ระยะเวลาที่เครื่อง Compute-Node ส่งผลลัพธ์กลับไปเครื่อง Front-End

$T_{Compute\text{-}Node}$ คือ เวลาที่ใช้ประมวลผลของเครื่อง Compute-Node

1. จากผลการทดสอบการประมวลผลการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์ ใช้เวลาประมวลผลที่ลดลง เมื่อเทียบกับการประมวลผลการโปรแกรมแบบลำดับดังนี้

1.1 ปัญหา PDPTW ขนาด 106 จุดรับ/ส่งสินค้า จากการประมวลผลการโปรแกรมแบบลำดับใช้เวลา 762 วินาทีหรือประมาณ 13 นาที ลดลงเหลือ 110 วินาที หรือประมาณ 2 นาที เมื่อมีการประมวลผลการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์ซึ่งสามารถเร่งความเร็วการประมวลผลได้ถึง 7 เท่า

1.2 ปัญหา PDPTW ขนาด 212 จุดรับ/ส่งสินค้า จากการประมวลผลการโปรแกรมแบบลำดับใช้เวลา 16,645 วินาทีหรือประมาณ 278 นาที ลดลงเหลือ 598 วินาที หรือประมาณ 10 นาที เมื่อมีการประมวลผลการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์ซึ่งสามารถเร่งความเร็วการประมวลผลได้ถึง 28 เท่า

1.3 ปัญหาPDPTWขนาด 422 จุดรับ/ส่งสินค้า จากการประมวลผลการโปรแกรมแบบลำดับใช้เวลา 263,030 วินาทีหรือประมาณ 4,384 นาที ลดลงเหลือ 6,063 วินาที หรือประมาณ 102 นาที เมื่อมีการประมวลผลการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์ซึ่งสามารถเร่งความเร็วการประมวลผลได้ถึง 44 เท่า

การเร่งสมรรถนะความเร็วการประมวลผลการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์นั้น จากผลการทดสอบแสดงให้เห็นถึงขนาดของปัญหามีความสัมพันธ์กับจำนวน Compute-Node ที่ใช้ประมวลผล, จำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูของแต่ละ Compute-Node และการเข้าถึงข้อมูลบริเวณหน่วยความจำบนจีพียูของแต่ละเทรด มีผลต่อการเร่งสมรรถนะแสดงดังกราฟ Speedup

2. การประเมินประสิทธิภาพของโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกด้วยกราฟ Speedup แสดงให้เห็นถึงปัจจัยหลักการเร่งสมรรถนะความเร็วการประมวลผลดังนี้

2.1 การเพิ่มจำนวน Compute-Node มีผลต่อการเร่งสมรรถนะความเร็วการประมวลผลจากผลการทดสอบแสดงให้เห็นถึงกรณีที่ดีที่สุดของปัญหา PDPTW ที่มีขนาด 106 จุดรับ/ส่งสินค้า การประมวลผลโดยเครื่อง Compute-Node จำนวน 2 เครื่องใช้เวลาประมวลผลรวมทั้งสิ้น 110.94 วินาทีหรือประมาณเกือบ 2 นาที ซึ่งเมื่อเทียบกับการคำนวณโปรแกรมแบบลำดับบน

เครื่องคอมพิวเตอร์ 1 เครื่องใช้เวลาประมวลผลรวมทั้งสิ้น 762 วินาทีหรือประมาณ 13 นาที การประมวลผลโดยเครื่อง Compute-Node จำนวน 2 เครื่องสำหรับการแก้ปัญหา PDPTW ที่มีขนาด 106 จุดรับ/ส่งสินค้าสามารถเร่งสมรรถนะความเร็วไปถึง 7 เท่าซึ่งถ้าเพิ่มจำนวนเครื่อง Compute-Node มากขึ้นอัตราการเร่งสมรรถนะจะเพิ่มขึ้นในหลักหน่วยทศนิยม เนื่องจากปัญหา PDPTW ขนาด 106 จุดรับ/ส่งสินค้ามีจำนวนงานที่ประมวลผล คือ 1,378 งาน กระจายให้แต่ละ Compute-Node เป็นจำนวน 689 จำนวนเมื่อใช้ Compute-Node จำนวน 2 เครื่อง แต่ละ Compute-Node กำหนดการทำงานของเทรดประมวลผล 1งาน/1เทรด จึงมี 689 เทรดที่ประมวลผลแบบขนานบนจีพียูของแต่ละ Compute-Node ซึ่งการประมวลผลของเทรดบนจีพียูใช้รอบเวลาการคำนวณที่เร็ว แต่ละ Compute-Node จึงใช้เวลาประมวลผลที่สั้น ดังนั้นถ้าใช้ Compute-Node จำนวน 3 เครื่อง แต่ละ Compute-Node จะมีการคำนวณเทรดแบบขนานบนจีพียูทั้งหมดประมาณ 460 เทรด ซึ่งจำนวนเทรดที่ประมวลผลแต่ละ Compute-Node มีจำนวนต่างกันไม่มากเมื่อเทียบกับการใช้จำนวน Compute-Node 2 เครื่องประมวลผล จึงทำให้สมรรถนะความเร็วการประมวลผลโดยเครื่อง Compute-Node จำนวน 3 เครื่องเพิ่มขึ้นจากสมรรถนะความเร็วการประมวลผลโดยเครื่อง Compute-Node จำนวน 2 เครื่องเพียงแค่หลักหน่วยทศนิยม และสำหรับกรณีที่ดีที่สุดของปัญหา PDPTW ที่มีขนาด 212 และ422 จุดรับ/ส่งสินค้า คือการประมวลผลโดยใช้เครื่อง Compute-Node จำนวน 4 เครื่อง ปัญหาขนาด 212 จุดรับ/ส่งสินค้าใช้เวลาประมวลผลบนระบบจีพียูคลัสเตอร์ประมาณ 10 นาทีสามารถเร่งสมรรถนะความเร็วการประมวลผลได้ถึงประมาณ 28 เท่า, ปัญหาขนาด 422 จุดรับ/ส่งสินค้าใช้เวลาประมวลผลบนระบบจีพียูคลัสเตอร์ประมาณ 102 นาทีสามารถเร่งสมรรถนะความเร็วการประมวลผลได้ถึงประมาณ 44 เท่า จากผลการทดสอบกับปัญหาขนาด 212 และ422 แสดงให้เห็นถึงสมรรถนะความเร็วการประมวลผลที่เพิ่มขึ้นเมื่อทำการเพิ่มจำนวนเครื่อง Compute-Node ที่ใช้ประมวลผลในระบบจีพียูคลัสเตอร์ เนื่องจากปัญหา PDPTW ขนาด 212 และ422 จุดรับ/ส่งสินค้า มีจำนวนงานที่ประมวลผล คือ 5,565 และ22,155งาน ตามลำดับ ซึ่งเป็นจำนวนงานขนาดใหญ่ที่ต้องใช้พลังการคำนวณที่สูง การกระจายงานไปให้แต่ละ Compute-Node ที่อยู่ในระบบเท่ากันๆ จะช่วยลดเวลาประมวลผลได้ดีกว่าการประมวลผลเพียงเครื่องเดียว

การเพิ่มจำนวน Compute-Node ในการประมวลผลบนระบบจีพียูคลัสเตอร์สามารถเพิ่มสมรรถนะความเร็วการประมวลผลได้เมื่อกำหนดจำนวน Compute-Node ให้เหมาะสมกับขนาดปัญหา เพราะถ้ากำหนดจำนวน Compute-Node ที่ใช้ประมวลผลมากเกินไป สมรรถนะความเร็วก็อาจไม่ต่างไปจากการใช้จำนวน Compute-Node ที่น้อยกว่า ซึ่งถือเป็นการสิ้นเปลืองทรัพยากร

2.2 การกำหนดจำนวนเทรดภายในบล็อกสัมพันธ์กับจำนวนบล็อกที่ประมวลผลบนจีพียูของแต่ละ Compute-Node เนื่องจากสถาปัตยกรรมหน่วยประมวลผลกราฟิกรุ่น Nvidia Tesla M2050 ที่ใช้ในการทดสอบนี้มีจำนวนมัลติคอร์ทั้งสิ้น 14 มัลติคอร์ แต่ละมัลติคอร์มี 32 คอร์ ซึ่งการคำนวณ 1 บล็อกใช้การประมวลผลต่อ 1 มัลติคอร์ และการประมวลผลแต่ละเทรดในบล็อกนั้นจะประมวลผลโดยคอร์ที่เป็นส่วนประกอบของมัลติคอร์ แสดงว่ามีจำนวนบล็อกที่ทำการคำนวณแบบขนานบนจีพียู พร้อมกันมากสุด 14 บล็อก การคำนวณแต่ละมัลติคอร์จะสามารถเริ่มการคำนวณบล็อกถัดไปได้ต่อเมื่อทุกเทรดภายในบล็อกนั้นประมวลผลเสร็จสิ้น ดังนั้นถ้ากำหนดจำนวนเทรดภายในบล็อกที่มากเกินไป การคำนวณในแต่ละมัลติคอร์ต้องเสียเวลารอให้ทุกเทรดภายในบล็อกประมวลผลเสร็จสิ้นพร้อมกัน จึงจะสามารถคำนวณบล็อกถัดไปได้ แต่ถ้ากำหนดจำนวนเทรดภายในบล็อกที่น้อยเกินไป บล็อกมีจำนวนมากขึ้นการประมวลผลบนจีพียูอาจใช้เวลาประมวลผลมากขึ้นจากการสลับบล็อกขึ้นไปคำนวณหลายๆรอบบนแต่ละมัลติคอร์ซึ่งส่งผลต่อการเร่งสมรรถนะเช่นเดียวกัน

การกำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนหน่วยประมวลผลกราฟิกให้เหมาะสมกับจำนวนงานที่ประมวลผลบนแต่ละ Compute-Node สามารถเพิ่มสมรรถนะความเร็วการประมวลผลได้ จากผลการทดสอบแสดงให้เห็นถึงกรณีที่ดีที่สุดของปัญหา PDPTW ที่มีขนาด 106 จุดรับ/ส่งสินค้าจากการประมวลผลโดยเครื่อง Compute-Node จำนวน 2 เครื่อง กำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนแต่ละมัลติคอร์ของจีพียูอยู่ที่ระหว่าง 32 -64 เทรด/บล็อก ใช้เวลาประมวลผลประมาณ 2 นาที, กรณีที่ดีที่สุดของปัญหา PDPTW ที่มีขนาด 212 จุดรับ/ส่งสินค้าจากการประมวลผลโดยเครื่อง Compute-Node จำนวน 4 เครื่อง กำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนแต่ละมัลติคอร์ของจีพียูเท่ากับ 64 เทรด/บล็อก ใช้เวลาประมวลผลประมาณ 10 นาที และกรณีที่ดีที่สุดของปัญหา PDPTW ที่มีขนาด 422 จุดรับ/ส่งสินค้า จากการประมวลผลโดยเครื่อง Compute-Node จำนวน 4 เครื่อง กำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนแต่ละมัลติคอร์ของจีพียู เท่ากับ 32 เทรด/บล็อกใช้เวลาประมวลผลประมาณ 101 นาที

2.3 นอกจากปัจจัยการเพิ่มจำนวน Compute-Node และการกำหนดจำนวนเทรด/บล็อกที่ประมวลผลบนจีพียูมีผลต่อการเร่งสมรรถนะแล้ว การเพิ่มสรรถนะความเร็วการประมวลผลด้วยการจัดการการใช้หน่วยความจำบนจีพียูของแต่ละเทรดเป็นปัจจัยสำคัญที่สามารถเร่งสมรรถนะให้ดีขึ้นกว่าเดิม เนื่องจากการใช้เวลาเข้าถึงข้อมูลบริเวณหน่วยความจำแต่ละประเภทบนจีพียูนั้นไม่เท่ากัน รอบของเวลา(Time period) ที่ใช้เข้าถึงข้อมูลผ่านหน่วยความจำแบบแชร์บนจีพียูสั้นกว่ารอบของเวลาที่ใช้เข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอลบนจีพียู แต่หน่วยความจำแบบ แชร์บนจีพียูมีข้อจำกัดในเรื่องของขนาด และการเข้าใช้ข้อมูลร่วมกันเฉพาะเทรดที่อยู่ภายในบล็อกเดียวกันเท่านั้น

กระบวนการเข้าใช้หน่วยความจำบริเวณแชร์บนจีพียู เริ่มต้นด้วยการอนุญาตให้เทรดแรกของแต่ละบล็อกทำการคัดลอกข้อมูลจากบริเวณหน่วยความจำแบบโกลบอลไปยังหน่วยความจำแชร์ จากนั้นทุกเทรดภายในบล็อกถึงจะเริ่มเข้าถึงข้อมูลนั้นได้บริเวณหน่วยความจำแบบแชร์ แต่กำหนดสิทธิ์การถึงข้อมูลของแต่ละเทรดเพียงแค่อ่านได้เท่านั้นไม่สามารถเปลี่ยนแปลงค่าของข้อมูลได้เพื่อป้องกันการเกิดปัญหา Data Hazard

กำหนดข้อมูลที่แต่ละเทรดใช้ร่วมกันบริเวณหน่วยความจำแบบแชร์บนจีพียูสำหรับการทดสอบนี้ คือ ข้อมูลอินพุต เนื่องจากการคำนวณแต่ละเทรดมีการเข้าใช้ข้อมูลอินพุตบ่อยๆ จึงเลือกให้เทรดแรกของแต่ละบล็อกคัดลอกข้อมูลอินพุตจากหน่วยความจำแบบโกลบอลไปยังหน่วยความจำแบบแชร์ ทำให้เทรดอื่นๆภายในบล็อกใช้เวลาประมวลผลที่ลดลงกว่าการเข้าถึงข้อมูลบริเวณหน่วยความจำแบบโกลบอลเพียงอย่างเดียว

การเพิ่มสมรรถนะการประมวลผลด้วยการจัดการการใช้หน่วยความจำบนจีพียูของแต่ละเทรดจะให้ผลลัพธ์ที่ดีขึ้นอยู่กับ2ปัจจัยแรกที่กล่าวมาด้วย คือ การเพิ่มจำนวน Compute-Node ที่ใช้ประมวลผลบนระบบจีพียูคลัสเตอร์ และการกำหนดจำนวนเทรด/บล็อกที่ประมวลผลบนจีพียู ดังนั้นถ้าจำนวนงานที่ประมวลผลบนแต่ละ Compute-Node มีขนาดใหญ่ และแบ่งจำนวนเทรด/บล็อกบนจีพียูที่น้อยเกินไป การใช้หน่วยความจำแบบแชร์บนจีพียูของแต่ละเทรดจะไม่สามารถเพิ่มสมรรถนะให้ดีกว่าการจัดการให้แต่ละเทรดทำการเข้าถึงข้อมูลได้เฉพาะบริเวณหน่วยความจำแบบโกลบอลบนจีพียู เนื่องจากการแบ่งจำนวนเทรด/บล็อกบนจีพียูที่น้อยเกินไป สำหรับงานขนาดใหญ่บน Compute-Node ทำให้มีคิวของบล็อกจำนวนมากที่ต้องคำนวณซึ่งทำให้เสียเวลาสลับบล็อกเข้าไปคำนวณบนแต่ละมัลติคอร์ และแต่ละบล็อกต้องเสียเวลารอให้เทรดแรกของบล็อกคัดลอกข้อมูลจากหน่วยความจำแบบโกลบอล ไปยังหน่วยความจำแบบแชร์ทุกเทรดภายในบล็อกจึงจะสามารถประมวลผลได้ แต่ถ้าแบ่งจำนวนเทรด/บล็อกบนจีพียูที่มากเกินไปสำหรับงานขนาดใหญ่บน Compute-Node ทำให้มีจำนวนบล็อกที่ต้องคำนวณบนจีพียูน้อยลงแต่จะไม่สามารถเร่งสมรรถนะได้ดีถ้ามีการใช้หน่วยความจำบริเวณแชร์ เนื่องจากการเวลาที่ใช้คำนวณบนแต่ละมัลติคอร์จะเสียเวลารอให้ทุกเทรดภายในบล็อกประมวลผลเสร็จจึงจะสามารถคำนวณบล็อกอื่นๆได้ และสำหรับงานขนาดเล็ก สามารถกำหนดจำนวนเทรด/บล็อกบนจีพียูเท่าใดก็ได้ ก็สามารถเร่งสมรรถนะความเร็วการประมวลผลได้เมื่อมีการใช้หน่วยความจำแบบแชร์บนจีพียู เนื่องการประมวลของเทรดบนจีพียูมีการคำนวณที่เร็วยิ่งงานมีจำนวนน้อย และมีการใช้หน่วยความจำแบบแชร์บนจีพียูยิ่งทำให้การประมวลผลของงานเสร็จเร็ว จึงสามารถเร่งสมรรถนะความเร็วการประมวลผลได้

3. ผลทดสอบการออกแบบการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาการจัดเส้นทางพาหนะที่มีจุดรับ/ส่งหลายจุด ปัจจัยทั้ง 3 ที่กล่าวมานั้นมีความสัมพันธ์กับขนาดของปัญหา PDPTW ซึ่งส่งผลต่อการเร่งสมรรถนะความเร็วการประมวลผล และผลลัพธ์เวลาที่ดีที่สุดของการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด แสดงดังนี้

3.1 กรณีที่ดีที่สุดของการแก้ปัญหา PDPTW ขนาด 106 จุดรับ/ส่งสินค้า คือ การคำนวณโดยใช้เครื่อง Compute-Node จำนวน 2 เครื่อง แต่ละ Compute-Node กำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32 หรือ64 เทรด/บล็อก และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอล และแชร์บนจีพียู ซึ่งสามารถเร่งความเร็วการประมวลผลได้ถึง 7 เท่า

3.2 กรณีที่ดีที่สุดของการแก้ปัญหา PDPTW ขนาด 212 จุดรับ/ส่งสินค้า คือ การคำนวณโดยใช้เครื่อง Compute-Node จำนวน 4 เครื่อง แต่ละ Compute-Node กำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 64 เทรด/บล็อก และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอล และแชร์บนจีพียู ซึ่งสามารถเร่งความเร็วการประมวลผลได้ถึง 28 เท่า

3.3 กรณีที่ดีที่สุดของการแก้ปัญหา PDPTW ขนาด 422 จุดรับ/ส่งสินค้า จากการคำนวณโดยใช้เครื่อง Compute-Node จำนวน 4 เครื่อง แต่ละ Compute-Node กำหนดจำนวนเทรดภายในบล็อกที่ประมวลผลบนจีพียูจำนวน 32 เทรด/บล็อก และอนุญาตให้แต่ละเทรดเข้าถึงข้อมูลผ่านหน่วยความจำแบบโกลบอลบนจีพียูเท่านั้น ซึ่งสามารถเร่งความเร็วการประมวลผลได้ถึง 44 เท่า

สรุป

วิทยานิพนธ์นี้ศึกษาแนวคิดการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิกเพื่อนำไปปรับประยุกต์ออกแบบการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการหาผลลัพธ์ของแบบจำลองสถานการณ์ต่างๆ ในภาคธุรกิจที่ต้องการพลังการคำนวณสูง ในงานวิจัยนี้เสนอแนวคิดการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์สำหรับการจำลองสถานการณ์ไม่ต่อเนื่อง เพื่อศึกษาปัจจัยที่มีผลกระทบต่อการเร่งสมรรถนะการประมวลผลบนหน่วยประมวลผลกราฟิกจึงได้ทำออกแบบการทดสอบ 3 ปัจจัยหลักที่ส่งผลต่อความเร็วประมวลผลบนหน่วยประมวลผลกราฟิกทั้งการกำหนดการทำงานของเทรด, การสุ่มตัวเลข และการเข้าถึงข้อมูลบนหน่วยประมวลผลกราฟิก ซึ่งจากผลทดสอบประสิทธิภาพการประมวลผลแบบขนานบนหน่วยประมวลกราฟิกกับการจำลองสถานการณ์ปัญหาระบบคลังสินค้าสามารถเร่งความเร็วการประมวลผลได้ถึง 4 เท่า และจากการทดสอบนี้เองทำให้ผู้วิจัยมีแนวคิดในการพัฒนาการคำนวณแบบขนานบนระบบจีพียูคลัสเตอร์ สำหรับการจำลองสถานการณ์ปัญหาการจัดเส้นทางยานพาหนะขนาดต่างๆ เพื่อเป็นแนวทางการประยุกต์ใช้ความสามารถในการประมวลผลแบบขนานบนระบบจีพียูคลัสเตอร์กับงานที่ต้องการใช้พลังการคำนวณสูง ซึ่งจากผลการทกสอบประสิทธิภาพของการจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งจำนวน 106, 212 และ422 พบว่า การประมวลผลบนระบบจีพียูคลัสเตอร์ สามารถเร่งสมรรถนะความเร็วการประมวลผลได้ถึง 7 เท่ากับปัญหาขนาด 106, 28 เท่ากับปัญหาขนาด 212 และ 44 เท่ากับปัญหาขนาด 422 ตามลำดับ ซึ่งการเพิ่มจำนวน Compute-Node ในระบบ, การจัดการเทรด และการเข้าถึงข้อมูลบนหน่วยประมวลผลกราฟิก เป็นปัจจัยหลักในการเร่งสมรรถนะความเร็วในการประมวลผลบนระบบจีพียูคลัสเตอร์

งานวิจัยนี้ได้ออกแบบแนวทางการโปรแกรมแบบขนานบนระบบจีพียูคลัสเตอร์กับการคำนวณหาผลลัพธ์แบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งสินค้าหลายจุด โดยมีการพัฒนาส่วนเชื่อมต่อกับผู้ใช้ผ่านไมโครซอฟต์เอ็กเซลเพื่อให้สะดวกแก่การส่งงานเข้าไปประมวลผลบนระบบจีพียูคลัสเตอร์ แนวทางการพัฒนาเพิ่มเติมออกแบบการคำนวณให้แต่ละ Compute-Node มีการคำนวณแบบขนานของหน่วยประมวลผลกราฟิกตั้งแต่ 2 การ์ดขึ้นไป และยังคงสามารถทำงานร่วมกันได้บนระบบจีพียูคลัสเตอร์ ศึกษาการเร่งสมรรถนะความเร็วในการประมวลผลเพื่อวางแนวทางในการสร้าง Super Computer ต่อไป

เอกสารและสิ่งอ้างอิง

Adam, J., J. Wladyslaw and M. Lichtenstein. 2008. Tabu Search on GPU. **International Journal of Universal Computer Science** 14: 2416-2427.

Antonie, L., Y. Mati and Z. Binder. 2001. A tabu search heuristic for the single vehicle pickup and delivery problem with time windows. **Intelligent Manufacturing** 12: 497–508.

Baker, M. 2000. **Cluster Computing White Paper.** University of Portsmouth, United Kingdom.

Barney, B. 2007. **Introduction to Parallel Computing**. Lawrence Livermore National Laboratory. Available Source: https://computing.llnl.gov/tutorials/parallel_comp/, May 26, 2008.

Bill, D. 2009. **The Future of GPU Computing**. Available Source: http://www.nvidia.com, November 18, 2009.

Enhua, L. 2008. Emerging technology about GPGPU, pp. 618-622. *In* **IEEE Asia Pacific Conference Circuits and Systems APCCAS 2008**. McHall, New York.

Gray, M.A. 2009. Getting Started with GPU Programming. **Computing in Science & Engineering** 11(4): 61-64.

Hamdy, A.T. 2003. **OPERATIONS RESEARCH**. University of Arkansas Fayetteville.

Hong, L. and H. Linda. 2008. Efficient Parallelization of Stochastic Simulation Algorithm for Chemically Reacting Systems on the Graphics Processing Unit. **High Performance Computing Applications** 14: 2416-2427.

Hoong, C.L. and Z. Liang. 2001. Pickup and Delivery with Time Windows : Algorithms and Test Case Generation, pp. 333-340. *In* **Proceedings of the 13th International Conference on Tools with Artificial Intelligence.** Dallas, TX, USA.

Kalyan, S.P. 2006. Discrete-event Execution Alternatives on General Purpose Graphical Processing Unit (GPGPU), pp. 74-81. *In* **Proceedings of the 20th Workshop on Principle of Advance and Distributed Simulation (PADS'06)**. Washington DC, USA.

Kumar, V., A. Grama, A. Gupta and G. Karypis. 1994. **Introduction to Parallel Computing: Design and Analysis of Algorithms.** The Benjamin/Cummings Publishing Company, Inc., California.

Lao, H.C. and Z. Liang. 2001. Pickup and Delivery with Time Windows: Algorithms and Test Case Generation, pp. 455-472. *In* **Proceeding of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI).** Dallas, TX, USA.

Li, H. and A. Lim. 2001. A Metaheuristic for the Pickup and Delivery Problem with Time Windows, pp. 160-167. *In* **Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI).** Dallas, TX, USA.

_____. 2004. **Benchmarks Vehicle Routing and Travelling Salesperson Problems.** Available Source: http://www.top.sintef.no/vrp/benchmarks.html, May 10, 2008.

Mark, H. 2008. **CUDA Fluid Simulation in NVIDIA PhysX.** NVIDIA Developer Technology, USA.

NVIDIA Corporation. 2010. **NVIDIA CUDA C Programming Guide.** NVIDIA Corporation, USA.

Pharr, M. and R. Fernando. 2005. **GPU Gems 2: Programming Techniques for High-**

Performance Graphics and General-Purpose Computation. Addison Wesley Professional, New York.

Petruzzi, N.C. and M. Dada. 1999. Pricing and the Newsvendor Problem: A Review with Extensions. **Operations Research archive** 47(2): 183-194.

Quan, L. and M.M. Dessouky. 2006. A New Insertion-based Construction Heuristic for Solving the Pickup and Delivery Problems with Time Windows. **European Journal of Operational Research** 175(2): 672-678.

Quinn, M.J. 2003. **Parallel Programming in C with MPI and OpenMP.** McGraw-Hill, New York.

Savelsbergh, M.W.P. and M. Sol. 1995. The General Pickup and Delivery Problem. **Transportation Science** 29(1): 17-29.

Shigeyoshi, T. and N. Fujimoto. 2011. Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study, pp. 35-39. *In* **Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference.** New York, NY, USA.

Silver, E.A., D.F. Pyke. and R.P. Peterson. 1998. **Inventory Management and Production Planning and Scheduling**. 3rd ed. John Wiley & Sons Pte Ltd., New York.

Steve, P. and K. Miller. 1998. Random Number Generators: Good Ones Are Hard To Find. **Communications of the ACM** 31(10): 65-80.

Tom, R.H. 2008. Parallel Processing With CUDA. **Microprocessor Report.** 58: 15-19.

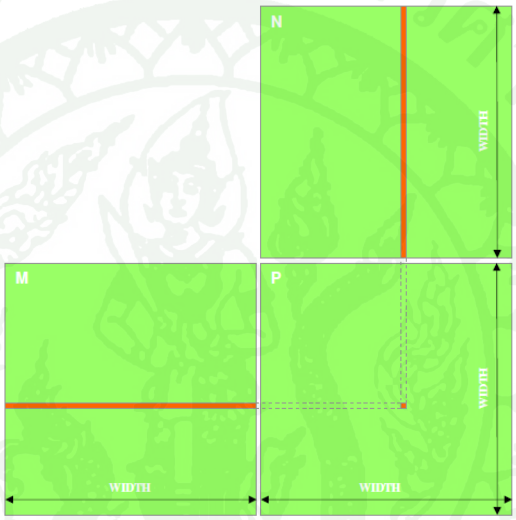
ภาคผนวก

ภาคผนวก ก

แนวคิดการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก

แนวคิดการโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก

การออกแบบการเขียนโปรแกรมแบบขนานบนหน่วยประมวลผลกราฟิก (NVIDIA CUDA C Programming Guide, 2010) ควรเลือกส่วนของโปรแกรมที่ไม่มีความเป็นขนานของข้อมูล (data parallelism) ให้ถูกประมวลผลในฝั่งโฮสต์ ส่วนที่มีความเป็นขนานของข้อมูลมากๆ นั้นควรถูกประมวลผลในฝั่งอุปกรณ์ ในที่นี้จะยกตัวอย่างแนวคิดการเขียนโปรแกรมการคูณแมทริกซ์ M x N ดังภาพผนวกที่ ก1



ภาพผนวกที่ ก1 แนวคิดการคูณแมทริกซ์ M X N แบบขนาน

ที่มา: NVIDIA CUDA C Programming Guide (2010)

ขั้นตอนการทำงานของโปรแกรม เริ่มต้นทำการจองข้อมูลแมทริกซ์บนหน่วยความจำของหน่วยประมวลผลกราฟิกสำหรับแมทริกซ์ M, N และ P ทำการย้ายข้อมูลจากโฮสต์ไปยังอุปกรณ์ ก่อนการประมวลผลการคูณแมทริกซ์จะเริ่มต้นขึ้นบนหน่วยประมวลผลกราฟิก เมื่อทุกเทรดประมวลผลบนหน่วยประมวลผลกราฟิก และเขียนผลลัพธ์ลงแมทริกซ์ P เสร็จสิ้น โฮสต์จะทำการอ่านข้อมูลแมทริกซ์ P กลับมายังหน่วยความจำหลัก และทำการคืนพื้นที่บนหน่วยความจำบนหน่วยประมวลผลกราฟิก

การรับส่งข้อมูลระหว่างโฮสต์ และอุปกรณ์นั้นจำเป็นต้องมีการจองพื้นที่บริเวณหน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิก โดยการเรียกใช้ฟังก์ชัน cudaMalloc ซึ่งต้องระบุขนาดพื้นที่การจองไปด้วย การเรียกใช้ฟังก์ชัน cudaMemcpy ทำให้สามารถส่งข้อมูลระหว่างโฮสต์และอุปกรณ์ได้โดยต้องระบุข้อมูลต้นทาง (Source), ข้อมูลปลายทาง (Destination), จำนวนไบต์ (Bytes) ที่ทำการคัดลอกข้อมูล และประเภทของการส่งข้อมูลซึ่งประกอบด้วย Host to Host, Host to Device, Device to Host, Device to Device จากนั้นเมื่อใช้พื้นที่หน่วยความจำแบบโกลบอลบนหน่วยประมวลผลกราฟิกเสร็จสิ้น ต้องทำการคืนพื้นที่ที่จองไว้ด้วยการเรียกใช้ฟังก์ชัน cudaFree

วิธีการโปรแกรมการจองพื้นที่บนหน่วยความจำหลัก และหน่วยความจำบนหน่วยประมวลผลกราฟิกเพื่อทำการกำหนดค่าให้แมทริกซ์ แสดงดังภาพผนวกที่ ก2 และ ก3 ตามลำดับ

```
int size = Width * Width * sizeof(float);
float *Mhost = (float *)malloc(size);
```

ภาพผนวกที่ ก2 คำสั่งการจองหน่วยความจำหลักของโฮสต์ (Host Memory)

ที่มา: NVIDIA CUDA C Programming Guide (2010)

```
float *Mdevice;
cudaMalloc((void**)&Mdevice,size);
```

ภาพผนวกที่ ก3 คำสั่งการจองหน่วยความจำแบบโกลบอลของอุปกรณ์ (Device Memory)

ที่มา: NVIDIA CUDA C Programming Guide (2010)

ทำการโปรแกรมการจองหน่วยความจำแบบโกลบอลดังภาพผนวกที่ ก3 กับแมทริกซ์ M, N และP จากนั้นกำหนดค่าให้กับแมทริกซ์ M, N และP บนหน่วยความจำหลักเพื่อทำการคัดลอกข้อมูลไปยังหน่วยความจำแบบโกลบอลของอุปกรณ์ ดังภาพผนวกที่ ก4

```
cudaMemcpy(Mdevice, Mhost, size, cudaMemcpyHostToDevice);
```

ภาพผนวกที่ ก4 คำสั่งคัดลอกข้อมูลจากหน่วยความจำหลักของโฮสต์ไปยังหน่วยความจำแบบโกลบอลของอุปกรณ์

ที่มา: NVIDIA CUDA C Programming Guide (2010)

เมื่อเสร็จสิ้นการประมวลแบบขนานผลบนหน่วยประมวลผลกราฟิกจะทำการคัดลอกข้อมูลของแมทริกซ์ P จากหน่วยความจำแบบโกลบอลของอุปกรณ์ไปยังหน่วยความจำหลักของโฮสต์ด้วยการโปรแกรมดังภาพผนวกที่ ก5

```
cudaMemcpy(Phost, Pdevice, size, cudaMemcpyDeviceToHost);
```

ภาพผนวกที่ ก5 คำสั่งคัดลอกข้อมูลจากหน่วยความจำแบบโกลบอลของอุปกรณ์ไปยังหน่วยความจำหลักของโฮสต์

ที่มา: NVIDIA CUDA C Programming Guide (2010)

จากฟังก์ชันการทำงานที่ได้อธิบายดังภาพผนวกที่ ก2 – ก5 สามารถเขียนโปรแกรมการคูณแมทริกซ์ให้สามารถประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก ดังภาพผนวกที่ ก6

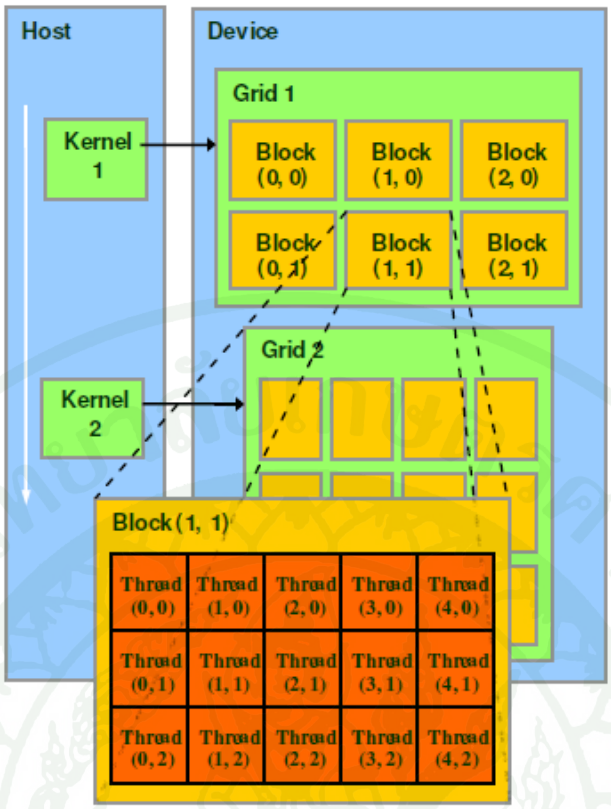
```
__global__ void MatrixMulKernel(float* Md, float* Nd, float* Pd, int
Width)
{
      // 2D Thread ID
      int tx = threadIdx.x;
      int ty = threadIdx.y;
      // Pvalue stores the Pd element that is computed by the thread
      float Pvalue = 0;
      for (int k = 0; k < Width; ++k)
      {
      float Mdelement = Md[ty * Md.width + k];
      float Ndelement = Nd[k * Nd.width + tx];
      Pvalue += Mdelement * Ndelement;
      }
      // Write the matrix to device memory each thread writes one
element
      Pd[ty * Width + tx] = Pvalue;
}
```

ภาพผนวกที่ ก6 การเขียนโปรแกรมการคูณแมทริกซ์

ที่มา: NVIDIA CUDA C Programming Guide (2010)

ฟังก์ชัน __global__ เป็นฟังก์ชันเคอร์เนลที่ถูกเรียกจากโฮสต์เพื่อเรียกส่วนการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก, ฟังก์ชัน threadIdx.x และthreadIdx.y เป็นฟังก์ชันที่แสดงตำแหน่งของเทรดในแนวคิดแบบลูกบาศก์ การทำงานของแต่ละเทรดใช้การโปรแกรมเดียวกันกับบนการคำนวณบนหน่วยประมวลผลกราฟิกจึงต้องมีส่วนที่สามารถแบ่งแยกได้ว่าเทรดไหนเป็นเทรดไหนโดยการใช้ฟังก์ชัน threadIdx.x และthreadIdx.y นั่นเอง

การโปรแกรมดังภาพผนวกที่ ก6 ในฟังก์ชัน __global__ แต่ละเทรดเริ่มต้นทำงานจากการรับค่าตำแหน่งในแกน x และแกน y จากฟังก์ชัน threadIdx.x และthreadIdx.y ตามลำดับ จากนั้นประกาศตัวแปร Pvalue เพื่อเก็บค่าผลบวกของการคูณแมทริกซ์ ทำการคำนวณโดยการวนไล่ตามจำนวนแถวของแมทริกซ์ M และจำนวนคอลัมน์ของแมทริกซ์ N เพื่อนำแต่ละตำแหน่งของแมทริกซ์มาคูณกัน เช่น $Thread_{2,3}$ หมายถึงการทำ “ Dot Product ” ระหว่างแถวที่ 2 ของแมทริกซ์ M และคอลัมน์ที่ 3 ของแมทริกซ์ N จากนั้นนำผลลัพธ์ที่ได้นำไปใส่แมทริกซ์ P ในตำแหน่งที่ (2,3) ซึ่งตำแหน่งได้จากการคำนวณ ty * Md.width + tx ซึ่งเสมือน ty เป็นหมายเลขแถว และtx เป็นหมายเลขคอลัมน์ของแมทริกซ์ P ดังภาพผนวกที่ ก7



ภาพผนวกที่ ก7 ภาพจำลองโครงสร้างกริด, บล็อก และเทรด

ที่มา: NVIDIA CUDA C Programming Guide (2010)

ภาพผนวกที่ ก7 แสดงให้เห็นการประมวลผลแบบขนานบนหน่วยประมวลผลกราฟิก เริ่มต้นขึ้นเมื่อฟังก์ชันเคอร์เนลถูกเรียกใช้งาน ซึ่งกริดเทียบได้กับการเรียกฟังก์ชันเคอร์เนล 1 ครั้ง ประกอบไปด้วยบล็อกที่มีการอ้างตำแหน่งด้วยฟังก์ชัน blockIdx.x และblockIdx.y แต่ละบล็อกประกอบด้วยจำนวนเทรดเท่าๆกันเพื่อประสิทธิภาพในการจัดการ

ภาคผนวก ข

อัลกอริทึมการแก้ปัญหา News Dealer

อัลกอริทึมการแก้ปัญหา News Dealer

การหาผลลัพธ์ให้กับแบบจำลองสถานการณ์ปัญหา News Dealer คำนวณโดยการนำผลลัพธ์ที่เป็นไปได้ทั้งหมดเข้าไปคำนวณกับโมเดลคณิตศาสตร์ที่ถูกจำลองขึ้นมาจากการสุ่มตัวเลขและเพิ่มจำนวนรอบการคำนวณเพื่อหาผลกำไรเฉลี่ยในทุกๆผลลัพธ์ที่เป็นไปได้ จากนั้นเลือกผลลัพธ์ที่ให้ค่าผลกำไรเฉลี่ยสูงสุดมาเป็นคำตอบที่เหมาะสมที่สุดให้กับแบบจำลองสถานการณ์

กำหนดพารามิเตอร์ ก่อนการคำนวณการแก้ปัญหา News Dealer ดังนี้

ตารางผนวกที่ ข1 การกำหนดพารามิเตอร์ สำหรับปัญหา News Dealer

ลำดับที่	ชื่อตัวแปร	ความหมาย
1.	Good_Day	ค่าสถิติโอกาสความน่าเป็นการเกิดสถานการณ์ “ยอดขายดี”
2.	Fair_Day	ค่าสถิติโอกาสความน่าเป็นการเกิดสถานการณ์ “ยอดขายปกติ”
3.	Poor_Day	ค่าสถิติโอกาสความน่าเป็นการเกิดสถานการณ์ “ยอดขายแย่”
4.	Purchase_P	ราคาต้นทุนของหนังสือพิมพ์ 1 ฉบับ
5.	Selling_P	ราคาขายของหนังสือพิมพ์ 1 ฉบับ
6.	Scrap_P	ราคาขายคืนให้โรงพิมพ์ของหนังสือพิมพ์ 1 ฉบับ
7.	Demand_P	ความต้องการหนังสือพิมพ์ของผู้ซื้อรายย่อย
8.	Day	จำนวนวัน ที่ใช้จำลองสถานการณ์ล่วงหน้า
9.	Iteration	จำนวนรอบการคำนวณ
10.	Num	จำนวนคำตอบที่เป็นไปได้
11.	Avg	อาร์เรย์ขนาด Num จัดเก็บผลกำไรเฉลี่ย ของแต่ละคำตอบที่เป็นไปได้

การหาคำตอบสำหรับการแก้ปัญหา News Dealer เพื่อหาจำนวนการสั่งหนังสือพิมพ์ล่วงหน้าที่ให้ค่าผลกำไรสูงสุด มีขั้นตอนดังนี้

1. กำหนดค่าเป็นเปอร์เซ็นต์สำหรับการเกิดสถานการณ์ยอดขายใน 1 วัน (Good_Day, Fair_Day และ Poor_day) แสดงดังภาพผนวกที่ ข1 เพื่อใช้เป็นเกณฑ์การสุ่มตัวเลข การเกิดสถานการณ์ต่างๆ

Type of Newsday	
Type	**Probability**
Good	**0.35** ← *Temp*1
Fair	**0.45** ← *Temp*2
Poor	**0.20** ← *Temp*3

$$Good\_Day = Temp1 * 100$$
$$Fair\_Day = (Temp1 + Temp2) * 100$$
$$Poor\_Day = (Temp1 + Temp2 + Temp3) * 100$$

ภาพผนวกที่ ข1 วิธีคำนวณเปอร์เซ็นต์การเกิดสถานการณ์ของยอดขาย

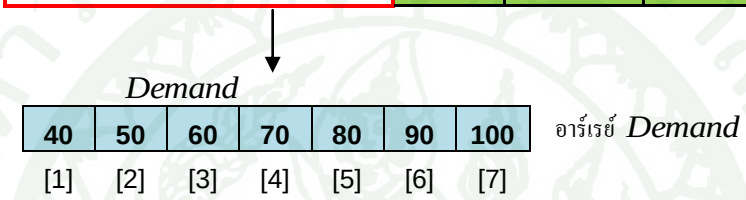
2. กำหนดราคาต้นทุน, ราคาขาย และราคาขายคืนของหนังสือพิมพ์ 1 ฉบับให้กับตัวแปร Purchase_P, Selling_P และ Scrap_P ตามลำดับ แสดงดังภาพผนวกที่ ข2

General Parameters	
Purchase Price	**$0.33** ← *Purchase_P*
Selling Price	**$0.50** ← *Selling_P*
Scrap Value	**$0.05** ← *Scrap_P*

ภาพผนวกที่ ข2 ข้อมูลราคาซื้อ ขายหนังสือพิมพ์ 1 ฉบับ

3. กำหนดให้ Demand คือ อาร์เรย์ขนาด Num ที่จัดเก็บข้อมูล “จำนวนความต้องการหนังสือพิมพ์ของผู้ซื้อรายย่อย” แสดงดังภาพผนวกที่ ข3

Distribution of Newspapers Demanded			
Demand	Demand Probabilities		
	Good	Fair	Poor
40	0.03	0.10	0.44
50	0.05	0.18	0.22
60	0.15	0.40	0.16
70	0.20	0.20	0.12
80	0.35	0.08	0.06
90	0.15	0.04	0.00
100	0.07	0.00	0.00



ภาพผนวกที่ ข3 การจัดเก็บข้อมูลลงอาร์เรย์ Demand เมื่อ Num = 7

4. กำหนดให้ P_Demand คือ อาร์เรย์ขนาด 3 X Num จัดเก็บข้อมูล “เปอร์เซ็นต์จำนวนความต้องการหนังสือพิมพ์ของผู้ซื้อรายย่อย เมื่อเกิดสถานการณ์ยอดขายต่างๆ” โดยให้แถวของอาร์เรย์แทน “สถานการณ์ยอดขาย” และคอลัมน์ของอาร์เรย์แทน “เปอร์เซ็นต์จำนวนความต้องการ” แสดงดังภาพผนวกที่ ข4

Distribution of Newspapers Demanded		
Demand	Demand Probabilities	
	Good	
40	0.03	← *Temp*1
50	0.05	← *Temp*2
60	0.15	← *Temp*3
70	0.20	← *Temp*4
80	0.35	← *Temp*5
90	0.15	← *Temp*6
100	0.07	← *Temp*7

$$P\_Demand[1][1] = Temp1 * 100$$
$$P\_Demand[1][2] = (Temp1 + Temp2) * 100$$
$$P\_Demand[1][3] = (Temp1 + Temp2 + Temp3) * 100$$
....
$$P\_Demand[1][7] = \sum_{i=1}^{7} Temp_i * 100$$

i. วิธีการคำนวณเปอร์เซ็นต์จำนวนความต้องการหนังสือพิมพ์ของผู้ซื้อรายย่อย เมื่อเกิดสถานการณ์ “ยอดขายดี” ใช้วิธีการคำนวณนี้กับการเกิดสถานการณ์ “ยอดขายปกติ” และ “ยอดขายแย่”

	%ความต้องการ Demand[1] ฉบับ	%ความต้องการ Demand[2] ฉบับ	%ความต้องการ Demand[3] ฉบับ	%ความต้องการ Demand[4] ฉบับ	%ความต้องการ Demand[5] ฉบับ	%ความต้องการ Demand[6] ฉบับ	%ความต้องการ Demand[7] ฉบับ
Good day=1	3	8	23	43	78	93	100
Fair day=2	10	28	68	88	96	100	100
Poor day=3	44	66	82	94	100	100	100

ii. ข้อมูลในอาร์เรย์ P_Demand

ภาพผนวกที่ ข4 อาร์เรย์ P_Demand

5. กำหนดให้ k = 1 และ count =1

6. กำหนดให้การสั่งจำนวนหนังสือพิมพ์ล่วงหน้าคือ Number_Purchased = Demand[k]

7. คำนวณหากำไรเฉลี่ยของการสั่งจำนวนหนังสือพิมพ์ล่วงหน้า เริ่มต้นกำหนดให้ i=1

8. สุ่มตัวเลขในช่วง 0 – 100 ขึ้นมาเพื่อจำลองการเกิดสถานการณ์ ดังต่อไปนี้

8.1 ถ้าอยู่ในช่วง 0 ถึง Good_Day แสดงว่า สุ่มการเกิดสถานการณ์ “ยอดขายดี” ซึ่งกำหนดให้ TypeDay = 1

8.2 หรือถ้าอยู่ในช่วง Good_Day + 1 ถึง Fair_Day แสดงว่า สุ่มการเกิดสถานการณ์ “ยอดขายปกติ” ซึ่งกำหนดให้ TypeDay = 2

8.3 หรือถ้าอยู่ในช่วง Fair_Day + 1 ถึง 100 แสดงว่า สุ่มการเกิดสถานการณ์ “ยอดขายแย่” ซึ่งกำหนดให้ TypeDay = 3

9. สุ่มตัวเลขในช่วง 0 – 100 ขึ้นมาเพื่อจำลองจำนวนความต้องการของผู้ซื้อ ด้วยการตรวจสอบกับข้อมูลในอาร์เรย์ P_Demand[TypeDay][]

9.1 ถ้าอยู่ในช่วง 0 ถึง P_Demand[TypeDay][1] แสดงว่า จำนวนความต้องการหนังสือพิมพ์ของลูกค้ารายย่อย คือ Random_demand = Demand[1]

9.2 หรือถ้าอยู่ในช่วง P_Demand[TypeDay][j] +1 ถึง P_Demand[TypeDay][j+1] โดยที่ $1 \le j \le (Num - 1)$ แสดงว่า จำนวนความต้องการหนังสือพิมพ์ของลูกค้ารายย่อย Random_demand = Demand[j+1]

10. ตัวแปร Revenue คือค่ารายได้ทั้งหมดจากการขายหนังสือพิมพ์ คำนวณจากภาพผนวกที่ ข5

- ถ้า $Number\_Purchased \langle Random\_demand$ กำหนดให้

$$\text{Re}venue = Number\_Purchased * Selling\_P$$

- หรือถ้า $Number\_Purchased \rangle demand$ กำหนดให้

$$\text{Re}venue = Random\_demand * Selling\_P$$

ภาพผนวกที่ ข5 วิธีคำนวณรายได้จากการขายหนังสือพิมพ์

11. Lost_Profit คือ กำไรที่ควรจะได้รับจากการขายหนังสือพิมพ์ คำนวณดังภาพผนวกที่ ข6

- ถ้า $demand \rangle Number\_Purchased$ กำหนดให้

$$Lost\_\text{Pr}ofit = (demand - Number\_Purchased) * (Selling\_P - Purchase\_P)$$

- หรือถ้า $demand \langle Number\_Purchased$ กำหนดให้

$$Lost\_\text{Pr}ofit = 0$$

ภาพผนวกที่ ข6 วิธีคำนวณกำไรที่ควรจะได้รับจากการขายหนังสือพิมพ์

12. $Scrap\_Sale$ คือ รายได้จากการขายหนังสือพิมพ์คืนโรงพิมพ์ คำนวณดังภาพผนวกที่ ข7

- ถ้า $(Number\_Purchased - Random\_demand) \rangle 0$ กำหนดให้

$$Scrap\_Sale = (Number\_Purchased - Random\_demand) * (Scrap\_P)$$

- หรือถ้า $demand \langle Number\_Purchased$ กำหนดให้

$$Scrap\_Sale = 0$$

ภาพผนวกที่ ข7 วิธีคำนวณรายได้จากการขายหนังสือพิมพ์คืนให้โรงพิมพ์

13. Daily_Cost คือ ราคาซื้อหนังสือพิมพ์จากโรงพิมพ์ (เงินลงทุน) คำนวณดังภาพผนวกที่ ข8

$$Daily\_Cost = Number\_Purchased \times Purchase\_P$$

ภาพผนวกที่ ข8 วิธีคำนวณเงินลงทุน

14. Daily_Profit คือ ผลกำไรที่ได้รับ คำนวณดังภาพผนวกที่ ข9

$$Daily\_Profit = Revenue - Lost\_Profit + Scrap\_Sale - Daily\_Cost$$

ภาพผนวกที่ ข9 วิธีคำนวณผลกำไร

15. Total_Profit คือ ผลรวมของกำไรจากการการขายหนังสือพิมพ์ทั้งหมด Day วัน คำนวณดังภาพผนวกที่ ข10

$$Total\_Profit = Total\_Profit + Daily\_Profit$$

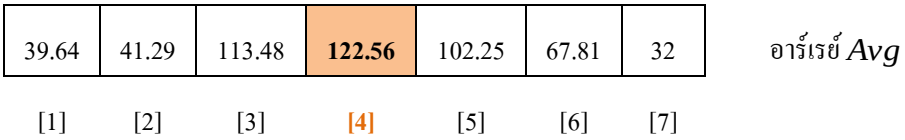
ภาพผนวกที่ ข10 วิธีคำนวณผลกำไรรวมจากการจำลองสถานการณ์

16. ทำการคำนวณผลกำไรจากการจำลองวันที่ i + 1 ถัดไป โดยย้อนกลับที่ทำข้อ8 จนกระทั่ง i = Day

17. ทำการคำนวณผลกำไรจากการจำลองรอบที่ count + 1 โดยย้อนกลับที่ทำข้อ 7 จนกระทั่ง Count = Iteration

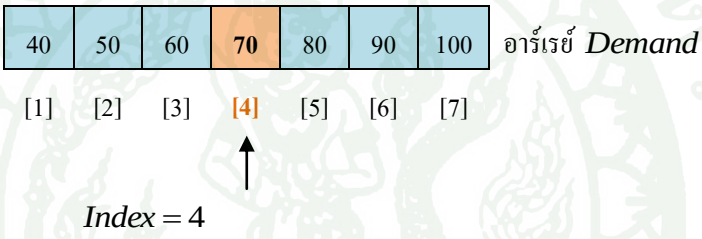
18. คำนวณหาผลกำไรเฉลี่ยของการสั่งหนังสือพิมพ์ล่วงหน้า Demand[k] ฉบับ จาก Avg[k] = Total_Profit / Iteration จากนั้นย้อยกลับที่ข้อ 6 เพื่อหากำไรเฉลี่ยของการสั่งหนังสือพิมพ์ล่วงหน้า Demand[k+1] ฉบับ จนกระทั่ง k = Num

19. ค้นหาตำแหน่งในอาร์เรย์ Avg ที่มีค่าสูงสุด และกำหนดค่าให้ตัวแปร Index แสดงดังภาพผนวกที่ ข11



ภาพผนวกที่ ข11 ค้นหาตำแหน่งที่ทำให้ค่าเฉลี่ยกำไรสูงสุด จากการจำลองสถานการณ์

20. เลือกคำตอบจากอาร์เรย์ Demand [Index] เป็นคำตอบที่ให้ค่าเฉลี่ยผลกำไรสูงสุด แสดงดังผนวกที่ ข12



ภาพผนวกที่ ข12 วิธีการเลือกคำตอบที่ดีที่สุด จากการจำลองสถานการณ์

ภาคผนวก ค

อัลกอริทึมการแก้ปัญหา PDPTW

อัลกอริทึมการแก้ปัญหา PDPTW

ขั้นตอนการหาผลลัพธ์ให้กับแบบจำลองสถานการณ์ปัญหา PDPTW ประกอบด้วย 2 ขั้นตอน คือ 1) คำนวณหาคำตอบเริ่มต้น ด้วยกระบวนการ Insertion Heuristic และ 2) ทำการค้นหาคำตอบใกล้เคียง (Neighborhood Search) จากบริเวณคำตอบที่ได้จากขั้นตอนที่ 1 ด้วยกระบวนการค้นหาแบบทาบู และปรับปรุงคำตอบปัจจุบันเมื่อได้คำตอบข้างเคียงที่ดีกว่า

กำหนดพารามิเตอร์ ก่อนการคำนวณการแก้ปัญหา PDPTW ดังนี้

ตารางผนวกที่ ค1 การกำหนดพารามิเตอร์ สำหรับปัญหา PDPTW

ลำดับที่	ชื่อตัวแปร	ความหมาย
1.	$N = \{0,1,...,n\}$	เซตของจุดปล่อยรถ และหมายเลขลำดับร้านค้าทั้งหมดที่มีการร้องขอการบริการ
2.	n	จำนวนร้านค้าทั้งหมดที่มีการร้องขอการบริการ (ไม่รวมร้านค้าลำดับที่ 0)
3.	$q[i], i \in N$	น้ำหนักสินค้าที่ทำการโหลดขึ้น/ลงรถบรรทุก ณ ร้านค้าลำดับที่ i โดยมีเงื่อนไข -ถ้า $q[i] \rangle 0$ หมายถึง ร้านค้าลำดับที่ i เป็น “จุดบรรทุกสินค้า” ไปส่งร้านค้าลำดับอื่นๆ -ถ้า $q[i] \langle 0$ หมายถึง ร้านค้าลำดับที่ i เป็น “จุดส่งสินค้า” จากร้านค้าลำดับอื่นๆ -ถ้า $q[i] = 0$ หมายถึง ร้านค้าลำดับที่ i คือ “จุดปล่อยรถบรรทุกสินค้า”
4.	$t[i][j], i\ j \in N$	ระยะเวลาการเดินทางระหว่างร้านค้าลำดับที่ i และ j โดยใช้การคำนวณตามทฤษฎีพีทาโกรัสจากตำแหน่งที่ตั้งในแนวแกน X และY ของร้านค้าลำดับที่ i และ j เมื่อทราบระยะห่างระหว่างร้านค้าลำดับที่ i และ j นำไปคูณกับอัตราเร็วการเคลื่อนที่ของรถบรรทุก จึงได้ระยะเวลาการเดินทางระหว่างร้านค้าลำดับที่ i และ j

ตารางผนวกที่ ค1 (ต่อ)

ลำดับที่	ชื่อตัวแปร	ความหมาย
5.	$e[i], i \in N$	ช่วงเวลาการเปิดร้านค้า ลำดับที่ i
6.	$l[i], i \in N$	ช่วงเวลาการปิดร้านค้า ลำดับที่ i
7.	$p[i], i \in N$	จุดรับสินค้า ของร้านค้าลำดับที่ i โดยเงื่อนไข -ถ้า $p[i] \neq 0$ หมายถึง ร้านค้าลำดับที่ i รอรับสินค้าจากร้านค้าลำดับที่ $p[i]$ -ถ้า $p[i] = 0$ หมายถึง ร้านค้าลำดับที่ i ไม่ได้รอรับสินค้าจากร้านค้าใดๆ
8.	$d[i], i \in N$	จุดส่งสินค้า ของร้านค้าลำดับที่ i โดยเงื่อนไข -ถ้า $d[i] \neq 0$ หมายถึง ร้านค้าลำดับที่ i บรรทุกสินค้าส่งไปยังร้านค้าลำดับที่ $d[i]$ -ถ้า $d[i] = 0$ หมายถึง ร้านค้าลำดับที่ i ไม่ได้บรรทุกสินค้าส่งไปยังร้านค้าใดๆ
9.	$s[i], i \in N$	ระยะเวลาการให้บริการ ณ ร้านค้าลำดับที่ i
10.	$M = \{1,2,...,m\}$	เซตของหมายเลขลำดับของรถบรรทุกสินค้าที่มีอยู่ในระบบทั้งหมด m คัน
11.	C	จำนวนความจุของรถบรรทุกสินค้า
12.	$TabuSize$	ขนาดของข้อมูลที่ถูกกำหนดสถานะ “ต้องห้าม” (Tabu List Size)
13.	A	อาร์เรย์จัดเก็บหมายเลขร้านค้าที่เป็น“จุดบรรทุกสินค้า” โดยที่ $q[i] \rangle 0$, $i \in N$ แสดงดังภาพผนวกที่ ค1

i q[i]

Customer	X Coordination	Y Coordination	Demand/Load	Window Start Time	Window End Time	Service Time	Pickup Node	Delivery Node
0	40	50	0	0	1236	0	0	0
1	45	68	**10**	0	1127	90	0	75
2	45	70	-20	0	1125	90	8	0
3	42	66	**10**	0	1129	90	0	10
4	42	68	-20	727	782	90	6	0
5	42	65	**10**	0	1130	90	0	9
6	40	69	**20**	621	702	90	0	4
7	40	66	**20**	0	1130	90	0	11
8	38	68	**20**	255	324	90	0	2
9	38	70	-10	534	605	90	5	0
10	35	66	-10	357	410	90	3	0
11	35	69	-20	448	505	90	7	0
12	25	85	-30	0	1107	90	13	0
13	22	75	**30**	30	92	90	0	12
14	22	85	-40	567	620	90	16	0
15	20	80	-20	384	429	90	18	0
16	20	85	**40**	475	528	90	0	14
17	18	75	**20**	99	148	90	0	19
18	15	75	**20**	179	254	90	0	15
19	15	80	-20	278	345	90	17	0
20	30	50	**10**	10	73	90	0	22
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
106	26	32	-10	0	1123	90	50	0

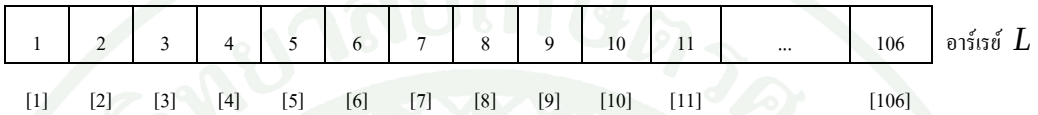
1	3	5	6	7	8	13	16	17	18	23	...	98
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		[53]

อาร์เรย์ A

ภาพผนวกที่ ค1 วิธีการเลือกข้อมูลบางส่วนใส่อาร์เรย์ q

ขั้นตอนการค้นหาคำตอบเริ่มต้น ด้วยแนวคิด Insertion Heuristic สำหรับแบบจำลองสถานการณ์การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด มีดังต่อไปนี้

1. ให้ L คือ อาร์เรย์จัดเก็บหมายเลขลำดับร้านค้าทั้งหมดที่มีการร้องขอการบริการ ซึ่งมีขนาด n แสดงดังภาพผนวกที่ ค2



ภาพผนวกที่ ค2 การจัดเก็บข้อมูลในอาร์เรย์ L

2. ทำการเรียงลำดับช่วงเวลาเปิดทำการ ($e[i]$) ของสมาชิกในอาร์เรย์ A จากน้อยไปมาก แสดงดังภาพผนวกที่ ค3

$e[i]$

Customer	X Coordination	Y Coordination	Demand/Load	**Window Start Time**	Window End Time	Service Time	Pickup Node	Delivery Node
1	45	68	10	**0**	1127	90	0	75
3	42	66	10	**0**	1129	90	0	10
5	42	65	10	**0**	1130	90	0	9
6	40	69	20	**621**	702	90	0	4
7	40	66	20	**0**	1130	90	0	11
8	38	68	20	**255**	324	90	0	2
13	22	75	30	**30**	92	90	0	12
16	20	85	40	**475**	528	90	0	14
17	18	75	20	**99**	148	90	0	19
18	15	75	20	**179**	254	90	0	15
20	30	50	10	**10**	73	90	0	22
...	...	...	...	**...**	...	...	...	...
98	58	75	20	**0**	1115	90	0	94

1	3	5	7	31	37	50	52	82	83	95	...	68
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		[53]

อาร์เรย์ A

ภาพผนวกที่ ค3 การจัดเรียงข้อมูลในอาร์เรย์ A

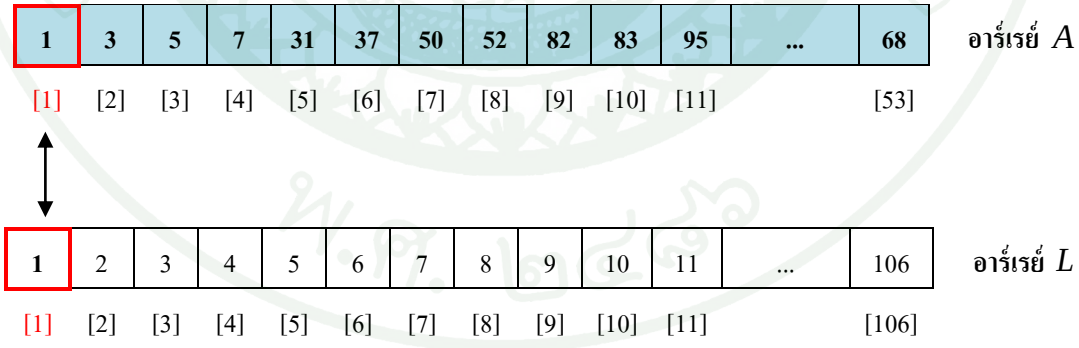
3. กำหนดให้ R คือ อาร์เรย์ 2 มิติเก็บเส้นทางการเดินทางของรถบรรทุก ซึ่งมีขนาด $m \times n$ โดยให้แถวของอาร์เรย์แทน “หมายเลขรถ” และคอลัมน์ของอาร์เรย์แทน “จุดจอดรับ/ส่งสินค้า” มีสถานะ “ว่าง” แสดงดังภาพผนวกที่ ค4

	จุดจอดที่ 1	จุดจอดที่ 2	จุดจอดที่ 3	จุดจอดที่ 4	จุดจอดที่ 5	จุดจอดที่ 6	จุดจอดที่ 7	จุดจอดที่ 8	จุดจอดที่ 9	จุดจอดที่ 10	จุดจอดที่ 11	จุดจอดที่ ...	จุดจอดที่ n
รถคันที่ 1	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ 2	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ 3	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ ...	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
รถคันที่m-1	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ m	0	0	0	0	0	0	0	0	0	0	0	...	0

ภาพผนวกที่ ค4 กำหนดสถานะ “ว่าง” ให้กับเส้นทางการเดินทางของรถทุกคัน

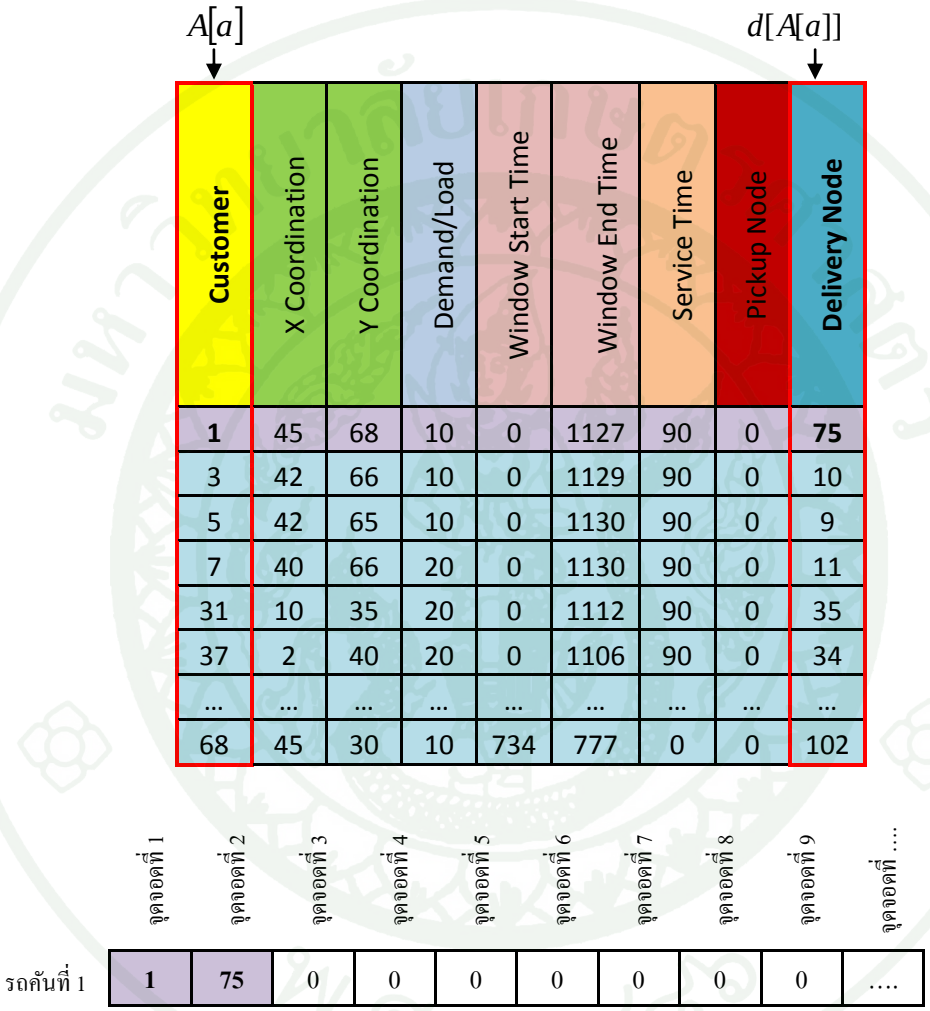
4. ให้ k แทนหมายเลขรถบรรทุกที่ให้บริการ เริ่มต้นกำหนดให้ $k=1$

5. เริ่มต้นการคำนวณหาเส้นทางของรถบรรทุกคันที่ k ($R[k][]$) โดยที่ $1 \le k \le m$ เลือกข้อมูลในอาร์เรย์ A ตำแหน่ง a ใดๆ ($A[a]$) โดยที่ $1 \le a \le \frac{n}{2}$ ซึ่งข้อมูล $A[a]$ต้องเป็นสมาชิกในอาร์เรย์ L แสดงดังภาพผนวกที่ ค5

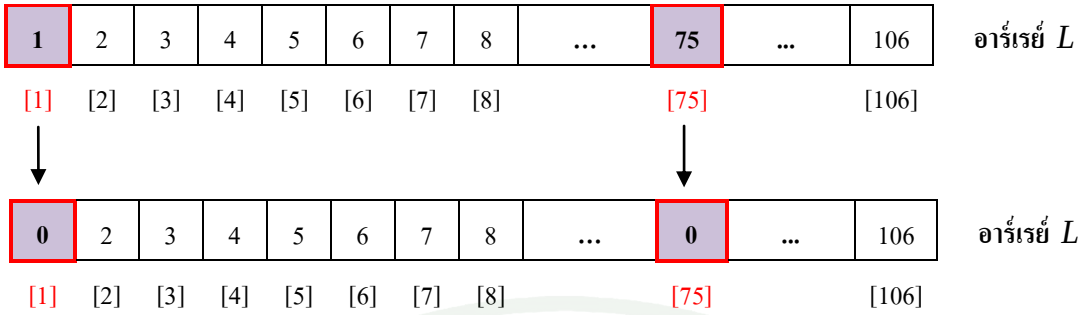


ภาพผนวกที่ ค5 ข้อมูลในตำแหน่ง $A[a]$ เป็นสมาชิกในอาร์เรย์ L

6. จัดเก็บ “จุดบรรทุกสินค้า ($A[a]$)” และ“จุดส่งสินค้า ($d[A[a]]$)” ที่ถูกคัดเลือกในข้อ 5. ลงในเส้นทางของรถบรรทุกคันที่ k ($R[k][]$) แสดงดังภาพผนวกที่ ค6 และทำการลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L เพื่อเป็นการทำสัญลักษณ์ว่า “ร้านค้าลำดับที่ $A[a]$ และ $d[A[a]]$ ได้ถูกรับบริการแล้ว” แสดงดังภาพผนวกที่ ค7

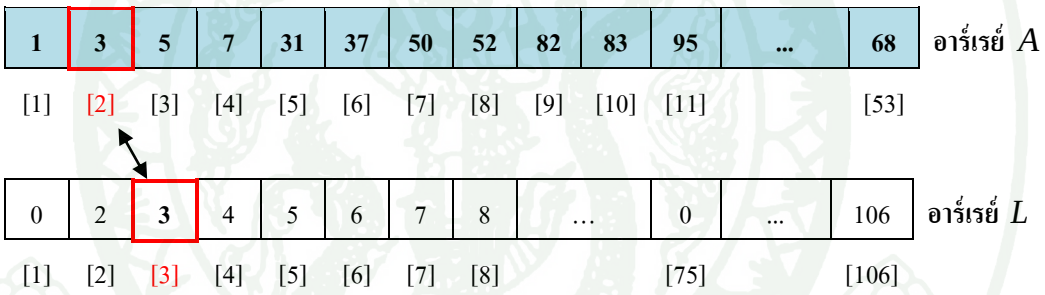


ภาพผนวกที่ ค6 การเพิ่มข้อมูล $A[a]$ และ $d[A[a]]$ ในอาร์เรย์ $R[k][]$ กรณี $k=1$



ภาพผนวกที่ ค7 การลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L

7. พิจารณาแทรก “จุดบรรทุกสินค้า” อื่นๆลงในเส้นทางการเดินทางของรถบรรทุกคันที่ k โดยเลือกข้อมูลในอาร์เรย์ A ตำแหน่ง a ใดๆ($A[a]$) โดยที่ $1 \le a \le \frac{n}{2}$ ซึ่งข้อมูล $A[a]$ต้องเป็นสมาชิกในอาร์เรย์ L แสดงดังภาพผนวกที่ ค8



ภาพผนวกที่ ค8 ข้อมูลในตำแหน่ง $A[a]$ เป็นสมาชิกในอาร์เรย์ L

ทำการแทรก “จุดบรรทุกสินค้า ($A[a]$)” และ“จุดส่งสินค้า ($d[A[a]]$)” ที่ได้พิจารณาลงในเส้นทางการเดินทางของรถบรรทุกคันที่ k ($R[k][]$) แทรกทุกตำแหน่งที่เป็นไปได้บนเส้นทางเดินทางของรถบรรทุกคันเดิมเพื่อเลือกตำแหน่งที่แทรกแล้วได้ระยะเวลาที่ใช้เดินทางของรถสั้นที่สุด ซึ่งขั้นตอนการคำนวณระยะเวลานั้นอยู่ในเงื่อนไขของกรอบเวลาเปิด/ปิดทำการของร้านค้านั้นๆ, ผลรวมของระยะเวลาการเดินทางระหว่างร้านค้าที่อยู่ในเส้นทางการเดินรถอยู่ในเงื่อนไขของกรอบเวลาการทำงานของรถบรรทุก และขีดจำกัดการบรรทุกของรถตลอดเส้นทางการเดินทาง แสดงดังภาพผนวกที่ ค9 – ค10

รถคันที่ 1	1	75	0	0	0	0	0	0	….

i. เส้นทางการเดินรถเดิมของรถคันที่ k (กรณี k=1)

$A[a]$ $d[A[a]]$

Customer	X Coordination	Y Coordination	Demand/Load	Window Start Time	Window End Time	Service Time	Pickup Node	Delivery Node
1	45	68	10	0	1127	90	0	75
3	42	66	10	0	1129	90	0	**10**
5	42	65	10	0	1130	90	0	9
7	40	66	20	0	1130	90	0	11
31	10	35	20	0	1112	90	0	35
37	2	40	20	0	1106	90	0	34
…	…	…	…	…	…	…	…	…
68	45	30	10	734	777	0	0	102

ii. พิจารณา “จุดบรรทุกสินค้า ($A[a]$)” และ “จุดส่งสินค้า ($d[A[a]]$)”

ภาพผนวกที่ ค๑ วิธีการแทรก $A[a]$ และ $d[A[a]]$ ทุกตำแหน่งที่เป็นไปได้ภายในอาร์เรย์ $R[k][]$ เพื่อคำนวณหาระยะเวลาเดินทางรวมทั้งเส้นทาง และเลือกตำแหน่งที่แทรกแล้วได้ระยะทางที่สั้นที่สุดเป็นเส้นทางการเดินทางของรถบรรทุกคันที่ k

	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3	คอลัมน์ที่ 4	คอลัมน์ที่ 5	คอลัมน์ที่ 6	คอลัมน์ที่ 7	คอลัมน์ที่ 8	คอลัมน์ที่ ...	เวลาเดินทางรวม
แทรกครั้งที่1	3	10	1	75	0	0	0	0	…	52
แทรกครั้งที่2	3	1	10	75	0	0	0	0	…	*
แทรกครั้งที่3	3	1	75	10	0	0	0	0	…	49
แทรกครั้งที่4	1	3	10	75	0	0	0	0	…	*
แทรกครั้งที่5	1	3	75	10	0	0	0	0	…	*
แทรกครั้งที่6	1	75	3	10	0	0	0	0	…	48

*ไม่สามารถคำนวณระยะเวลาเดินทางรวมได้เนื่องจากไม่อยู่ในเงื่อนไขบังคับ

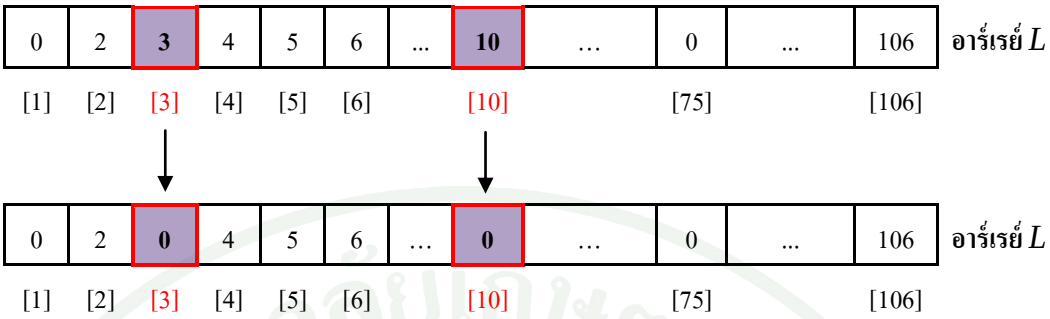
iii. วิธีการแทรก $A[a]$ และ $d[A[a]]$ ทุกตำแหน่งที่เป็นไปได้ในเส้นทางการเดินรถของรถคันที่ k

	จุดจอดที่ 1	จุดจอดที่ 2	จุดจอดที่ 3	จุดจอดที่ 4	จุดจอดที่ 5	จุดจอดที่ 6	จุดจอดที่ 7	จุดจอดที่ 8	จุดจอดที่ 9	จุดจอดที่ ...	จุดจอดที่ n
รถคันที่ 1	1	75	**3**	**10**	0	0	0	0	0	…	0

iv. เลือกตำแหน่งที่แทรกแล้วได้ระยะเวลาเดินทางรวมที่สั้นที่สุด กำหนดให้เป็นเส้นทางเดินรถใหม่ของรถคันที่ k

ภาพผนวกที่ ค10 วิธีการแทรก $A[a]$ และ $d[A[a]]$ ทุกตำแหน่งที่เป็นไปได้ภายในอาร์เรย์ $R[k][]$ เพื่อคำนวณหาระยะเวลาเดินทางรวมทั้งเส้นทาง และเลือกตำแหน่งที่แทรกแล้วได้ระยะทางที่สั้นที่สุดเป็นเส้นทางการเดินทางของรถบรรทุกคันที่ k (ต่อ)

เมื่อเลือกตำแหน่งการวางเส้นทางเดินรถที่ให้ระยะเวลาสั้นที่สุดแล้ว ทำการลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L เพื่อเป็นการทำสัญลักษณ์ว่า “ร้านค้าลำดับที่ $A[a]$ และ $d[A[a]]$ ได้ถูกรับบริการแล้ว” แสดงดังภาพผนวกที่ ค11



ภาพผนวกที่ ค11 การลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L

8. ทำการพิจารณา “จุดบรรทุกสินค้า” อื่นๆที่มีการร้องขอการบริการด้วยวิธีการดังข้อ 7. จนกระทั่งไม่สามารถคำนวณหาระยะทางรวมทั้งเส้นทางได้ จึงหยุดการเพิ่มเส้นทางการเดินทางของรถบรรทุกคันที่ k

9. ทำการตรวจสอบข้อมูลสมาชิกทั้งหมดในอาร์เรย์ L

9.1 ถ้ามีสมาชิกบางตัว $\neq 0$ ให้ย้อนกลับไปทำข้อ 5. เพื่อคำนวณหาเส้นทางการเดินทางของรถบรรทุกคันที่ $k+1$

9.2 ถ้าสมาชิกทุกตัว $= 0$ แสดงว่าสิ้นสุดการหาคำตอบเริ่มต้น

จากขั้นตอนการคำนวณหาคำตอบเริ่มต้น ด้วยแนวคิด Insertion Heuristic ทั้ง 9 ข้อ คำตอบตอบที่ได้ คือ จำนวนรถทั้งหมดที่ให้การบริการทุกการร้องขอจำนวน k คัน และเส้นทางการเดินทางของรถแต่ละคัน จากข้อมูลในอาร์เรย์ R แสดงดังภาพผนวกที่ ค12

	จุดจอดที่ 1	จุดจอดที่ 2	จุดจอดที่ 3	จุดจอดที่ 4	จุดจอดที่ 5	จุดจอดที่ 6	จุดจอดที่ 7	จุดจอดที่ 8	จุดจอดที่ 9	จุดจอดที่ 10	จุดจอดที่ 11	จุดจอดที่ ...	จุดจอดที่ n
รถคันที่ 1	5	1	75	3	10	9	37	34	52	50	106	...	0
รถคันที่ 2	7	31	35	11	95	100	98	94	82	85	0	...	0
รถคันที่ 3	20	83	96	92	84	97	101	23	22	21	0	...	0
รถคันที่ ...	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
รถคันที่12	63	93	99	69	0	0	0	0	0	0	0	...	0
รถคันที่ 13	38	36	0	0	0	0	0	0	0	0	0	...	0

ภาพผนวกที่ ค12 เส้นทางการให้บริการร้านค้าต่างๆ ของรถแต่ละคัน จากตัวอย่างมีการใช้จำนวนรถบรรทุกสินค้าทั้งสิ้น 13 คัน คอยบริการทุกการร้องขอ

เมื่อได้คำตอบเริ่มต้นแล้วขั้นตอนถัดไปทำการค้นหาคำตอบใกล้เคียงบริเวณที่พบคำตอบเริ่มต้นด้วยแนวคิดการค้นหาคำตอบแบบทาบู จากนั้นทำการปรับปรุงค่าคำตอบปัจจุบันเมื่อได้ค่าคำตอบใกล้เคียงที่ดีกว่าได้ดังนี้

1. ทำการเรียงลำดับช่วงเวลาเปิดทำการ($e[i]$) ของสมาชิกในอาร์เรย์ A จากน้อยไปมาก แสดงดังภาพผนวกที่ ค13

$e[i]$

Customer	X Coordination	Y Coordination	Demand/Load	**Window Start Time**	Window End Time	Service Time	Pickup Node	Delivery Node
1	45	68	10	**0**	1127	90	0	75
3	42	66	10	**0**	1129	90	0	10
5	42	65	10	**0**	1130	90	0	9
6	40	69	20	**621**	702	90	0	4
7	40	66	20	**0**	1130	90	0	11
8	38	68	20	**255**	324	90	0	2
13	22	75	30	**30**	92	90	0	12
16	20	85	40	**475**	528	90	0	14
17	18	75	20	**99**	148	90	0	19
18	15	75	20	**179**	254	90	0	15
20	30	50	10	**10**	73	90	0	22
...	...	...	...	**...**	...	...	...	...
98	58	75	20	**0**	1115	90	0	94

1	**3**	**5**	**7**	**31**	**37**	**50**	**52**	**82**	**83**	**95**	**...**	**68**
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]		[53]

อาร์เรย์ A

ภาพผนวกที่ ค13 การจัดเรียงข้อมูลในอาร์เรย์ A

2. กำหนดให้ Y คือ อาร์เรย์ 2 มิติ ขนาด $k \times n$ เก็บเส้นทาง “จุดบรรทุกสินค้า” ของรถที่ให้บริการทั้งหมดจากคำตอบเริ่มต้น ตรวจสอบจาก $d[R[i][j]] \neq 0$ โดย $i \in k$ และ $j \in n$ กำหนดให้แถวของอาร์เรย์แทน “หมายเลขรถ” และคอลัมน์ของอาร์เรย์แทน “จุดจอดบรรทุกสินค้า” แสดงดังภาพผนวกที่ ค14

	จุดจอดบรรทุกสินค้าที่ 1	จุดจอดบรรทุกสินค้าที่ 2	จุดจอดบรรทุกสินค้าที่ 3	จุดจอดบรรทุกสินค้าที่ 4	จุดจอดบรรทุกสินค้าที่ 5	จุดจอดบรรทุกสินค้าที่ 6	จุดจอดบรรทุกสินค้าที่ 7	จุดจอดบรรทุกสินค้าที่ 8	จุดจอดบรรทุกสินค้าที่ 9	จุดจอดบรรทุกสินค้าที่ 10	จุดจอดบรรทุกสินค้าที่ 11	จุดจอดบรรทุกสินค้าที่...	จุดจอดบรรทุกสินค้าที่ n
รถคันที่ 1	5	1	3	37	52	50	0	0	0	0	0	...	0
รถคันที่ 2	7	31	95	98	82	0	0	0	0	0	0	...	0
รถคันที่ 3	20	83	96	97	23	0	0	0	0	0	0	...	0
รถคันที่ ...	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
รถคันที่12	63	93	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ 13	38	0	0	0	0	0	0	0	0	0	0	...	0

ภาพผนวกที่ ค14 “จุดจอดบรรทุกสินค้า”ของรถบรรทุกที่ให้บริการทุกคันจากคำตอบเริ่มต้น

กำหนดให้ตัวแปร *MinCar* มีค่าเท่ากับ *k* เพื่อใช้ในการเปรียบเทียบจำนวนรถที่ให้บริการของคำตอบเริ่มต้นกับคำตอบตอบใหม่ที่ถูกปรับปรุงโดยใช้วิธีการค้นหาแบบทาบู

3. ให้ตัวแปร *TabuSize* มีค่าเท่ากับขนาดของลิสต์ทาบู (Tabu List Size) และกำหนดให้คำตอบปัจจุบัน (คำตอบเริ่มต้น) เป็นคำตอบที่ดีที่สุด

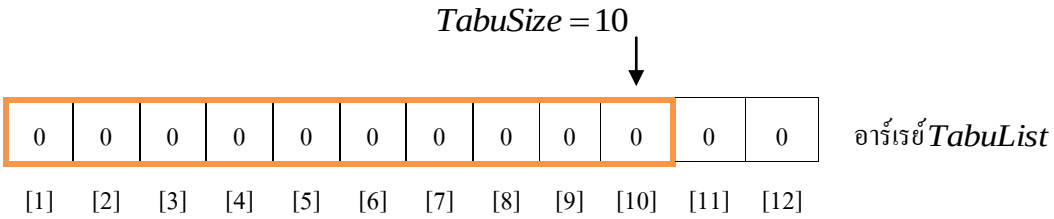
3.1 ให้ *TabuStatus* คือ อาร์เรย์ขนาด *n* ที่ระบุ “หมายเลขรถคันที่ให้บริการจุดจอดบรรทุกสินค้าที่มีสถานะ “ต้องห้าม” ” แสดงดังภาพผนวกที่ ค15

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]		[106]
0	0	0	0	0	0	0	0	...	0

อาร์เรย์ *TabuStatus*

ภาพผนวกที่ ค15 ข้อมูลในอาร์เรย์ *TabuStatus*

3.2 ให้ *TabuList* คือ อาร์เรย์ ขนาด *TabuSize* จัดเก็บข้อมูล “จุดจอดบรรทุกสินค้า” ที่อยู่ในสถานะ “ต้องห้าม” และให้ *IndexTabu* เก็บตำแหน่งข้อมูลที่ถูกจัดเก็บภายในอาร์เรย์ *TabuList* แสดงดังภาพผนวกที่ ค16



ภาพผนวกที่ ค16 ข้อมูลในอาร์เรย์ *TabuList*

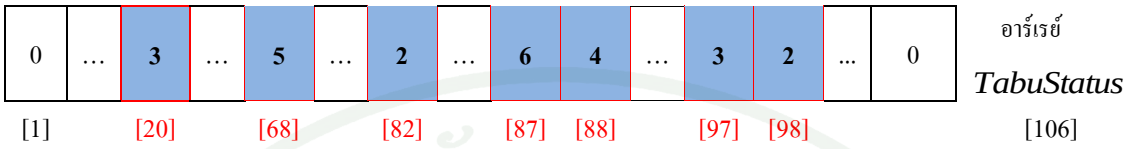
4. เตรียมข้อมูลก่อนการค้นหาคำตอบข้างเคียง

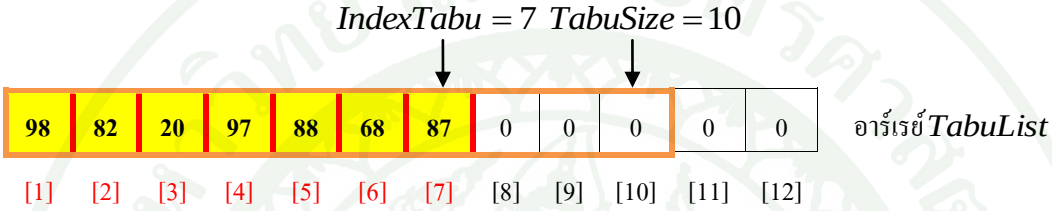
4.1 สุ่มเลือก "จุดจอดบรรทุกสินค้า" ของรถบรรทุก ในอาร์เรย์ Y เพื่อกำหนดสถานะ "ต้องห้าม" แสดงดังภาพผนวกที่ ค17

	จุดจอดบรรทุกสินค้าที่ 1	จุดจอดบรรทุกสินค้าที่ 2	จุดจอดบรรทุกสินค้าที่ 3	จุดจอดบรรทุกสินค้าที่ 4	จุดจอดบรรทุกสินค้าที่ 5	จุดจอดบรรทุกสินค้าที่ 6	จุดจอดบรรทุกสินค้าที่ 7	จุดจอดบรรทุกสินค้าที่ 8	จุดจอดบรรทุกสินค้าที่...	จุดจอดบรรทุกสินค้าที่ n
รถคันที่ 1	5	1	3	37	52	50	0	0	…	0
รถคันที่ 2	7	31	95	98	82	0	0	0	…	0
รถคันที่ 3	20	83	96	97	23	0	0	0	…	0
รถคันที่ 4	67	24	27	88	0	0	0	0		0
รถคันที่ 5	43	42	74	28	68	0	0	0	…	0
รถคันที่ 6	90	87	71	58	0	0	0	0	…	0
รถคันที่ ...	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
รถคันที่ 13	38	0	0	0	0	0	0	0	…	0

ภาพผนวกที่ ค17 สุ่มเลือก "จุดจอดบรรทุกสินค้า" ของรถบรรทุก ในอาร์เรย์ Y

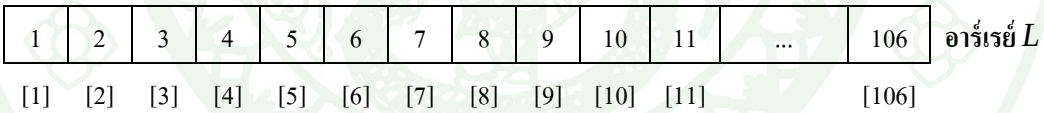
4.2 กำหนดสถานะ “ต้องห้าม” ให้แก่ “จุดจอดบรรทุกสินค้า” ที่ถูกสุ่มขึ้นโดยการจัดเก็บ“จุดจอดบรรทุกสินค้า”นั้นลงในอาร์เรย์ *TabuList* และจัดเก็บข้อมูลหมายเลขรถที่ให้บริการลงในอาร์เรย์ *TabuStatus* เพื่อดำเนินการค้นหาคำตอบข้างเคียง แสดงดังภาพผนวกที่ ค18





ภาพผนวกที่ ค18 การจัดเก็บข้อมูลสุ่มจากอาร์เรย์ *Y* ลง *TabuStatus* และ *TabuList*

5. ให้ *L* คือ อาร์เรย์จัดเก็บหมายเลขลำดับร้านค้าทั้งหมดที่มีการร้องขอการบริการ ซึ่งมีขนาด *n* แสดงดังภาพผนวกที่ ค19



ภาพผนวกที่ ค19 การจัดเก็บข้อมูลในอาร์เรย์ *L*

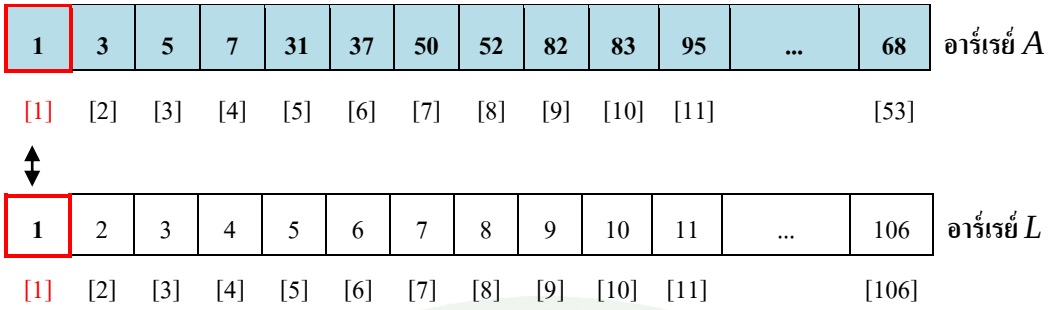
6. กำหนดให้ *RN* (Route of Neighborhood) คือ อาร์เรย์ 2 มิติ เก็บเส้นทางการเดินรถของคำตอบข้างเคียง ซึ่งมีขนาด $m \times n$ โดยให้แถวของอาร์เรย์แทน “หมายเลขรถ” และคอลัมน์ของอาร์เรย์แทน “จุดจอดรับ/ส่งสินค้า” มีสถานะ “ว่าง” แสดงดังภาพผนวกที่ ค20

	จุดจอดที่ 1	จุดจอดที่ 2	จุดจอดที่ 3	จุดจอดที่ 4	จุดจอดที่ 5	จุดจอดที่ 6	จุดจอดที่ 7	จุดจอดที่ 8	จุดจอดที่ 9	จุดจอดที่ 10	จุดจอดที่ 11	จุดจอดที่ ...	จุดจอดที่ n
รถคันที่ 1	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ 2	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ 3	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ ...	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
รถคันที่m-1	0	0	0	0	0	0	0	0	0	0	0	...	0
รถคันที่ m	0	0	0	0	0	0	0	0	0	0	0	...	0

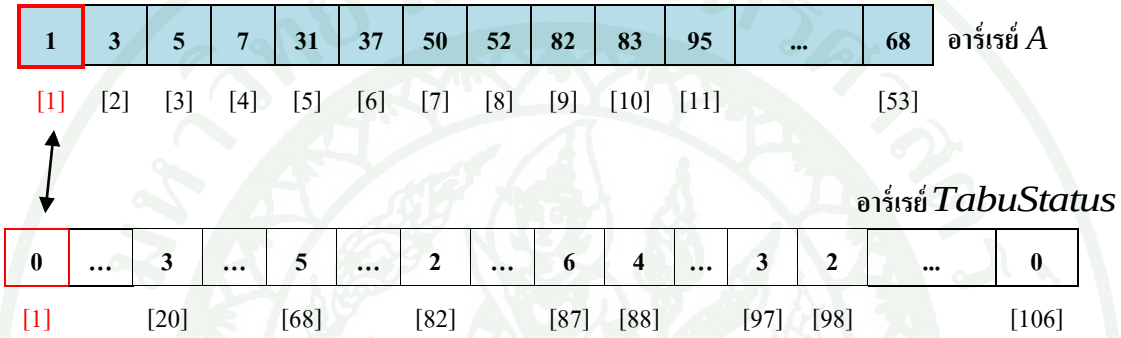
ภาพผนวกที่ ค20 กำหนดสถานะ “ว่าง” ให้กับเส้นทางการเดินทางรถของคำตอบข้างเคียง

7. ให้ k แทนหมายเลขรถบรรทุกที่ให้บริการ เริ่มต้นกำหนดให้ $k=1$

8. เริ่มต้นการคำนวณหาเส้นทางของรถบรรทุกคันที่ k ($RN[k][]$) โดยที่ $1 \le k \le m$ เลือกข้อมูลในอาร์เรย์ A ตำแหน่ง a ใดๆ ($A[a]$) โดยที่ $1 \le a \le \frac{n}{2}$ ซึ่งข้อมูล $A[a]$ ต้องเป็นสมาชิกในอาร์เรย์ L และไม่ถูกกำหนดสถานะ “ต้องห้าม” จากหมายเลขรถคันที่ k มาก่อน ซึ่งตรวจสอบกับข้อมูลอาร์เรย์ *TabuStatus* แสดงดังภาพผนวกที่ ค21



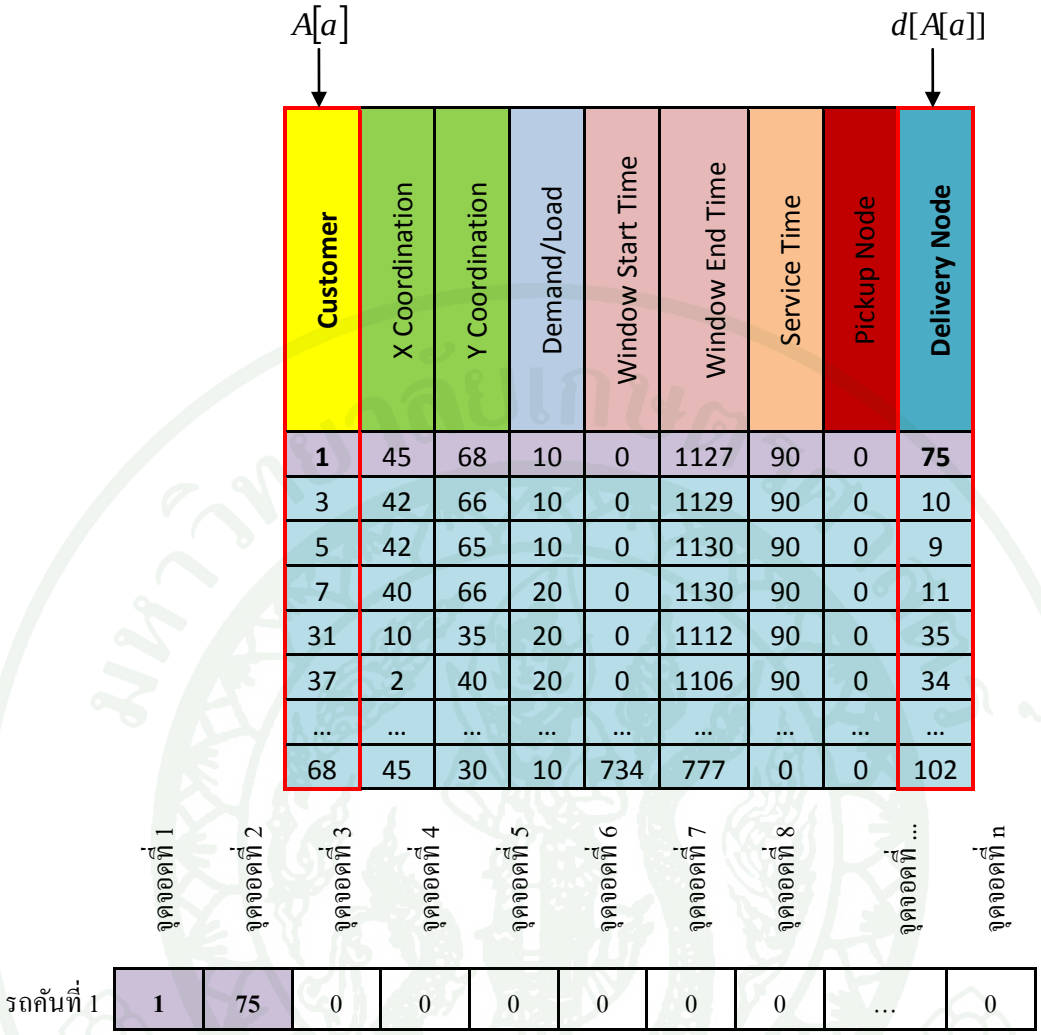
i. ข้อมูลในตำแหน่ง $A[a]$ เป็นสมาชิกในอาร์เรย์ L



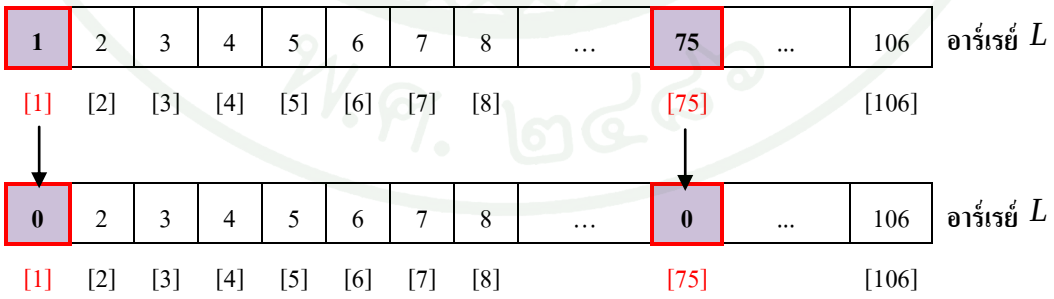
ii. ข้อมูลในตำแหน่ง $A[a]$ ไม่ถูกกำหนดสถานะ “ต้องห้าม” จากรถคันที่ k มาก่อน (กรณี k=1)

ภาพผนวกที่ ค21 วิธีการเลือก $A[a]$ กรณีการค้นหาคำตอบข้างเคียง

9. จัดเก็บ “จุดบรรทุกสินค้า ($A[a]$)” และ “จุดส่งสินค้า ($d[A[a]]$)” ที่ถูกคัดเลือกในข้อ 8 ลงในเส้นทางของรถบรรทุกคันที่ k ($RN[k][]$) แสดงดังภาพผนวกที่ ค22 และทำการลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L เพื่อเป็นการทำสัญลักษณ์ว่า “ร้านค้าลำดับที่ $A[a]$ และ $d[A[a]]$ ได้ถูกรับบริการแล้ว” แสดงดังภาพผนวกที่ ค23

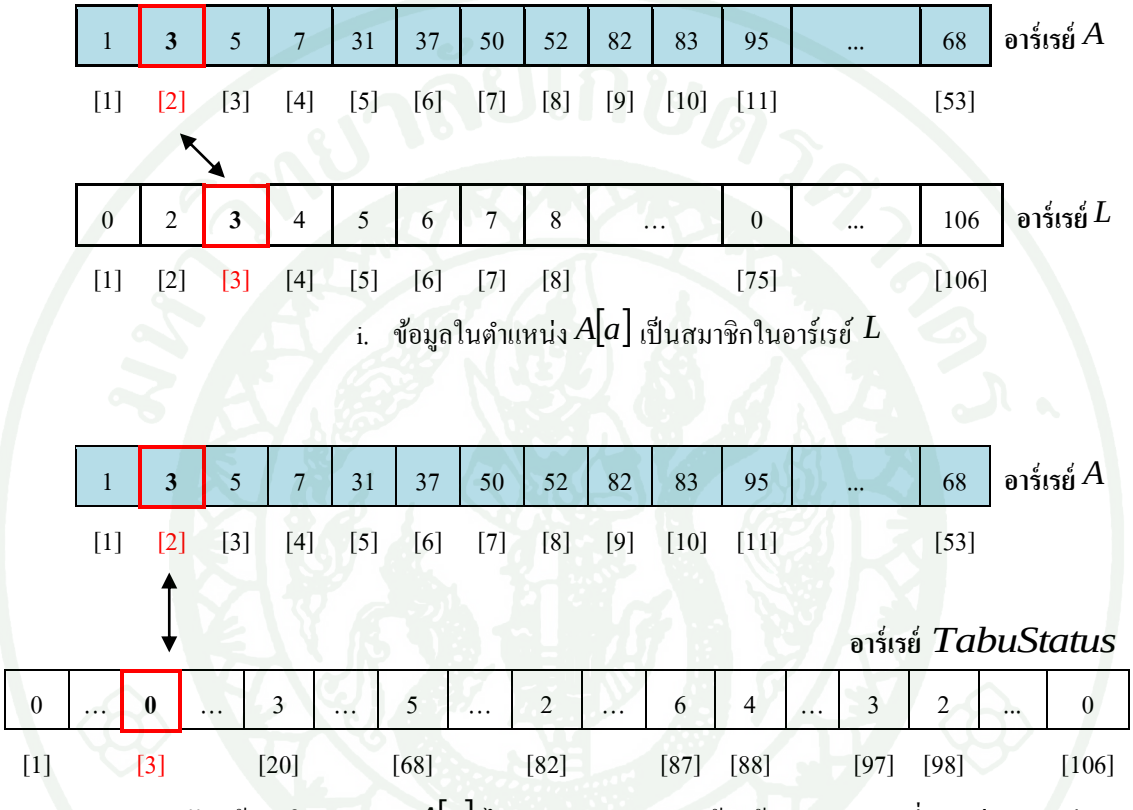


ภาพผนวกที่ ค22 การเพิ่มข้อมูล $A[a]$ และ $d[A[a]]$ ในอาร์เรย์ $RN[k][]$ กรณี $k=1$



ภาพผนวกที่ ค23 การลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L

10. พิจารณาแทรก “จุดบรรทุกสินค้า” อื่นๆลงในเส้นทางการเดินทางของรถบรรทุกคันที่ k โดยเลือกข้อมูลในอาร์เรย์ A ตำแหน่ง a ใดๆ($A[a]$)โดยที่ $1 \le a \le \frac{n}{2}$ ซึ่งข้อมูล $A[a]$ต้องเป็นสมาชิกในอาร์เรย์ L และไม่ถูกกำหนดสถานะ “ต้องห้าม” จากหมายเลขรถคันที่ k มาก่อน โดยการตรวจสอบจากอาร์เรย์ *TabuStatus* แสดงดังภาพผนวกที่ ค24



i. ข้อมูลในตำแหน่ง $A[a]$ เป็นสมาชิกในอาร์เรย์ L

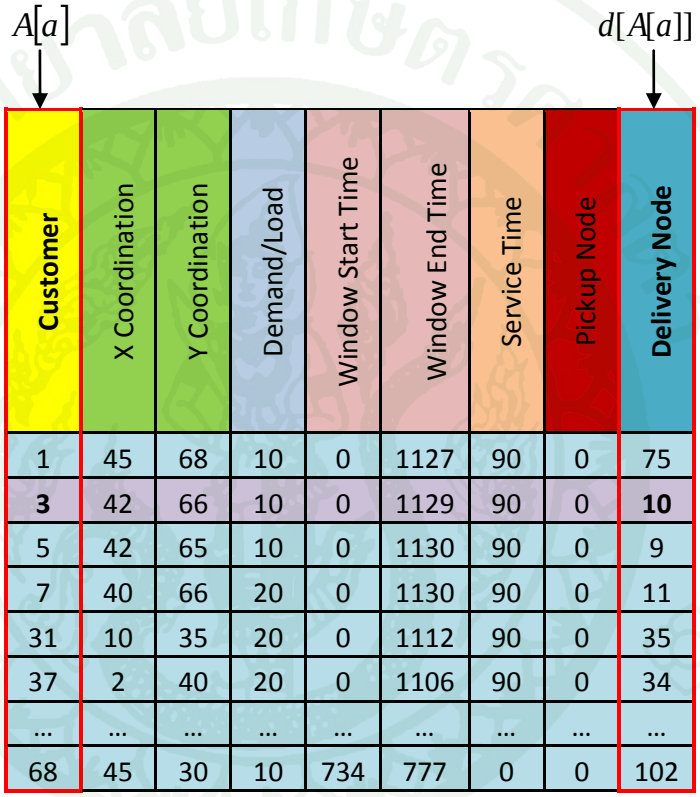
ii. ข้อมูลในตำแหน่ง $A[a]$ ไม่ถูกกำหนดสถานะ “ต้องห้าม” จากรถคันที่ k มาก่อน (กรณี k=1)

ภาพผนวกที่ ค10 วิธีการเลือก $A[a]$ กรณีการค้นหาคำตอบข้างเคียง

ทำการแทรก “จุดบรรทุกสินค้า ($A[a]$) ” และ“จุดส่งสินค้า ($d[A[a]]$)” ที่ได้พิจารณาลงในเส้นทางการเดินทางของรถบรรทุกคันที่ k ($RN[k][]$) แทรกทุกตำแหน่งที่เป็นไปได้บนเส้นทางเดินทางของรถบรรทุกคันเดิม เพื่อเลือกตำแหน่งที่แทรกแล้วได้ระยะเวลาที่ใช้เดินทางของรถสั้นที่สุด ซึ่งขั้นตอนการคำนวณระยะเวลานั้นอยู่ในเงื่อนไขของกรอบเวลาเปิด/ปิดทำการของร้านค้านั้นๆ, ผลรวมของระยะเวลาการเดินทางระหว่างร้านค้าที่อยู่ในเส้นทางการเดินรถอยู่ในเงื่อนไขของกรอบเวลาการทำงานของรถบรรทุก และขีดจำกัดการบรรทุกของรถตลอดเส้นทางการเดินทาง แสดงดังภาพผนวกที่ ค25 – ค26

	จุดจอดที่ 1	จุดจอดที่ 2	จุดจอดที่ 3	จุดจอดที่ 4	จุดจอดที่ 5	จุดจอดที่ 6	จุดจอดที่ 7	จุดจอดที่ 8	จุดจอดที่ …	จุดจอดที่ n
รถคันที่ 1	1	75	0	0	0	0	0	0	…	0

i. เส้นทางการเดินรถเดิมของรถคันที่ k (กรณี k=1)



Customer	X Coordination	Y Coordination	Demand/Load	Window Start Time	Window End Time	Service Time	Pickup Node	Delivery Node
1	45	68	10	0	1127	90	0	75
3	42	66	10	0	1129	90	0	**10**
5	42	65	10	0	1130	90	0	9
7	40	66	20	0	1130	90	0	11
31	10	35	20	0	1112	90	0	35
37	2	40	20	0	1106	90	0	34
…	…	…	…	…	…	…	…	…
68	45	30	10	734	777	0	0	102

ii. พิจารณา “จุดบรรทุกสินค้า ($A[a]$) ” และ“จุดส่งสินค้า ($d[A[a]]$)”

ภาพผนวกที่ ค25 วิธีการแทรก $A[a]$ และ $d[A[a]]$ ทุกตำแหน่งที่เป็นไปได้ภายในอาร์เรย์ $R[k][]$ เพื่อคำนวณหาระยะเวลาเดินทางรวมทั้งเส้นทาง และเลือกตำแหน่งที่แทรกแล้วได้ระยะทางที่สั้นที่สุดเป็นเส้นทางการเดินทางของรถบรรทุกคันที่ k

	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3	คอลัมน์ที่ 4	คอลัมน์ที่ 5	คอลัมน์ที่ 6	คอลัมน์ที่ 7	คอลัมน์ที่ 8	คอลัมน์ที่ ...	คอลัมน์ที่ n	เวลาเดินทางรวม
แทรกครั้งที่1	3	10	1	75	0	0	0	0	…	0	52
แทรกครั้งที่2	3	1	10	75	0	0	0	0	…	0	*
แทรกครั้งที่3	3	1	75	10	0	0	0	0	…	0	49
แทรกครั้งที่4	1	3	10	75	0	0	0	0	…	0	*
แทรกครั้งที่5	1	3	75	10	0	0	0	0	…	0	*
แทรกครั้งที่6	1	75	3	10	0	0	0	0	…	0	48

*ไม่สามารถคำนวณระยะเวลาเดินทางรวมได้เนื่องจากไม่อยู่ในเงื่อนไขบังคับ

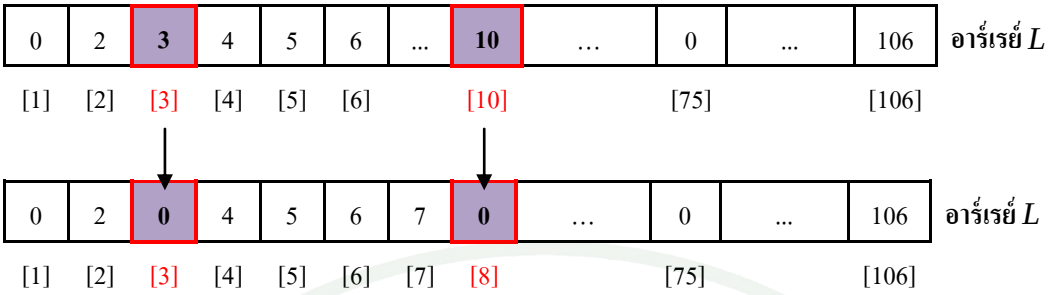
iii. วิธีการแทรก $A[a]$ และ $d[A[a]]$ ทุกตำแหน่งที่เป็นไปได้ในเส้นทางการเดินรถของรถคันที่ k

	จุดจอดที่ 1	จุดจอดที่ 2	จุดจอดที่ 3	จุดจอดที่ 4	จุดจอดที่ 5	จุดจอดที่ 6	จุดจอดที่ 7	จุดจอดที่ 8	จุดจอดที่ ...	จุดจอดที่ n
รถคันที่ 1	1	75	3	10	0	0	0	0	…	0

iv. เลือกตำแหน่งที่แทรกแล้วได้ระยะเวลาเดินทางรวมที่สั้นที่สุด กำหนดให้เป็นเส้นทางเดินรถใหม่ของรถคันที่ k

ภาพผนวกที่ ค26 วิธีการแทรก $A[a]$ และ $d[A[a]]$ ทุกตำแหน่งที่เป็นไปได้ภายในอาร์เรย์ $R[k][]$ เพื่อคำนวณหาระยะเวลาเดินทางรวมทั้งเส้นทาง และเลือกตำแหน่งที่แทรกแล้วได้ระยะทางที่สั้นที่สุดเป็นเส้นทางการเดินทางของรถบรรทุกคันที่ k (ต่อ)

เมื่อเลือกตำแหน่งการวางเส้นทางเดินรถที่ให้ระยะเวลาสั้นที่สุดแล้ว ทำการลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L เพื่อเป็นการทำสัญลักษณ์ว่า “ร้านค้าลำดับที่ $A[a]$ และ $d[A[a]]$ ได้ถูกรับบริการแล้ว” แสดงดังภาพผนวกที่ ค27



ภาพผนวกที่ ค27 การลบข้อมูล $A[a]$ และ $d[A[a]]$ ออกจากอาร์เรย์ L

11. ทำการพิจารณา “จุดบรรทุกสินค้า” อื่นๆที่มีการร้องขอการบริการด้วยวิธีการดังข้อ 10. จนกระทั่งไม่สามารถคำนวณหาระยะทางรวมทั้งเส้นทางได้ จึงหยุดการเพิ่มเส้นทางการเดินทางของรถบรรทุกคันที่ k

12. ทำการตรวจสอบข้อมูลสมาชิกทั้งหมดในอาร์เรย์ L

12.1 ถ้ามีสมาชิกบางตัว $\neq 0$ ให้ย้อนกลับไปทำข้อ 8 เพื่อคำนวณหาเส้นทางการเดินทางของรถบรรทุกคันที่ $k+1$

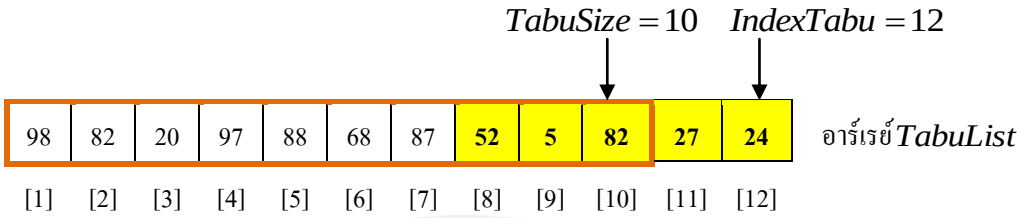
12.2 ถ้าสมาชิกทุกตัว $= 0$ แสดงว่าสิ้นสุดการหาคำตอบข้างเคียง

13. เปรียบเทียบคำตอบข้างเคียง กับคำตอบที่ดีที่สุด

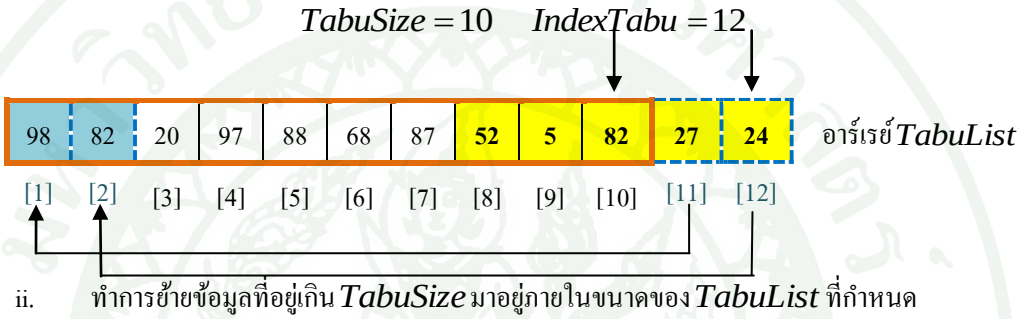
13.1 ถ้าคำตอบข้างเคียงสามารถให้คำตอบที่ดีกว่า หรือเท่ากับคำตอบที่ดีที่สุด คือ $k \leq MinCar$ ให้ทำการปรับคำตอบข้างเคียงนั้นเป็นคำตอบที่ดีที่สุด

14. ตรวจสอบขนาดของ $TabuList$ ซึ่งจัดเก็บข้อมูลที่อยู่ในสถานะ “ต้องห้าม” ก่อนการเริ่มค้นหาคำตอบข้างเคียงใหม่อีกครั้ง

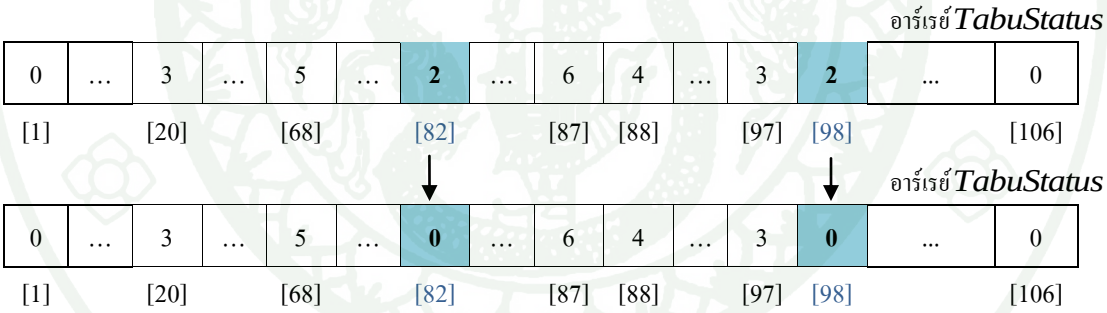
14.1 ถ้า $IndexTabu \rangle TabuSize$ ให้ทำการปลดข้อมูลบางส่วนออกจากการเป็นสถานะ “ต้องห้าม” พร้อมกับคืนค่า สถานะ“ปกติ” ข้อมูลบางส่วนจะได้รับการปลดปล่อยออกมาแสดงดังภาพผนวกที่ ค28



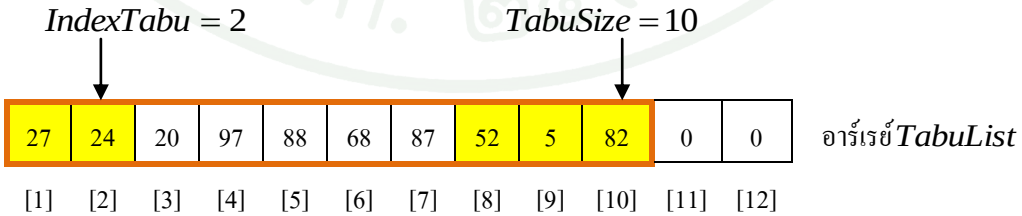
i. แสดงการเพิ่ม “จุดจอดบรรทุกสินค้า” ที่มีสถานะ“ต้องห้าม”ในอาร์เรย์ *TabuList* จนกระทั่งเกินขนาดที่สามารถรองรับได้ (กรณี $TabuSize = 10$)



ii. ทำการย้ายข้อมูลที่อยู่เกิน *TabuSize* มาอยู่ภายในขนาดของ *TabuList* ที่กำหนด



iii. ข้อมูลบางส่วนที่ถูกปลดในอาร์เรย์ *TabuList* จะถูกกำหนดเปลี่ยนสถานะ“ต้องห้าม” เป็น “ปกติ” ในอาร์เรย์ *TabuStatus* ด้วย



iv. ปรับ *IndexTabu* ให้เท่ากับจำนวนข้อมูลที่ถูกย้ายเข้ามา เพื่อชี้ตำแหน่งข้อมูลที่เพิ่มเข้ามาในครั้งถัดไป

ภาพผนวกที่ ค28 วิธีการตรวจสอบขนาดของ *TabuList*

15. ค้นหาคำตอบข้างเคียงอื่นๆ โดยการย้อนกลับไปทำซ้ำในข้อ 4. ใหม่อีกครั้ง จนกระทั้งครบจำนวนรอบ(Iteration)ที่กำหนดไว้

16. เมื่อทำการค้นหาคำตอบข้างเคียงจากคำตอบที่ดีที่สุดในแต่ละรอบที่กำหนด เสร็จสิ้น เราจะได้คำตอบที่เหมาะสมที่ได้จากการปรับปรุงคำตอบโดยวิธีการทาบู

ประวัติการศึกษา และการทำงาน

ชื่อ–นามสกุล	นางสาวกีรติญา ศรีมูล
วัน เดือน ปี ที่เกิด	วันที่ 21 เดือนพฤษภาคม พ.ศ. 2529
สถานที่เกิด	จังหวัดเชียงใหม่
ประวัติการศึกษา	วท.บ. (วิทยาการคอมพิวเตอร์) มหาวิทยาลัยศิลปากร