ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

ปริญญา

วิศวกรรมคอมพิวเตอร์ | วิศวกรรมคอมพิวเตอร์

สาขา | **ภาควิชา**

เรื่อง การพัฒนากรอบการทำงานแบบสมรรถนะสูงในการแก้ปัญหาทรานสปอร์ตเทชั่นซิมูเลชั่นและออฟติไมเซชั่น

The Development of a High Performance Framework for Transportation Simulation and Optimization Problem

นามผู้วิจัย นายรักพงค์ แก้วพวง

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

(รองศาสตราจารย์วรา วราวิทย์, Ph.D.)

หัวหน้าภาควิชา

(ผู้ช่วยศาสตราจารย์เขมะฑัต วิภาตะวนิช, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

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

วิทยานิพนธ์

เรื่อง

การพัฒนากรอบการทำงานแบบสมรรถนะสูงในการแก้ปัญหา ทรานสปอร์ตเทชั่นซิมูเลชั่นและออฟติไมเซชั่น

The Development of a High Performance Framework for Transportation Simulation and Optimization Problem

โดย

นายรักพงค์ แก้วพวง

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)
พ.ศ. 2553

รักพงค์ แก้วพวง 2553: การพัฒนากรอบการทำงานแบบสมรรถนะสูงในการแก้ปัญหาทรานสปอร์ตเทชั่นซิมูเลชั่นและออฟติไมเซชั่น ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D. 128 หน้า

วิทยานิพนธ์นี้เสนอแนวความคิดในการพัฒนากรอบการทำงานแบบสมรรถนะสูงสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งเป็นปัญหาที่เกิดขึ้นจริงในอุตสาหกรรมการขนส่ง ปัญหานี้โดยปกติจะมีขนาดใหญ่และซับซ้อน ทำให้การประมวลผลบนคอมพิวเตอร์เครื่องเดียวใช้เวลานานมาก ในวิทยานิพนธ์นี้จึงเสนอกรอบการทำงานแบบสมรรถนะสูงที่สามารถรวบรวมพลังการประมวลผลจากหลายหน่วยประมวลผลให้กับการแก้ปัญหา และมีความสามารถในการทำงานบนเครื่องแบบมัลติคอร์และมัลติคอร์คลัสเตอร์ได้โดยไม่ต้องปรับแก้โปรแกรมแต่อย่างใด ในงานวิจัยนี้ได้ประยุกต์ใช้เทคโนโลยีของไมโครซอฟท์ ซีซีอาร์และดีเอสเอส ซึ่งสามารถซ่อนความซับซ้อนของการคำนวณแบบขนานจากผู้ใช้ได้อย่างดี จากการทดสอบพบว่า โปรแกรมต้นแบบที่พัฒนาขึ้นสามารถลดเวลาในการประมวลผลได้อย่างมาก โดยในกรณีที่ดีที่สุดของการทดสอบพบว่า เวลาในการประมวลผลลดถึง 56.68 เท่า เมื่อใช้มัลติคอร์คลัสเตอร์ จำนวน 16 โหนด 64 คอร์ ในการประมวลผล

_____ _____ ___/___/___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Rakpong Kaewpuang 2010: The Development of a High Performance Framework for Transportation Simulation and Optimization Problem. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Assistant Professor Putchong Uthayopas, Ph.D. 128 pages.

This thesis presents a high performance framework for PDPTW (Pickup and Delivery Problem with Time Windows) which is an important problem in transportation industry. Usually, this problem is a very large and complex problem which requires a very long computation time on a single computer. This thesis propose a framework that collect the computing power from many processors and use it to solve this problem. Furthermore, this framework allows the problem being developed to run on a single multicore computer and scale to multicore cluster without change. In this work, microsoft CCR/DSS is used to hide the complexity of parallel computing from users. From the experiments, it has been found that the prototype application can reduce the computation time significantly. For the best case, the execution time is reduced by 56.68 times using 16 node 64 core cluster running in parallel.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ภุชงค์ อุทโยภาศ ประธานกรรมการที่ปรึกษาวิทยานิพนธ์ รองศาสตราจารย์ ดร.วรา วราวิทย์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ผู้ช่วยศาสตราจารย์ ดร.จุฑา พิชิตลำเค็ญ และ รองศาสตราจารย์ ดร.พีรยุทธ์ ชาญเศรษฐิกุล สำหรับความรู้ทางด้านวิชาการ ตลอดจนคำแนะนำในการดำเนินชีวิต และขอกราบขอบพระคุณ คณาจารย์และเจ้าหน้าที่ของภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่าน รวมถึงพี่ๆ และน้องๆ ในศูนย์วิจัยคอมพิวเตอร์สมรรถนะสูงและเครือข่ายคอมพิวเตอร์ สำหรับการสั่งสอนและความช่วยเหลือต่างๆ ซึ่งทำให้วิทยานิพนธ์เล่มนี้ เสร็จสมบูรณ์ได้ ขอกราบขอบพระคุณบุคคลที่สำคัญที่สุดในชีวิตของผู้วิจัย คือ พ่อ แม่ และพี่ชาย ครอบครัวของผู้วิจัย สำหรับความรักและความห่วงใยตลอดมา ซึ่งเป็นกำลังใจและเป็นแรงผลักดันที่สำคัญอย่างยิ่งในการเรียนและการทำวิทยานิพนธ์เล่มนี้

ผู้วิจัยหวังเป็นอย่างยิ่งว่าวิทยานิพนธ์เล่มนี้ สามารถสร้างประโยชน์ให้กับผู้ที่สนใจ ซึ่งถ้าหากมีข้อผิดพลาดประการใด ต้องขออภัยมา ณ ที่นี้

รักพงค์ แก้วพวง

พฤษภาคม 2553

สารบัญ

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

การพัฒนากรอบการทำงานแบบสมรรถนะสูงในการแก้ปัญหาทรานสปอร์ตเทชั่นซิมูเลชั่นและออฟติไมเซชั่น

The Development of a High Performance Framework for Transportation Simulation and Optimization Problem

คำนำ

ในปัจจุบันการใช้แบบจำลองและการใช้ออฟติไมเซชั่นในการออกแบบผลิตภัณฑ์กลายเป็นสิ่งจำเป็นในอุตสาหกรรมสมัยใหม่ ดังเช่น การจำลองการชนกัน (Crash simulation) ในอุตสาหกรรมยานยนต์, การออกแบบยาในอุตสาหกรรมยา (Virtual screening), การจำลองการเกิดมลพิษ (Pollution simulation) เพื่อการปกป้องสภาวะแวดล้อมและการแก้ปัญหาของอุตสาหกรรมการขนส่ง (Transportation optimization) ที่ช่วยลดเวลา, ลดค่าขนส่ง และประหยัดพลังงาน ซึ่งการใช้แบบจำลองและการใช้ออฟติไมเซชั่นนี้มีการคำนวณที่ซับซ้อน, ต้องการพลังในการประมวลผลที่สูงและต้องการขนาดของหน่วยเก็บข้อมูลจำนวนมหาศาล ทำให้ระบบคอมพิวเตอร์สมรรถนะสูงที่มีเนื้อที่เก็บข้อมูลจำนวนมากหรือขนาดใหญ่เป็นสิ่งที่ต้องการขององค์กรทั้งภาครัฐและเอกชน เพื่อใช้ในการวิจัยและทางด้านธุรกิจ ซึ่งวัตถุประสงค์ขององค์กร คือ เพื่อเพิ่มประสิทธิภาพในการทำงานเพื่อให้ได้ผลลัพธ์ตรงตามวัตถุประสงค์โดยใช้ระยะเวลาในการหาผลลัพธ์น้อยที่สุด ทำให้คอมพิวเตอร์สมรรถนะสูงเป็นที่ต้องการขององค์กร แต่เนื่องจากเครื่องคอมพิวเตอร์สมรรถนะสูงมีราคาและค่าใช้จ่ายในการบำรุงรักษาที่สูงทำให้องค์กรต้องเสียค่าใช้จ่ายเป็นจำนวนมาก แนวคิดของการกระจายงานและการทำงานแบบขนาน สามารถนำมาประยุกต์ใช้เพื่อช่วยลดระยะเวลาของการทำงานและสามารถรองรับกับงานที่ต้องใช้ความสามารถของคอมพิวเตอร์ในการทำงานได้ดีเทียบเท่ากับคอมพิวเตอร์สมรรถนะสูง ซึ่งทำให้ประหยัดค่าใช้จ่ายได้เป็นอย่างมาก และด้วยการเปลี่ยนแปลงทางด้านเทคโนโลยีทำให้เกิดเทคโนโลยีที่ช่วยเพิ่มประสิทธิภาพของการประมวลผล (Asanovic *et al.*, 2006) คือ เทคโนโลยีกริด (Grid), คลัสเตอร์ (Cluster) และเทคโนโลยีมัลติคอร์ (Multicore)

เทคโนโลยีที่สามารถรองรับการคำนวณแบบขนานได้อย่างมีประสิทธิภาพ คือ ระบบคลัสเตอร์ เทคโนโลยีกริด และเทคโนโลยีมัลติคอร์ ระบบคลัสเตอร์จะเป็นการนำเครื่องคอมพิวเตอร์หลายเครื่องมาเชื่อมต่อกัน และสามารถทำงานประสานกันโดยการสร้างระบบคลัสเตอร์ขึ้นมาก็เพื่อเพิ่มความเร็วของการทำงานโดยมีการใช้เทคโนโลยีการประมวลผลแบบขนานเข้ามาร่วมด้วย ในส่วนของเทคโนโลยี

กริดเป็นเทคโนโลยีที่มุ่งในการรวบรวมทรัพยากรต่างๆ ของระบบคอมพิวเตอร์ที่กระจายกันอยู่บนระบบเครือข่าย (Network) มาใช้งานร่วมกันเสมือนเป็นระบบเดียวกันให้กับผู้ใช้ ทำให้พลังในการประมวลผลสูงขึ้น ซึ่งทั้งระบบคลัสเตอร์และเทคโนโลยีกริดเป็นการรวมคอมพิวเตอร์หลายเครื่องมาช่วยกันทำงาน ทำให้การที่จะพัฒนาโปรแกรมแบบขนานบนระบบเหล่านี้เป็นเรื่องที่ค่อนข้างยุ่งยาก เพราะต้องมีค่าใช้จ่ายเกี่ยวกับคอมพิวเตอร์ ซึ่งในปัจจุบันเราสามารถที่จะพัฒนาโปรแกรมแบบขนานได้บนเครื่องคอมพิวเตอร์หนึ่งเครื่อง เพราะด้วยเทคโนโลยีมัลติคอร์ที่สามารถตอบสนองการทำงานได้หลายงานพร้อมกัน

คอมพิวเตอร์ในปัจจุบันได้มีการพัฒนาเทคโนโลยีในการประมวลผลที่สามารถทำงานหลายงานได้พร้อมกันบนคอมพิวเตอร์หนึ่งเครื่อง เพราะด้วยเทคโนโลยีที่เรียกว่ามัลติคอร์โปรเซสเซอร์ (Multicore processor) ที่นำโปรเซสเซอร์(Processor) หลายตัวมารวมไว้ที่โปรเซสเซอร์เดียว ซึ่ง ณ เวลานี้เทคโนโลยีมัลติคอร์สามารถที่จะนำโปรเซสเซอร์ตั้งแต่ 2, 4, 6 และ 8 โปรเซสเซอร์มารวมไว้ในหนึ่งโปรเซสเซอร์ ทำให้สมรรถนะและประสิทธิภาพในการประมวลผลสูงขึ้น ลดระยะเวลาในการทำงาน และประหยัดพลังงาน ซึ่งทั้งหมดรวมอยู่ภายในเครื่องคอมพิวเตอร์หนึ่งเครื่อง ทำให้เครื่องคอมพิวเตอร์มีสมรรถนะการทำงานเทียบเท่ากับคอมพิวเตอร์สมรรถนะสูง และด้วยราคาที่ถูกลง ทำให้สามารถหาซื้อได้ง่ายและเป็นที่แพร่หลาย ทำให้คอมพิวเตอร์ในปัจจุบันสามารถที่จะพัฒนาโปรแกรมแบบขนานได้ง่ายขึ้น

การพัฒนาโปรแกรมแบบขนานบนเทคโนโลยีมัลติคอร์สามารถที่จะทำได้หลายวิธี เช่น การพัฒนาโปรแกรมแบบ Message passing, การพัฒนาโปรแกรมแบบ Multithreading (Carver and Tai, 2006) และการพัฒนาโปรแกรมแบบ Concurrency and Coordination Runtime/Decentralized Software Services (CCR/DSS) ซึ่ง CCR/DSS เป็นส่วนหนึ่งที่ทางไมโครซอฟท์ได้พัฒนาชุดเครื่องมือที่ใช้สำหรับพัฒนาแอพพลิเคชั่น (Application) เกี่ยวกับหุ่นยนต์ที่เรียกว่า Microsoft Robotics Developer Studio (MRDS) (Microsoft Robotics, 2008)โดยการเลือกเทคนิคสำหรับการพัฒนาโปรแกรมแบบขนานที่เหมาะสมเป็นสิ่งสำคัญอย่างยิ่งในการที่จะทำให้การพัฒนาเป็นไปได้ง่ายและมีสมรรถนะสูง ซึ่งการพัฒนาโปรแกรมแบบขนานนี้ สามารถที่จะช่วยเพิ่มสมรรถนะและประสิทธิภาพของการทำงาน แต่ในการพัฒนาโปรแกรมแบบขนานยังมีความซับซ้อนและต้องใช้เทคนิคในการกระจายงาน เพื่อให้เหมาะสมกับแต่ละชนิดของงาน ซึ่งเป็นเรื่องที่ยุ่งยากสำหรับการพัฒนา

ปัจจุบันในภาคธุรกิจและอุตสาหกรรมด้านต่างๆ มีการเจริญเติบโตทางเศรษฐกิจที่รวดเร็ว, การแข่งขันสูงและมีการเปลี่ยนแปลงของต้นทุนในด้านต่างๆ อย่างรวดเร็ว ทำให้ต้องพบกับปัญหาในการตัดสินใจที่มีความซับซ้อนสูงเนื่องจากมีองค์ประกอบด้านต่างๆเข้ามาเกี่ยวข้อง ปัญหาการขนส่ง (Transportation problem) เป็นหนึ่งในปัญหาที่ต้องมีการตัดสินใจที่ซับซ้อน ต้องการคำตอบของการตัดสินใจที่ดีที่สุด ซึ่งหมายถึง ผลกำไรสูงสุด, ค่าใช้จ่ายในการดำเนินการน้อยที่สุดและคำตอบมีประสิทธิภาพสูงสุด เป็นต้น โดยวัตถุประสงค์ของปัญหาการขนส่ง คือ การส่งสินค้าให้มีประสิทธิภาพสูงสุด ทำให้การตัดสินใจต้องคำนึงถึงเงื่อนไขหลายด้าน อาทิเช่น ราคาน้ำมัน, เส้นทาง, ชนิดของยานพาหนะและเวลาเปิด/ปิดของร้านค้า เป็นต้น ดังนั้นปัญหาการขนส่งจึงต้องใช้เวลาในการประมวลผลที่สูงสำหรับการตัดสินใจเพื่อให้เป็นไปอย่างมีประสิทธิภาพสูงสุดหรือเป็นที่น่าพอใจ

ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง (Pickup and Delivery Problem with Time Windows) ในทางเทคนิคจะเรียกว่าเป็นปัญหาของการให้บริการของการขนส่งสินค้าจากการร้องขอ (Request) จากจุดรับ/ส่งโดยใช้ยานพาหนะ (Vehicle) ที่มีอยู่อย่างจำกัด ซึ่งแต่ละการร้องขอจะมีความเกี่ยวข้องกับการเคลื่อนย้ายปริมาณของสินค้าจากจุดรับสินค้า (Pickup location) ไปส่งยังจุดส่งสินค้า (Delivery location) ซึ่งการรับ/ส่งสินค้านั้นจะมีช่วงเวลา (Time window) ในการรับและการส่งสินค้าของแต่ละร้านค้าเป็นเงื่อนไขในการกำหนดเวลาของการรับ/ส่งสินค้า ซึ่งทำให้การพิจารณาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จำเป็นต้องมีทรัพยากรด้านการประมวลผลของคอมพิวเตอร์ที่สูง ทำให้อาจไม่สามารถคำนวณหรือเวลาที่ใช้ในการประมวลผลอาจยาวนานเกินกว่าที่จะยอมรับได้บนเครื่องคอมพิวเตอร์เครื่องเดียว ดังนั้น ด้วยเหตุผลเหล่านี้จึงมีความจำเป็นที่ต้องทำการพัฒนาการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เพื่อลดเวลาในการแก้ปัญหา และทำให้สามารถรองรับกับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งที่มีขนาดใหญ่ได้มากยิ่งขึ้น

เหตุผลในการพัฒนาการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งของงานวิจัยนี้ คือ

1. การสร้างระบบคอมพิวเตอร์สมรรถนะสูงที่รองรับการคำนวณแบบขนานทั้งแบบ Message passing และ Multithreading สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง หรือเรียกได้ว่าเป็นระบบมัลติคอร์คลัสเตอร์ โดยใช้

เครื่องมัลติคอร์คอมพิวเตอร์ส่วนบุคคลในองค์กรที่เชื่อมต่อกันบนระบบเครือข่ายเดียวกัน เพื่อลดต้นทุนในการจัดหาเครื่องคอมพิวเตอร์มาใช้ในการสร้างระบบ และการติดตั้งซอฟต์แวร์ต่างๆสำหรับสร้างระบบต้องง่ายสำหรับผู้ใช้ทั่วไป อีกทั้งระบบมัลติคอร์คลัสเตอร์เมื่อไม่ต้องการใช้งานก็สามารถนำเครื่องคอมพิวเตอร์เหล่านั้นกลับไปใช้ทำงานได้เหมือนเดิม

2. ระบบมัลติคอร์คลัสเตอร์ที่รองรับการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่มีความสามารถทนต่อความผิดพร่อง (Fault-tolerance) ของระบบ เมื่อเกิดความล้มเหลวจากการทำงานในกลุ่มคอมพิวเตอร์เพื่อให้ระบบสามารถดำเนินการต่อไปได้

3. การใช้งานระบบมัลติคอร์คลัสเตอร์ที่รองรับการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งนั้น เพื่อให้ง่ายสำหรับผู้ใช้ ระบบได้จัดเตรียม Graphic User Interface (GUI) สำหรับรองรับกับผู้ใช้ เพื่อให้สะดวกกับการใช้งานระบบ และซ่อนความซับซ้อนของระบบการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ทำให้ผู้ใช้รู้สึกเสมือนทำงานอยู่บนเครื่องคอมพิวเตอร์เครื่องเดียว

โดยสรุป งานวิจัยนี้เสนอการออกแบบและพัฒนาระบบตามแนวคิดของกรอบการทำงานสมรรถนะสูงเพื่อรองรับการคำนวณแบบขนานสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง โดยสร้างระบบคอมพิวเตอร์สมรรถนะสูงจากเครื่องมัลติคอร์คอมพิวเตอร์ส่วนบุคคลในองค์กร สำหรับรองรับการคำนวณแบบขนานเพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง โดยระบบมัลติคอร์คลัสเตอร์สามารถทนต่อความผิดพร่องของระบบและซ่อนความซับซ้อนของการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งให้กับผู้ใช้ ซึ่งในงานวิจัยนี้เลือกใช้เครื่องมัลติคอร์คอมพิวเตอร์ส่วนบุคคลที่ติดตั้งระบบปฏิบัติการไมโครซอฟท์วินโดวส์ (Microsoft windows) สำหรับการสร้างระบบมัลติคอร์คลัสเตอร์ โดยงานวิจัยนี้ใช้เทคโนโลยี Concurrency and Coordination Runtime/Decentralized Software Services (CCR/DSS) ของไมโครซอฟท์ในการพัฒนาส่วนประกอบต่างๆของระบบการคำนวณแบบขนานทั้งแบบ Message passing และ Multithreading สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เหตุผลที่งานวิจัยนี้เลือกใช้เทคโนโลยี

CCR/DSS เนื่องจากเป็นเทคโนโลยีที่สนับสนุนการพัฒนาโปรแกรมแบบ Message passing และแบบ Multithreading ซึ่งสามารถเพิ่มประสิทธิภาพในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งได้ดียิ่งขึ้น

วัตถุประสงค์

เสนอแนวคิดของกรอบการทำงานสมรรถนะสูงบนระบบมัลติคอร์คลัสเตอร์ที่สามารถทำงานบนระบบมัลติคอร์เครื่องเดียวและสามารถขยายเป็นระบบมัลติคอร์คลัสเตอร์ได้ง่าย โดยการพัฒนาจะใช้เครื่องมือซอฟต์แวร์ที่มีใช้กันแพร่หลาย

ขอบเขตการวิจัย

1. ออกแบบและพัฒนาแนวคิดของกรอบการทำงานสมรรถนะสูงเพื่อรองรับการคำนวณแบบขนานสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

2. พัฒนาซอฟต์แวร์ตามแนวทางที่นำเสนอโดยใช้เทคนิคการประมวลผลแบบขนานแบบ Concurrency and Coordination Runtime/Decentralized Software Services (CCR/DSS)

3. สร้างซอฟต์แวร์เพื่อยืนยันความถูกต้องโดยพัฒนาซอฟต์แวร์สำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

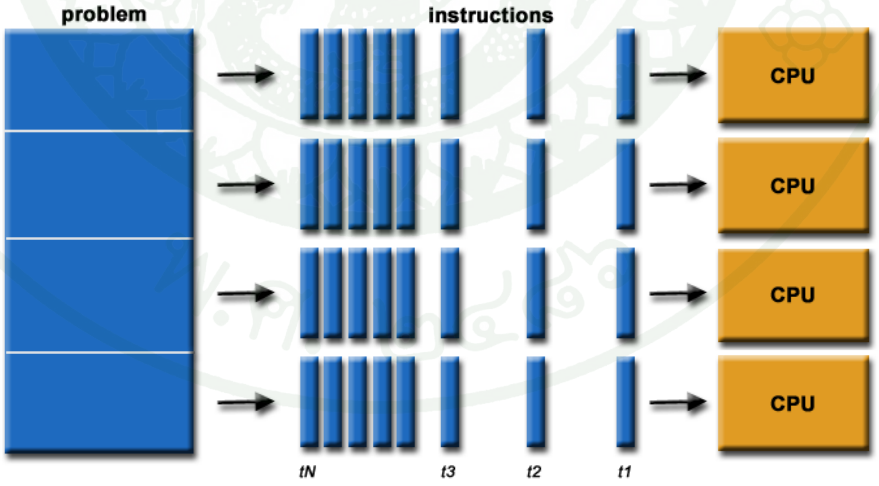
4. ทดสอบประสิทธิภาพของระบบที่ได้พัฒนา

การตรวจเอกสาร

ความรู้พื้นฐาน

งานวิจัยนี้ออกแบบและพัฒนาระบบสำหรับสร้างมัลติคอร์คลัสเตอร์ เพื่อรองรับการคำนวณแบบขนานทั้ง 2 แบบ คือ Message passing และ Multithreading สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง โดยมีความรู้พื้นฐานที่กล่าวต่อไปนี้เป็นการให้นิยามและบรรยายแนวคิดของความรู้พื้นฐาน

แนวคิดการประมวลผลแบบขนาน (Parallel Computing) (Kumar *et al*., 1994) เป็นกระบวนการแก้ปัญหาโดยการแบ่งปัญหาออกเป็นส่วนๆ ที่มีขนาดเล็กลง แล้วนำปัญหาแต่ละส่วนที่ได้จากการแบ่งไปคำนวณยังหน่วยประมวลผลหลายๆ ตัว ในเวลาเดียวกัน เพื่อลดเวลาในการแก้ปัญหาลง โดยลักษณะของการประมวลผลแบบขนานสามารถแสดงดังภาพที่ 1 ซึ่งแสดงการทำงานโดยใช้หน่วยประมวลผลหลายหน่วยประมวลผล และปัญหาจะถูกแบ่งออกเป็นส่วนๆ ที่สามารถแก้ปัญหาพร้อมกันได้ แต่ละส่วนจะถูกแบ่งเป็นชุดของคำสั่ง จากนั้นแต่ละคำสั่งจะถูกทำงานบนหน่วยประมวลผลแต่ละหน่วยประมวลผลพร้อมกัน



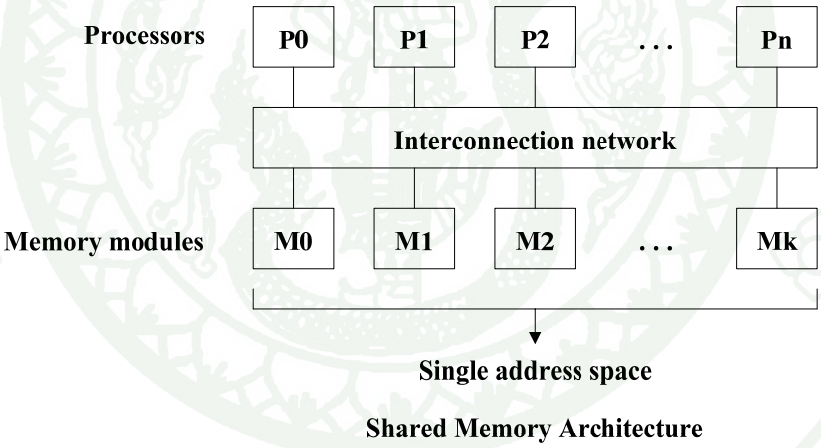
ภาพที่ 1 ลักษณะของการประมวลผลแบบขนาน

โครงสร้างของคอมพิวเตอร์แบบขนานแบ่งเป็น 2 ชนิดดังต่อไปนี้

1. โครงสร้างแบบที่ใช้หน่วยความจำร่วม (Shared memory) (Dongarra *et al*., 2003) โครงสร้างลักษณะนี้ประกอบด้วยกลุ่มของหน่วยประมวลผลที่เชื่อมต่อกันด้วยเครือข่ายภายในความเร็วสูงโดยที่หน่วยประมวลผลทั้งหมดจะใช้หน่วยความจำร่วมกัน ทำให้สามารถลดระยะเวลาในการติดต่อระหว่างหน่วยประมวลผลได้ ซึ่งมีลักษณะดังภาพที่ 2

- ข้อดีของโครงสร้างแบบนี้ คือ ง่ายสำหรับการเขียนโปรแกรมแบบขนาน เพราะมีการอ้างอิงที่หน่วยความจำแห่งเดียว

- ข้อเสียของโครงสร้างแบบนี้ คือ โปรแกรมเมอร์ (Programmer) จะต้องจัดการกับการประสานการทำงาน (Synchronization) เพื่อให้มั่นใจว่าการเข้าถึงข้อมูลในหน่วยความจำถูกต้อง

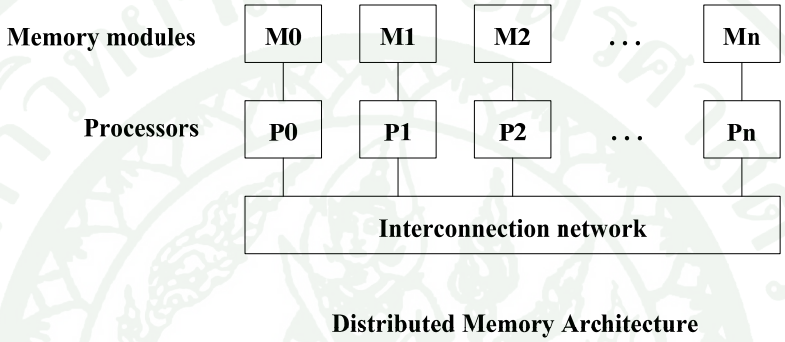


ภาพที่ 2 โครงสร้างคอมพิวเตอร์แบบขนานที่ใช้หน่วยความจำร่วม

2. โครงสร้างแบบที่ใช้หน่วยความจำแยก (Distributed memory) (Dongarra *et al*., 2003) โครงสร้างลักษณะนี้ประกอบด้วยกลุ่มของหน่วยประมวลผลที่เชื่อมต่อกันด้วยเครือข่ายความเร็วสูงโดยที่หน่วยประมวลผลแต่ละตัวจะมีหน่วยความจำแยกจากกัน การสื่อสารทำได้โดยการแลกเปลี่ยนข้อความ (Message) ผ่านเครือข่าย ดังภาพที่ 3

- ข้อดีของโครงสร้างแบบนี้ คือ หน่วยความจำมีขนาดใหญ่มากขึ้น และแต่ละหน่วยประมวลผลสามารถเข้าถึงหน่วยความจำของตัวเองได้โดยปราศจากการรบกวนจากหน่วยประมวลผลอื่น

- ข้อเสียของโครงสร้างแบบนี้ คือ เพิ่มความยุ่งยากในการเขียนโปรแกรมแบบขนาน เพราะโปรแกรมเมอร์จะต้องรับผิดชอบในการส่งผ่านข้อมูลระหว่างหน่วยประมวลผล



ภาพที่ 3 โครงสร้างคอมพิวเตอร์แบบขนานที่ใช้หน่วยความจำแยก

การวัดประสิทธิภาพของโปรแกรมแบบขนาน

การวัดประสิทธิภาพของโปรแกรมแบบขนาน เพื่อแสดงให้เห็นถึงประสิทธิภาพการทำงานแบบขนานว่ามีความเร็วในการแก้ปัญหาได้เร็วเท่าใด และแสดงให้เห็นถึงประสิทธิภาพของการใช้หน่วยประมวลผลได้ดีเพียงใด โดยการวัดประสิทธิภาพของโปรแกรมแบบขนานสามารถวัดได้จากค่าดังต่อไปนี้

Speedup (Quinn, 2003) คือ อัตราส่วนระหว่างเวลาที่ใช้ในการประมวลผลของโปรแกรมแบบลำดับและเวลาที่ใช้ในการประมวลผลของโปรแกรมแบบขนาน ซึ่งค่าที่ได้จาก Speedup แสดงให้เห็นถึงความเร็วในการประมวลผลแบบขนานมีความเร็วเป็นจำนวนกี่เท่าของความเร็วในการประมวลผลแบบลำดับซึ่งมีสมการดังนี้

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

$T_{Sequential}$ คือ เวลาที่ใช้ในการประมวลผลของโปรแกรมแบบลำดับ

$T_{Parallel}$ คือ เวลาที่ใช้ในการประมวลผลของโปรแกรมแบบขนาน

Efficiency (Quinn, 2003) เป็นการวัดการใช้ประโยชน์โปรเซสเซอร์ของโปรแกรมแบบขนาน ซึ่งมีอัตราส่วนระหว่าง Speedup กับจำนวนของโปรเซสเซอร์ที่ถูกใช้งาน ซึ่งมีสมการดังนี้

$$\text{Efficiency} = \frac{T_{Sequential}}{P \times T_{Parallel}} \quad (2)$$

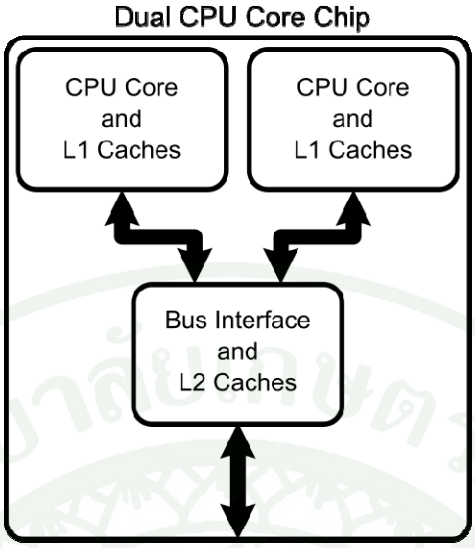
P คือ จำนวนของโปรเซสเซอร์ที่ถูกใช้

$T_{Sequential}$ คือ เวลาที่ใช้ในการประมวลผลของโปรแกรมแบบลำดับ

$T_{Parallel}$ คือ เวลาที่ใช้ในการประมวลผลของโปรแกรมแบบขนาน

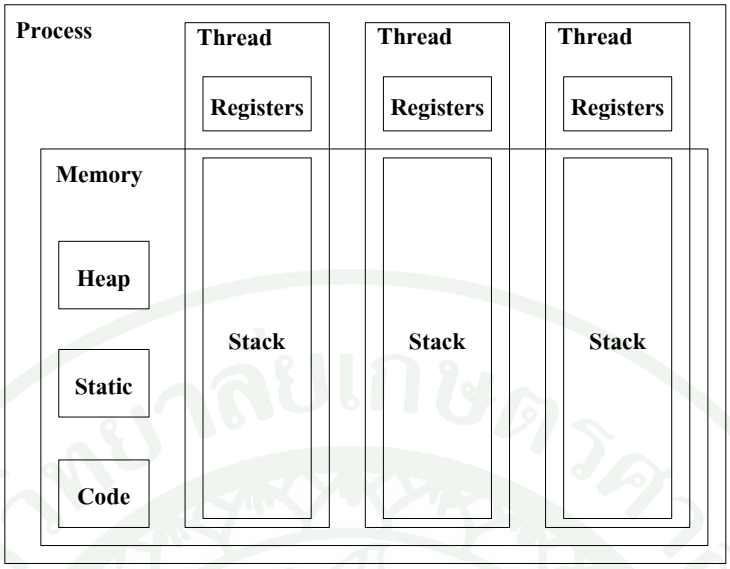
Throughput คือ หน่วยวัดประสิทธิภาพในการทำงานของเครื่องคอมพิวเตอร์ โดยวัดจากปริมาณงานที่ได้เมื่อเทียบกับเวลาที่ใช้ไปในการทำงาน

มัลติคอร์โปรเซสเซอร์ (Multicore Processor) (Dongarra, 2007) ปัจจุบันเทคโนโลยีมัลติคอร์โปรเซสเซอร์เป็นที่นิยมใช้งานกันอย่างแพร่หลาย เพราะสามารถรองรับการทำงานได้หลายงานในเวลาเดียว โดยภายในหน่วยประมวลผลนั้นประกอบด้วยหน่วยประมวลผลย่อย เรียกว่า คอร์ (Core) โดยแต่ละคอร์จะมีแคช (Cache) เป็นของตัวเองทำให้สามารถเข้าถึงข้อมูลในแคชได้อย่างรวดเร็ว และยังประกอบด้วยแคชที่แต่ละคอร์จะต้องใช้ร่วมกัน ซึ่งเทคโนโลยีแบบมัลติคอร์สามารถทำให้คอมพิวเตอร์หนึ่งเครื่องทำงานเสมือนกับคอมพิวเตอร์สมรรถนะสูง ซึ่งสามารถทำการประมวลผลแบบขนานได้อย่างมีประสิทธิภาพ และปัจจุบันนี้ได้มีโปรเซสเซอร์ที่มีจำนวนคอร์เพิ่มมากขึ้นถึง 8 คอร์ โดยตัวอย่างโครงสร้างทั่วไปของ Dual core processor ซึ่งเป็นส่วนหนึ่งของเทคโนโลยีมัลติคอร์โปรเซสเซอร์แสดงดังภาพที่ 4



ภาพที่ 4 โครงสร้างทั่วไปของ Dual core processor

มัลติเทรด (Multithread) (Tanenbaum, 2001) เทรด (Thread) คือส่วนประกอบย่อยของโปรเซส (Process) ซึ่งเทรดบางครั้งถูกเรียกว่า ไลท์เวทโปรเซส (Lightweight process) ซึ่งในระบบปฏิบัติการที่สามารถรองรับการสร้างเทรดได้หลายๆ เทรดในหนึ่งโปรเซส จะเรียกว่า มัลติเทรด ซึ่งมัลติเทรดสามารถรองรับการทำงานหลายๆ งานได้พร้อมกัน โดยไม่ต้องมีการเปลี่ยนจากโปรเซสหนึ่งไปสู่อีกโปรเซสหนึ่ง โดยเทรดจะใช้แอดเดรส (Address) ร่วมกัน โดยภายในโปรเซสที่ประกอบด้วยเทรดจะมีการใช้โค้ดร่วมกัน (Share code), ทรัพยากรต่างๆ (Resource) และข้อมูลต่างๆ โดยโปรเซสที่ควบคุมหลายๆ เทรดนั้นจะสามารถตอบสนองต่อความต้องการของผู้ใช้จำนวนมาก ซึ่งตัวอย่างของมัลติเทรดสามารถแสดงดังภาพที่ 5



ภาพที่ 5 โครงสร้างของมัลติเทรด

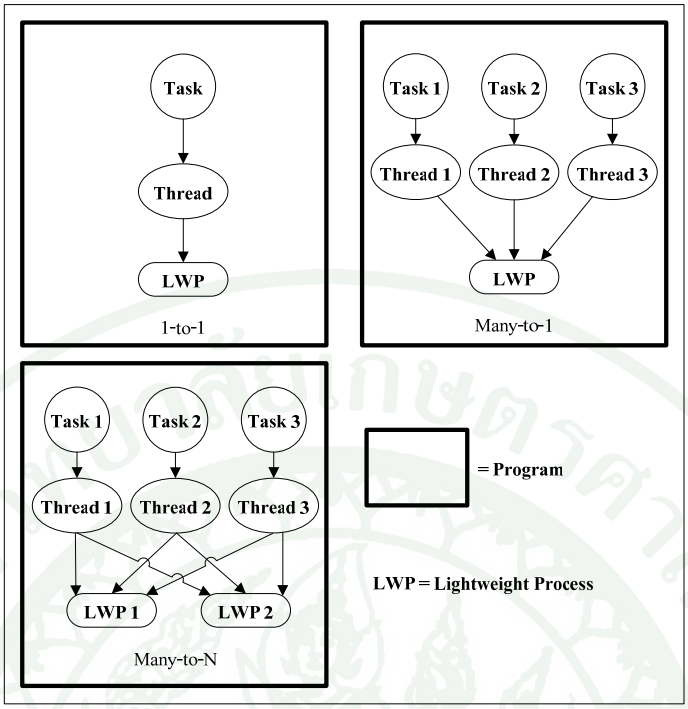
โมเดล (Model) ของมัลติเทรดมี 3 โมเดลดังต่อไปนี้

1. One-to-one เป็นโมเดลที่สามารถทำให้แต่ละเทรดสามารถทำงานได้พร้อมกัน โดยข้อจำกัดของโมเดลนี้ คือ การกำหนดจำนวนเทรดให้เหมาะกับงานหนึ่งๆ

2. Many-to-one เป็นโมเดลที่มีหลายเทรดแต่ไม่สามารถที่จะให้เทรดทำงานพร้อมๆ กันได้ เพราะมีเพียงแค่ 1 เทรดเท่านั้นที่จะสามารถทำงานในเวลาหนึ่งๆ

3. Many-to-many เป็นโมเดลที่ลดข้อจำกัดของโมเดล One-to-one และ Many-to-one ซึ่งทำให้หลายๆ เทรดสามารถทำงานได้พร้อมกันมากขึ้น

โมเดลทั้ง 3 โมเดลมีลักษณะดังภาพที่ 6



ภาพที่ 6 โมเดลของมัลติเทรด

ข้อดีของมัลติเทรดแสดงได้ดังต่อไปนี้

1. การใช้ทรัพยากรร่วมกัน ซึ่งในระบบมัลติเทรดนี้สามารถที่จะใช้ทรัพยากรต่างๆ ที่อยู่ในโปรเซส ซึ่งทำให้เทรดสามารถทำงานต่างกันในหน่วยความจำเดียวกันได้

2. เป็นประโยชน์ต่อสถาปัตยกรรมแบบมัลติโปรเซสเซอร์ (Multiprocessor) ซึ่งระบบมัลติเทรดจะช่วยเพิ่มประสิทธิภาพในการทำงานของมัลติโปรเซสเซอร์ ซึ่งเทรดแต่ละเทรดสามารถทำงานขนานกันไปที่แต่ละโปรเซสเซอร์

3. การตอบสนองการทำงาน โดยมัลติเทรดสามารถตอบสนองต่อความต้องการของผู้ใช้งานได้พร้อมกันหลายงาน เช่น การทำงานกับเว็บ (Web) ซึ่งสามารถดาวน์โหลด (Download) งานพร้อมกับสามารถตอบสนองต่อการทำงานของผู้ใช้ในลักษณะอื่นๆได้

4. ประหยัดเนื้อที่ในหน่วยความจำ เพราะเทรดจะใช้ทรัพยากรร่วมกันของโปรเซสที่เทรดนั้นอาศัยอยู่

Concurrency and Coordination Runtime (CCR)(Chrysanthakopoulos, 2007) เป็นรันไทม์ (Runtime) ที่ถูกออกแบบขึ้นมาสำหรับการพัฒนาโปรแกรมเกี่ยวกับหุ่นยนต์ (Robotic application) ซึ่ง CCR เป็นส่วนหนึ่งของ Microsoft Robotics Developer Studios (MRDS) โดย CCR มีกระบวนการจัดการกับเทรดและสามารถพัฒนาโปรแกรมแบบมัลติเทรดในลักษณะ Asynchronous (Chrysanthakopoulos and Singh, 2005) ได้ง่ายกว่าการพัฒนาโปรแกรมแบบมัลติเทรดในอดีต ซึ่งเป็นสิ่งที่สำคัญอย่างมากในการพัฒนาโปรแกรมเกี่ยวกับหุ่นยนต์ และยังสามารถนำมาใช้เป็นแบบอย่างสำหรับการพัฒนาโปรแกรมแบบมัลติเทรดได้ เนื่องจาก CCR มีความสามารถในการจัดการกับกลุ่มของเทรด (Thread pool) ได้อย่างมีประสิทธิภาพ นอกจากนั้น CCR (Johns and Taylor, 2008) ได้จัดเตรียมเฟรมเวิร์ค (Framework) สำหรับการสร้างกลุ่มของการติดต่อระหว่างเทรด ซึ่งการติดต่อสื่อสารระหว่างเทรดใช้วิธีการส่งข้อความไปยังพอร์ต (Port) โดยที่เทรดสามารถเขียนหรืออ่านข้อความจาก 1 พอร์ต หรือมากกว่า 1 พอร์ตได้ ซึ่งเฟรมเวิร์คนี้จะดูแลและจัดการกับพอร์ต, กลุ่มของเทรด และจัดการเกี่ยวกับ Dispatcher ซึ่ง Dispatcher มีหน้าที่เกี่ยวกับการสร้างเทรด, การเลือกเทรดไปใช้งาน, การนำงานเข้าสู่การประมวลผล และประสิทธิภาพการทำงานเกี่ยวกับ Load balance สำหรับกลุ่มงาน

CCR (Microsoft, 2009) ทำหน้าที่สำหรับการควบคุมและการจัดการเกี่ยวกับเทรดได้อย่างมีประสิทธิภาพ ซึ่งการจัดการ (Handler) เพื่อตอบสนองต่อข้อความที่ส่งเข้ามายังพอร์ต ซึ่งพอร์ตเหล่านี้จะถูกควบคุมดูแลโดยการจัดการพื้นฐานที่ประกอบด้วยดังต่อไปนี้

1. FormHandler เป็นกระบวนการจัดการเกี่ยวกับการทำงานของเทรด ซึ่งเทรดจะทำงานโดยไม่มีการอ่านค่าใดๆจากพอร์ต

2. Receive เป็นกระบวนการจัดการเกี่ยวกับการอ่าน 1 ข้อมูลจาก 1 พอร์ต

3. MultipleItemReceive เป็นการจัดการโดยการอ่านหลายๆข้อมูลตามจำนวนที่ได้กำหนดไว้จากพอร์ต ซึ่งข้อมูลเหล่านั้นจะต้องมีชนิดของข้อมูลเป็นชนิดเดียวกันทั้งหมด

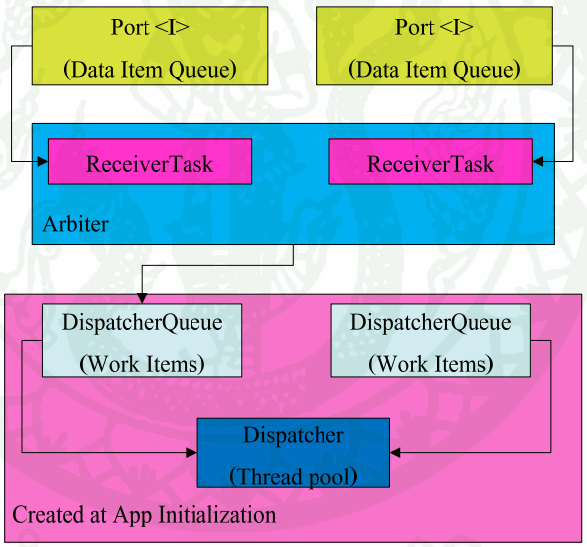
4. MultiplePortReceive เป็นการจัดการโดยการอ่าน 1 ข้อมูลจากหลายพอร์ต โดยที่ทุกๆพอร์ตจะต้องมีชนิดของข้อมูลเป็นชนิดเดียวกันทั้งหมด

5. JoinReceive เป็นการจัดการโดยการอ่านข้อมูลอย่างละ 1 ข้อมูลจากทั้ง 2 พอร์ต ซึ่งชนิดของข้อมูลทั้ง 2 พอร์ตนั้นสามารถมีชนิดที่แตกต่างกันได้

6. Choice เป็นการเลือกการทำงานอย่างใดอย่างหนึ่งเมื่ออ่านข้อมูลจาก 2 พอร์ต หรือมากกว่า 2 พอร์ต โดยเมื่อเลือกการทำงานอย่างใดอย่างหนึ่งแล้ว การทำงานอื่นที่เหลือจะละทิ้งไป

7. Interleave ประกอบด้วยกลุ่มของการทำงานของ 3 ชนิดที่ประกอบด้วย Concurrent, Exclusive หรือ Teardown จะถูกเรียกใช้ในกรณีที่จบการทำงาน ส่วน Concurrent จะสามารถทำงานพร้อมกันได้ แต่ Exclusive ไม่สามารถทำงานพร้อมกันได้

สถาปัตยกรรมของ CCR จะมีลักษณะดังภาพที่ 7



ภาพที่ 7 สถาปัตยกรรมของ CCR

โครงสร้างของ CCR (Richter, 2006) ประกอบด้วยคลาส (Class) พื้นฐานดังต่อไปนี้

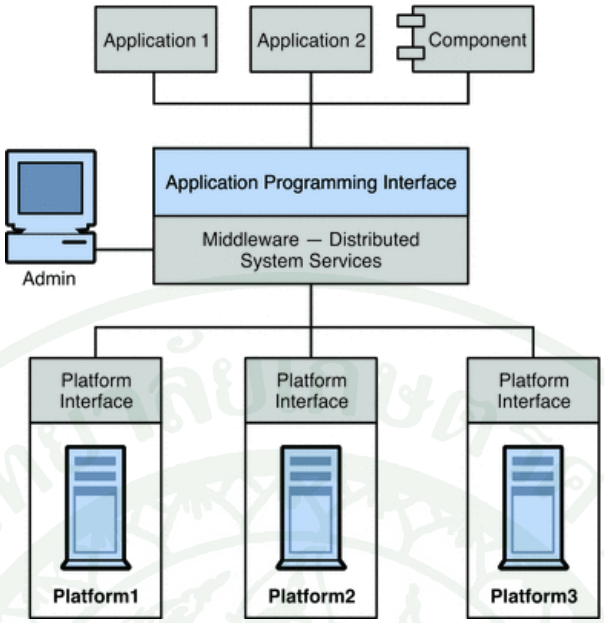
Dispatcher Class สำหรับการสร้างแอพพลิเคชั่น คลาสแรกที่ต้องสร้าง คือ Dispatcher ซึ่งเป็นออบเจ็ค (Object) โดยออบเจ็คนี้จะทำหน้าที่ในการจัดการกับกลุ่มเทรด ซึ่งเทรดเหล่านี้จะเรียกฟังก์ชันการทำงานเพื่อที่จะประมวลผลงานที่อยู่บนคิว (Queue) เมื่อสร้าง Dispatcher เรียบร้อยแล้ว เราสามารถระบุจำนวนเทรดตามความต้องการโดยค่าเริ่มต้นที่ Dispatcher กำหนด คือ สร้าง 1 เทรด สำหรับทุก 1 โปรเซสเซอร์บนเครื่องคอมพิวเตอร์ ซึ่งจำนวนของเทรดที่ถูกสร้างขึ้นโดย Dispatcher นั้นจะไม่มีการเปลี่ยนแปลงของจำนวนเทรด ซึ่งหมายถึงไม่มีการสร้างหรือทำลายเทรด ซึ่งแตกต่างจากกลุ่มของเทรดของ Common Language Runtime (CLR) โดย Dispatcher จะไม่มีเทรดพิเศษที่ใช้สำหรับตรวจสอบปริมาณงาน (Workload) และไม่มีเทรดที่ทำงานเกี่ยวกับการทำนายเพื่อเพิ่มหรือลดจำนวนของเทรดในกลุ่มเทรด ซึ่งลักษณะของกลุ่มเทรดที่ถูกสร้างโดย Dispatcher นี้จะช่วยเพิ่มประสิทธิภาพของการทำงานให้สูงขึ้น ขณะสร้าง Dispatcher เราสามารถกำหนดลำดับความสำคัญ (Priority) ของกลุ่มเทรดได้ โดยค่าลำดับความสำคัญเริ่มต้นของกลุ่มเทรดจะมีค่าเป็นลำดับความสำคัญแบบปกติ (Normal priority)โดยจำนวนกลุ่มเทรดของ CCR สามารถสร้างได้ตามความต้องการและสามารถกำหนดลำดับความสำคัญของแต่ละกลุ่มเทรดให้มีลำดับความสำคัญที่แตกต่างกันได้ตามต้องการเช่นกัน ซึ่งคุณสมบัติเหล่านี้ทำให้ CCR สามารถรองรับกับประเภทของงานที่มีความแตกต่างกันได้ ซึ่งต่างจากกลุ่มเทรดของ CLR ที่สามารถสร้างได้เพียง 1 กลุ่มเทรดเท่านั้น

DispatcherQueue Class เมื่อ Dispatcher ถูกสร้างเสร็จแล้วขั้นตอนต่อไปจะต้องสร้าง DispatcherQueue ซึ่ง DispatcherQueue เป็นคิวที่จัดเก็บฟังก์ชันพร้อมกับข้อมูล (ฟังก์ชันและข้อมูลรวมเรียกว่า งาน) ที่พร้อมประมวลผล ซึ่งมี Dispatcher คอยทำหน้าที่ดูแลการจัดการคิวนี้ ซึ่ง Dispatcher มีหน้าที่ในการเลือกงานที่อยู่ในคิวดังกล่าวและเลือกเทรดจากกลุ่มเทรด ซึ่งกลุ่มเทรดของ Dispatcher จะอยู่ในสถานะรอ (Waiting) จนกว่าจะมีงานปรากฏเข้ามาใน DispatcherQueue กรณีที่มีงานเข้ามายัง DispatcherQueue Dispatcher จะทำการเลือกเทรดจากกลุ่มเทรด และเทรดดังกล่าวจะเปลี่ยนสถานะจากการรอกลายเป็นสถานะพร้อมทำงานเพื่อรอรับงานไปประมวลผล และเมื่อเทรดดังกล่าวทำงานเสร็จเรียบร้อยแล้ว เทรดนั้นจะกลับเข้าสู่สถานะรอเช่นเดิม ความแตกต่างระหว่างกลุ่มเทรดของ CLR กับกลุ่มเทรดของ CCR คือ กลุ่มเทรดของ CLR กรณีที่มีงานเข้ามาอยู่ในคิว 500 งานจะไม่มีทางที่งานใหม่ที่เข้ามาในคิวจะได้รับการประมวลผล จนกว่า 500 งานแรกจะได้รับการประมวลผลจนเสร็จสิ้นทั้งหมด ซึ่งใน CCR สามารถที่จะใช้ DispatcherQueue ตัวหนึ่งสำหรับงานส่วนใหญ่และใช้อีก DispatcherQueue ตัวอื่นสำหรับงานที่มีลำดับความสำคัญสูงกว่า

ซึ่งทำให้งานใหม่ที่เข้ามาในคิวมีโอกาสที่จะได้รับการประมวลผล ก่อนโดยที่ไม่ต้องรออยู่ในคิวจนกระทั่ง 500 งานแรกได้รับการประมวลผลเสร็จ

Port and Arbiter Port ทำหน้าที่เสมือนคิวของข้อมูล ซึ่งสามารถที่จะรองรับชนิดของข้อมูลพื้นฐาน (รองรับข้อมูลพื้นฐานได้ 16 ชนิด) และสามารถใช้ในการเก็บผลลัพธ์จากการทำงาน ส่วน Arbiter เป็นคลาสที่ใช้ในการสร้างตัวรับงาน (ReceiverTask) เพื่อใช้ในการส่งงานไปยัง DispatcherQueue เพื่อให้กลุ่มเทรดของ Dispatcher นำงานเหล่านั้นไปประมวลผล โดยความสัมพันธ์ภายในสถาปัตยกรรมของ CCR มีลักษณะการทำงาน คือ เมื่อข้อมูลส่งไปที่พอร์ต และหลังจากการลงทะเบียนของตัวรับงานแล้ว และจะส่งข้อมูลไปยัง Arbiter ซึ่งเป็นตัวกำหนดว่าจะเลือกฟังก์ชันเพื่อที่จะประมวลผลด้วยข้อมูลเหล่านั้น ต่อจากนั้น Arbiter จะส่งข้อมูลและฟังก์ชันไปยัง DispatcherQueue แล้วกลุ่มเทรดของ Dispatcher จะทำการประมวลผลฟังก์ชันพร้อมกับข้อมูลที่ได้ระบุไว้

ระบบแบบกระจาย (Distributed System) (Tanenbaum and Steen, 2002) เป็นกลุ่มของคอมพิวเตอร์ที่มีความหลากหลาย คอมพิวเตอร์เหล่านี้จะมีความเป็นอิสระและกระจายอยู่ห่างกัน โดยการติดต่อสื่อสารระหว่างคอมพิวเตอร์ในระบบจะติดต่อกันผ่านทางอินเทอร์เฟส (Interface) ที่ใช้โปรโตคอล (Protocol) เดียวกัน ในระบบแบบกระจายขนาดของหน่วยประมวลผลบนเครื่องคอมพิวเตอร์จะมีความแตกต่างกัน เช่น Microprocessor, Workstation และ Minicomputer เป็นต้น โดยจะต้องมีซอฟต์แวร์(Software) ที่ช่วยให้ระบบแบบกระจายสามารถทำงานร่วมกันได้ซึ่งเรียกว่า มิดเดิ้ลแวร์ (Middleware) ซึ่งมิดเดิ้ลแวร์จะเป็นซอฟต์แวร์ที่ช่วยให้ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ที่มีความแตกต่างกันของแต่ละเครื่องคอมพิวเตอร์ในระบบแบบกระจายให้สามารถทำงานร่วมกันได้ ซึ่งตำแหน่งของมิดเดิ้ลแวร์ในระบบแบบกระจายแสดงดังภาพที่ 8



ภาพที่ 8 ตำแหน่งของมิดเดิ้ลแวร์บนระบบแบบกระจาย

ระบบแบบกระจายสร้างขึ้นเพื่อวัตถุประสงค์ดังต่อไปนี้

1. ช่วยในการติดต่อสื่อสาร เมื่อคอมพิวเตอร์จำนวนมากเชื่อมต่อกันโดยผ่านการติดต่อสื่อสารทางระบบเครือข่ายทำให้เสมือนกับเป็นเครื่องคอมพิวเตอร์เครื่องเดียว ซึ่งแต่ละเครื่องคอมพิวเตอร์สามารถที่จะแลกเปลี่ยนข้อมูลระหว่างกันได้ ทำให้ลดข้อจำกัดในการทำงานลงได้มากและสามารถรองรับกับการขยายตัวของระบบ

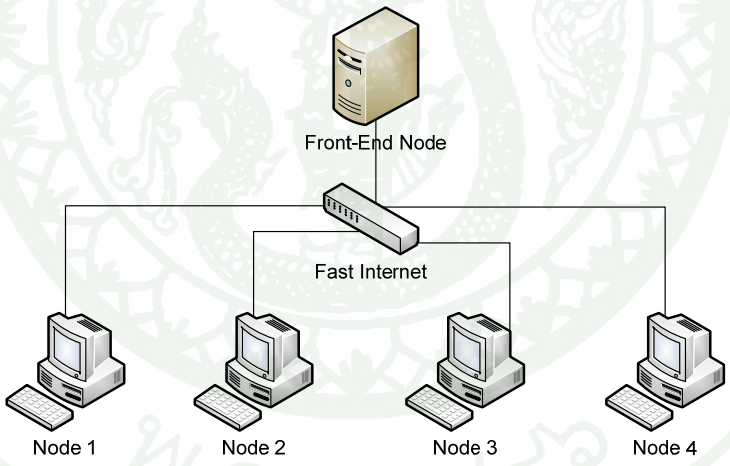
2. เพิ่มประสิทธิภาพในการประมวลผล ในระบบแบบกระจายสามารถที่จะกระจายงานให้กับแต่ละเครื่องคอมพิวเตอร์เพื่อช่วยกันทำงานไปพร้อมๆ กันได้

3. การใช้ทรัพยากรร่วมกัน ระบบสามารถที่จะให้เครื่องคอมพิวเตอร์หนึ่งสามารถใช้ทรัพยากรจากอีกเครื่องคอมพิวเตอร์หนึ่งได้ ซึ่งในระบบแบบกระจายมีกลไกสำหรับการใช้ทรัพยากรร่วมกันของเครื่องคอมพิวเตอร์ทั้งหมดทำให้เกิดกลุ่มของทรัพยากรขนาดใหญ่

4. เพิ่มความน่าเชื่อถือของระบบ การทำงานในระบบแบบกระจายถ้ามีเครื่องคอมพิวเตอร์เครื่องใดไม่สามารถทำงานได้ ซึ่งการหยุดการทำงานของเครื่องคอมพิวเตอร์ดังกล่าวจะไม่มีผลกระทบกับการทำงานของเครื่องคอมพิวเตอร์เครื่องอื่นๆ ในระบบ ซึ่งจะยังสามารถทำงานต่อไปได้ โดยใน

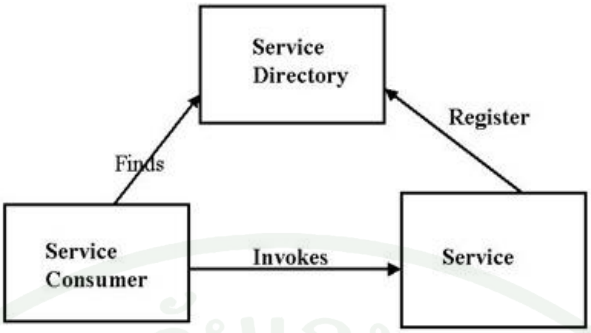
การตรวจสอบและแก้ไขกับเครื่องคอมพิวเตอร์ที่หยุดทำงานนั้น ระบบสามารถที่จะทำการแก้ไขและสามารถนำเครื่องคอมพิวเตอร์ดังกล่าวกลับมาทำงานร่วมกับระบบได้เหมือนเดิม

ระบบคลัสเตอร์ (Cluster) (Baker, 2000) คือ ระบบที่ประกอบด้วยเครื่องคอมพิวเตอร์หลายๆ เครื่องที่มีการเชื่อมต่อกันและสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ โดยแต่ละเครื่องจะมีการทำงานประสานกัน ทำให้สามารถเพิ่มประสิทธิภาพในการประมวลผล ซึ่งเทียบเท่ากับคอมพิวเตอร์สมรรถนะสูง แต่มีราคาและค่าใช้จ่ายที่แตกต่างกัน โดยโครงสร้างของระบบคลัสเตอร์จะประกอบด้วยกลุ่มของเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal computer), เครือข่ายความเร็วสูง และซอฟต์แวร์ที่ช่วยในการสร้างระบบคลัสเตอร์ ซึ่งลักษณะของระบบคลัสเตอร์คอมพิวเตอร์แสดงดังภาพที่ 9 โดยวัตถุประสงค์ของการนำระบบคลัสเตอร์คอมพิวเตอร์มาใช้งานก็เพื่อเร่งความเร็วของการทำงาน โดยมีการใช้เทคโนโลยีการประมวลผลแบบขนานหรือแบบกระจาย นอกจากนี้ยังทำให้ระบบมีความน่าเชื่อถือสูงขึ้น เนื่องจากระบบคลัสเตอร์คอมพิวเตอร์อาศัยการทำงานประสานกันของเครื่องคอมพิวเตอร์จำนวนมาก



ภาพที่ 9 ระบบคลัสเตอร์คอมพิวเตอร์

Service Oriented Architecture (SOA) (Erl, 2004) เป็นแนวคิดการสร้างระบบเสมือนขึ้นเพื่อใช้ในการคำนวณและเก็บข้อมูลซึ่งเสมือนว่าคอมพิวเตอร์หลายเครื่องรวมกันเป็นหนึ่งเดียวโดยผ่านการเชื่อมต่อ ซึ่ง SOA จะต้องมีการจัดการที่แตกต่างกันออกไปทางด้านระบบปฏิบัติการและทางด้านแอพพลิเคชั่นของเครื่องคอมพิวเตอร์ที่เชื่อมต่อกันนั้น โดยแนวคิดของ SOA มีลักษณะดังภาพที่ 10

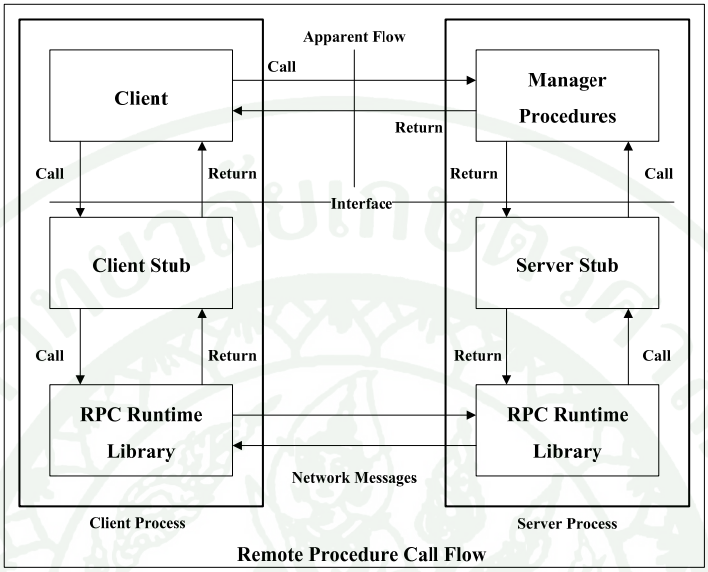


ภาพที่ 10 แนวคิดของ SOA

ส่วนประกอบของ SOA ประกอบด้วย 3 ส่วนต่อไปนี้ คือ ผู้ให้บริการ (Service), ผู้ขอบริการ (Service consumer) และตัวแทนของผู้ให้บริการ (Service directory) ซึ่งทั้งสามส่วนนี้จะมีการติดต่อกันโดยใช้ฟังก์ชันดังนี้ การประกาศบริการต่างๆ (Register), การค้นหา (Find) และการเรียกใช้ (Invoke) ฟังก์ชันเหล่านี้จะทำงานโดยเริ่มจากผู้ให้บริการจะทำการประกาศบริการต่างๆ ไว้บนฝั่งตัวแทนของผู้ให้บริการซึ่งทางฝั่งของผู้ขอบริการร้องขอใช้บริการต่างๆ ได้โดยการไปค้นหาบริการต่างๆ ที่ฝั่งตัวแทนผู้ให้บริการและเมื่อพบบริการที่ต้องการก็จะเรียกใช้ไปยังฝั่งผู้ให้บริการ

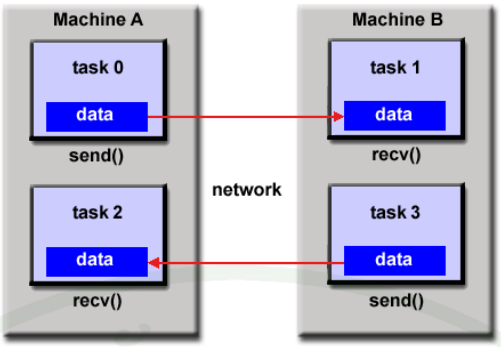
Remote Procedure Call (RPC) (Silberschatz, 2006) เป็นแนวคิดที่อนุญาตให้โปรแกรมที่อยู่บนเครื่องหนึ่งสามารถที่จะเรียกใช้การทำงาน (Procedure หรือ Subroutine จะมีลักษณะการทำงานคล้ายกับฟังก์ชัน) ที่อยู่บนอีกเครื่องหนึ่งได้ ซึ่ง RPC เป็นพื้นฐานในการพัฒนาโปรแกรมประยุกต์แบบไคลเอนต์-เซิร์ฟเวอร์ (Client/Server) ซึ่งทำงานผ่านระบบเครือข่าย โดยฝั่งไคลเอนต์จะเป็นฝั่งที่เรียกใช้งาน และฝั่งเซิร์ฟเวอร์ซึ่งเป็นฝั่งทำงาน โดยการเรียกใช้การทำงานของทางฝั่งผู้เรียกใช้นั้นจะมีลักษณะเหมือนกับการเรียกใช้ฟังก์ชัน ซึ่งการที่จะเรียกใช้การทำงานของฝั่งผู้ให้บริการที่อยู่ไกลออกไป ทางฝั่งผู้เรียกใช้งานจะต้องมีสิ่งที่ใช้ในการเชื่อมต่อกับฝั่งผู้ให้บริการที่เรียกว่า ไคลเอนต์ Stub และทางฝั่งผู้ให้บริการจะต้องมีสิ่งที่ใช้ในการเชื่อมต่อกับฝั่งผู้เรียกใช้ที่เรียกว่า เซิร์ฟเวอร์ Stub ซึ่งทางฝั่งผู้เรียกใช้จะติดต่อกับฝั่งผู้ให้บริการผ่านทางไคลเอนต์ Stub และในทางตรงกันข้ามฝั่งผู้ให้บริการก็จะติดต่อกับฝั่งผู้เรียกใช้ทางเซิร์ฟเวอร์ Stub โดยทางฝั่งผู้เรียกใช้จะส่งข้อมูลตามจำนวนพารามิเตอร์ (Parameter) ที่ตรงกับการทำงานทางฝั่งผู้ให้บริการซึ่งทางผู้เรียกใช้จะรอข้อมูลตอบกลับของทางฝั่งผู้ให้บริการ ต่อจากนั้นทางด้านผู้ให้บริการเมื่อได้รับข้อมูลของผู้เรียกใช้ก็จะทำการดึงข้อมูลนั้นมาทำการคำนวณแล้วส่งผลลัพธ์ตอบกลับไปยังฝั่งผู้เรียกใช้ และเมื่อทางฝั่งผู้เรียกใช้ได้รับ

ผลลัพธ์จากการคำนวณของ Procedure ของทางฝั่งผู้ให้บริการ ฝั่งผู้เรียกใช้ก็จะกลับเข้าสู่การทำงานของฝั่งตนเองต่อไป ซึ่งลักษณะของการทำงานตามแนวคิด RPC สามารถแสดงดังภาพที่ 11



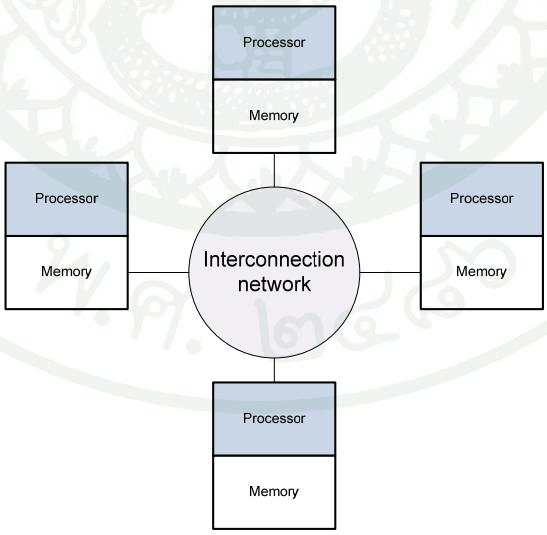
ภาพที่ 11 การทำงานของ RPC

Message Passing (Barney, 2007) การพัฒนาโปรแกรมแบบขนานตามแนวคิดของ Message passing นั้นจะมองว่าระบบคอมพิวเตอร์เป็นระบบแบบขนานที่มีหน่วยความจำแยกจากกัน (Distributed memory) ซึ่งกลุ่มของงานจะถูกเก็บไว้ที่หน่วยความจำของแต่ละหน่วยประมวลผล โดยการส่งผ่านข้อมูลระหว่างกันจะสามารถทำได้โดยการส่งข้อความ (Message) ผ่านไปบนระบบเครือข่าย และจะต้องมีการระบุรายละเอียดในการรับ/ส่ง เพื่อความถูกต้องของผู้รับและผู้ส่ง ซึ่งแสดงดังภาพที่ 12 โดยรูปแบบในการพัฒนาโปรแกรมแบบขนานตามแนวคิดของ Message passing โดยส่วนใหญ่จะนิยมรูปแบบของ Single Program Multiple Data (SPMD) ซึ่งจะมีเพียงหนึ่งโปรแกรมที่ทำงานบนแต่ละหน่วยประมวลผลแต่รับข้อมูลต่างกัน ซึ่ง SPMD นี้อยู่ในกลุ่มของสถาปัตยกรรมแบบ Multiple Instruction Multiple Data (MIMD) โดยสถาปัตยกรรมแบบ MIMD นั้น แต่ละหน่วยประมวลผลจะทำงานเป็นอิสระต่อกัน



ภาพที่ 12 รูปแบบการส่งผ่านข้อมูล

Message Passing Interface (MPI) (Wilkinson and Allen, 1999) เป็นไลบรารี (Library) ที่รองรับการพัฒนาแอพพลิเคชั่นแบบขนาน โดยใช้การสื่อสารสำหรับการแลกเปลี่ยนข้อมูลระหว่างโปรเซสที่ทำงานร่วมกันโดยแต่ละโปรเซสประกอบด้วยหน่วยความจำของตัวเอง ซึ่งโปรเซสสามารถเข้าถึงข้อมูลได้โดยตรง ถ้าหากต้องการข้อมูลที่อยู่ต่างโปรเซส จะต้องมีการสื่อสารระหว่างโปรเซส ซึ่งการสื่อสารระหว่างโปรเซสจะต้องประกอบด้วย ผู้ส่ง, ผู้รับ และข้อมูลจะต้องมีความถูกต้องโดย MPI จะมี Application Programming Interface (API) ที่ใช้รองรับการทำงานสำหรับการแลกเปลี่ยนข้อมูลระหว่างโปรเซสให้กับผู้ใช้ ซึ่งโมเดลของ Message passing มีลักษณะดังภาพที่ 13



ภาพที่ 13 โมเดลของ Message passing

Decentralized Software Services (DSS) (Microsoft, 2009) ถูกพัฒนาอยู่บน CCR โดย DSS ใช้สำหรับการสร้างแอพพลิเคชั่นภายใต้แนวคิดของการเชื่อมต่อระหว่างแอพพลิเคชั่นแบบหลวม (Loosely coupled service model) โดยแต่ละแอพพลิเคชั่นสามารถติดต่อสื่อสารกันด้วยวิธีการส่งข้อความระหว่างแอพพลิเคชั่น ซึ่ง DSS ได้จัดเตรียมโครงสร้างพื้นฐานสำหรับการติดต่อสื่อสารระหว่างแอพพลิเคชั่นไว้แล้ว ทำให้แอพพลิเคชั่นสามารถติดต่อสื่อสารและแลกเปลี่ยนข้อมูลระหว่างกันด้วยวิธีการส่งข้อความบนระบบเครื่อข่าย โดย DSS (Johns and Taylor, 2008) มีหน้าที่ในการควบคุมการทำงานของแอพพลิเคชั่นและรับผิดชอบการเริ่มและการหยุดการทำงานของเซอร์วิส พร้อมทั้งมีหน้าที่ในการจัดการกับการไหล (Flow) ของข้อความที่ส่งผ่านระหว่างเซอร์วิสด้วยวิธี Service forwarder ports โดย DSS ใช้โปรโตคอล Decentralized Software Services Protocol (DSSP) (Nielsen and Chrysanthakopoulos, 2007) และ Hypertext Transfer Protocol (HTTP) เป็นพื้นฐานสำหรับการติดต่อสื่อสารระหว่างเซอร์วิส ซึ่ง DSSP ถูกสร้างขึ้นบนพื้นฐานของโมเดล Representational State Transfer (REST) และ DSSP ยังมีลักษณะคล้ายกับ (Simple Object Access Protocol) SOAP ซึ่งเป็นโปรโตคอลสำหรับการสื่อสารและแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์บนระบบเครื่อข่าย (ใช้กับเว็บเซอร์วิส (Web services)) แต่ DSSP มีความแตกต่างจาก SOAP ตรงที่ DSSP แยกสถานะของเซอร์วิส (State) ออกจากพฤติกรรมการทำงาน (Behavior) และสามารถเข้าถึงเซอร์วิสได้โดยการดำเนินการ (Operations) ที่อยู่บนสถานะของเซอร์วิสได้ ซึ่งแตกต่างจากเว็บเซอร์วิสที่สถานะของเซอร์วิสจะถูกซ่อนไว้ วัตถุประสงค์ของการออกแบบ DSS ก็เพื่อให้ง่ายสำหรับการพัฒนาแอพพลิเคชั่น พร้อมทั้งช่วยเพิ่มประสิทธิภาพในการประมวลผลให้สูงขึ้นทั้งในรูปแบบของ Message passing และMultithreading

องค์ประกอบของเซอร์วิสที่ถูกสร้างบน DSS มีส่วนประกอบดังต่อไปนี้

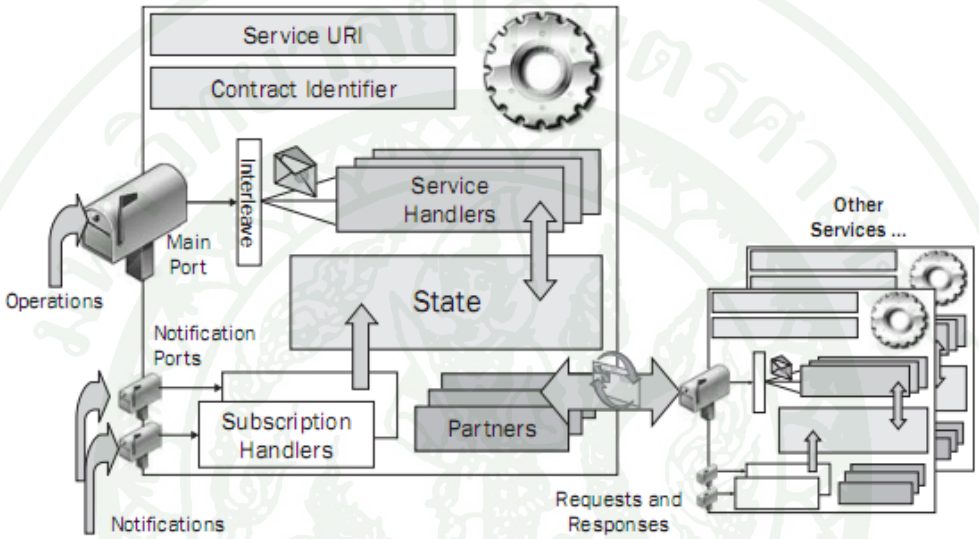
1. Contract เป็นส่วนที่ใช้ในการกำหนดข้อความที่ต้องการส่งไปยังเซอร์วิสต่างๆ ซึ่งเรียกว่า Contract identifier

2. Internal state เป็นข้อมูลที่เซอร์วิสนำไปใช้สำหรับการควบคุมการดำเนินงานของเซอร์วิสนั้น

3. Behaviors เป็นกลุ่มการดำเนินงานของเซอร์วิสมีหน้าที่เกี่ยวกับการปรับสถานะของเซอร์วิส และการจัดการเกี่ยวกับการรับ/ส่งข้อมูลจากเซอร์วิสอื่น

4. Execution context เป็นกลุ่มของเซอร์วิสต่างๆ ที่มีความเกี่ยวโยงกัน(Partnership) ซึ่งเซอร์วิสสามารถมีกลุ่มของเซอร์วิสได้ ซึ่งการสร้างกลุ่มของเซอร์วิสนั้นจะสามารถสร้างได้ในขั้นตอนของการเริ่มสร้างเซอร์วิส

ส่วนประกอบของเซอร์วิสบน DSS แสดงลักษณะดังภาพที่ 14



ภาพที่ 14 ส่วนประกอบของเซอร์วิสบน DSS

งานวิจัยที่เกี่ยวกับการทดสอบประสิทธิภาพการทำงานของ CCR และ DSS มีดังต่อไปนี้

การวิเคราะห์ประสิทธิภาพของ CCR และ DSS สำหรับการประมวลผลแบบขนาน ในงานวิจัยของ Qiu *et al.* (2007)ได้ทำการทดสอบประสิทธิภาพของ CCR และ DSS โดยการทดสอบประสิทธิภาพของ CCR นั้นจะเป็นการทดสอบการส่งข้อมูลเพื่อจับเวลาของการรับและการส่งข้อมูล ซึ่งลักษณะของการรับ/ส่งข้อมูลเป็นแบบ MPI ภายใต้ CCR โดยการทดสอบการรับ/ส่งข้อมูลจะใช้คำสั่งเช่น MPIBroadcast, MPISend และ MPIRecv เป็นต้น (CCR มีชุดคำสั่งในการรับ/ส่งข้อมูล เช่น Interleave ซึ่งเป็นคำสั่งที่มีลักษณะการทำงานเหมือนกับ MPIBroadcast เป็นต้น) ซึ่งเป็นคำสั่งตามมาตรฐานของ MPI แล้วเปรียบเทียบและแสดงผลของการทดสอบซึ่งแสดงให้เห็นถึงประสิทธิภาพของ CCR ในการทำงานบนเครื่องมัลติคอร์ และการทดสอบประสิทธิภาพของ DSS ได้ทดสอบโดยการส่งข้อความระหว่างคอร์ โดยเป็นการส่งการร้องขอและส่งผลตอบกลับ ซึ่งผลการทดสอบแสดงให้เห็นว่าเมื่อจำนวนของข้อความเพิ่มมากขึ้นเวลาที่ใช้จะลดลง โดยเฉพาะเมื่อข้อความมีจำนวนระหว่าง

50 ถึง 1,000 ข้อความ เวลาที่ใช้มีค่าเท่ากับ 40 ถึง 50 ไมโครวินาที หรือคิดเป็น Throughput จะมีค่าถึง 20,000 ถึง 25,000 ของการรับและส่งข้อความต่อวินาที

งานวิจัยของ Qiu *et al.* (2008)ได้ทำการทดสอบประสิทธิภาพของ CCR บนเครื่องมัลติคอร์ที่มีจำนวนคอร์แตกต่างกัน โดยวัดประสิทธิภาพของการแก้ปัญหาและเปรียบเทียบ Overhead ของการรับ/ส่งข้อมูลกับ MPICH2, MPJE, Nemesis และ mpiJava โดยปัญหาที่นำมาใช้ในการทดสอบประสิทธิภาพเป็นปัญหาทางด้าน Data minning ซึ่งผลการทดสอบแสดงให้เห็นว่าประสิทธิภาพการประมวลผลของ CCR สามารถให้ค่า Speedup ได้ถึง 7.5 เท่าบนเครื่องคอมพิวเตอร์ 8 คอร์ และให้ค่าของ Efficiency ได้ถึง 95 เปอร์เซ็นต์ นอกจากนั้นได้ทำการตรวจสอบ Overhead ที่เกิดจากหน่วยความจำ, ความเปลี่ยนแปลงของรันไทม์,และการประสานการทำงาน ซึ่งพบว่าเวลาที่เกิดจาก Overhead เหล่านี้มีผลกระทบต่อเวลาของการทำงานไม่มากนัก

งานวิจัยที่เกี่ยวข้อง

Pickup and Delivery Problem with Time Windows (PDPTW) (Ropke, 2005) เป็นปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งในทางเทคนิคเรียกว่าเป็นปัญหาของการให้บริการของการขนส่งสินค้าจากการร้องขอจากจุดรับ/ส่งโดยใช้ยานพาหนะที่มีอยู่อย่างจำกัด ซึ่งแต่ละการร้องขอจะมีความเกี่ยวข้องกับการเคลื่อนย้ายปริมาณของสินค้าจากจุดรับสินค้าไปส่งยังจุดส่งสินค้าโดยลักษณะของจุดรับ/ส่งสินค้า แบ่งออกเป็น 4 ลักษณะคือ 1) ยานพาหนะรับสินค้าจากจุดหนึ่งแล้วไปส่งอีกจุดหนึ่ง 2) รับสินค้าจากจุดหนึ่งแล้วไปส่งหลายจุด 3) รับสินค้าหลายจุดแล้วไปส่งหนึ่งจุด และ 4) จุดหนึ่งจุดสามารถเป็นได้ทั้งจุดรับและจุดส่งสินค้า ซึ่งทั้ง 4 ลักษณะจะมีช่วงเวลาในการรับและส่งสินค้ากำกับ โดยการเคลื่อนย้ายสินค้ามีช่วงเวลาของการเปิดรับสินค้าและเวลาการส่งสินค้า เป็นเงื่อนไข ซึ่งสำหรับการรับสินค้านั้น ยานพาหนะจะต้องเดินทางไปรับสินค้าภายในช่วงเวลาที่กำหนดหากเดินทางไปก่อนเวลาเปิดรับสินค้า ยานพาหนะต้องรอจนกว่าจะถึงเวลารับสินค้าและเมื่อรับสินค้าแล้วยานพาหนะจะต้องส่งสินค้าไปยังจุดส่งสินค้าภายในช่วงเวลาที่จุดส่งสินค้ากำหนดไว้ โดยวัตถุประสงค์สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง มีดังต่อไปนี้ 1) ทุกการร้องขอจะต้องได้รับการให้บริการทั้งหมด 2) ใช้จำนวนยานพาหนะสำหรับการให้บริการน้อยที่สุด 3) ลดค่าใช้จ่าย (Cost) ของการเดินทาง ตัวอย่างเช่น เดินทางโดยใช้ระยะทางที่สั้นที่สุด เป็นต้น ซึ่งวัตถุประสงค์เหล่านี้เป็นสิ่งที่บ่งบอกถึงต้นทุนของการดำเนินการทั้งหมด โดยผลลัพธ์ที่ได้รับจากการแก้ปัญหาการจัดเส้นทาง

ยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง คือ เส้นทางการเดินรถของการให้บริการการร้องขอของยานพาหนะแต่ละคัน โดยผลลัพธ์ที่มีประสิทธิภาพที่ดีจะช่วยลดต้นทุนของการให้บริการการขนส่งได้เป็นอย่างมากและช่วยเพิ่มผลกำไรให้แก่บริษัท

ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งนั้น เป็นปัญหาการตัดสินใจที่มีความซับซ้อน และเป็นปัญหาที่มีขนาดใหญ่ จำเป็นต้องใช้เวลาในการค้นหาผลลัพธ์ที่ดีที่สุดหรือผลลัพธ์ที่ยอมรับได้ ซึ่งลักษณะของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สามารถแสดงดังภาพที่15

ภาพที่ 15 ลักษณะตัวอย่างของปัญหา PDPTW

งานวิจัยนี้ได้ออกแบบและพัฒนาระบบสำหรับสร้างมัลติคอร์คลัสเตอร์ เพื่อรองรับการคำนวณแบบขนานทั้งแบบ Message passing และ Multithreading สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งมีงานวิจัยอื่นที่เกี่ยวข้องกับงานวิจัยดังต่อไปนี้

การแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เป็นปัญหาที่เกิดจากสถานการณ์จริงซึ่งเป็นการจัดส่งสินค้าจากที่หนึ่งไปยังอีกที่หนึ่งด้วยเงื่อนไขของเวลาของการจัดส่งสินค้า ซึ่งผลลัพธ์ของการแก้ปัญหาดังกล่าวจะสามารถประหยัดต้นทุนของการขนส่งได้เป็นอย่างมาก ดังนั้นจึงมีงานวิจัยหลายชิ้นได้เสนอวิธีการในรูปแบบต่างๆ เพื่อนำมาใช้ในการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เพื่อหาผลลัพธ์ที่ดีที่สุดหรือเป็นที่ยอมรับ งานวิจัยที่เกี่ยวกับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง มีดังนี้

Li and Lim (2001) เสนอวิธีการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งเรียกวิธีนี้ว่า Tabu-embedded simulated annealing วัตถุประสงค์ของวิธีนี้คือ การให้บริการแก่ลูกค้า (Customer) ครบทั้งหมด และการให้บริการแก่ลูกค้าโดยใช้จำนวนของยานพาหนะน้อยที่สุด ซึ่งผลการทดสอบกับปัญหาที่มีขนาด 100, 200 และ 600 ลูกค้า แสดงให้เห็นว่าผลลัพธ์ที่ได้จากการแก้ปัญหาเป็นผลลัพธ์ที่ดีที่สุดและบางผลลัพธ์เป็นผลลัพธ์ที่ดีในระดับหนึ่ง

Bent and Hentenryck (2006) ได้เสนอวิธีการแก้ปัญหาดังกล่าว โดยแบ่งวิธีการแก้ปัญหาเป็น 2 ขั้นตอน คือ ขั้นแรกจะใช้วิธี Simulated annealing (SA) เพื่อการลดจำนวนของเส้นทางหรือลดจำนวนยานพาหนะ และในขั้นที่สองจะใช้วิธี Large neighborhood search (LNS) เพื่อลดค่าใช้จ่ายของการเดินทาง โดยผลการทดสอบกับชุดปัญหาเดียวกับของ Li และ Lim แสดงให้เห็นว่าวิธีการนี้สามารถให้ผลลัพธ์ที่มีค่าดีกว่าผลลัพธ์ของ Li และ Lim สำหรับบางปัญหา และบางผลลัพธ์มีค่าใกล้เคียงกับผลลัพธ์ที่ดีที่สุดและสามารถลดค่าใช้จ่ายของการเดินทางได้

Rokpe and Pisinger (2006)ได้เสนอวิธีการแก้ปัญหาที่มีลักษณะคล้ายกับงานวิจัยของ Bent และ Hentenryck ซึ่งใช้การปรับปรุงวิธีการของ LNS ซึ่งเรียกว่า Adaptive large neighborhood search (ALNS) โดยวิธีการของทั้งสองคนได้เลือกจุดรับ/ส่งสินค้าออกจากยานพาหนะ และนำจุดรับ/ส่งสินค้ากลับเข้าสู่ยานพาหนะในวิธีที่แตกต่างกัน ซึ่งกระบวนการทำงานจะพยายามลดจำนวน

ของยานพาหนะลงทีละคัน หลังจากนั้นทำการเปรียบเทียบผลลัพธ์ที่ได้ใหม่กับผลลัพธ์เดิมและตรวจสอบเงื่อนไขการยอมรับผลลัพธ์ เมื่อทดสอบกับชุดปัญหาเดียวกันกับของ Li และ Lim ผลปรากฏว่า วิธีการนี้ให้ผลลัพธ์ที่ดีกว่าวิธีการของ Bent และ Hentenryck และผลลัพธ์ที่ได้มีค่าใกล้เคียงกับผลลัพธ์ที่ดีที่สุดเกือบทุกปัญหา

Tabu search (Tabu search algorithm, 1993) เป็นวิธีการหาผลลัพธ์ของปัญหาการตัดสินใจ โดย Tabu search เป็นวิธีการเรียนรู้จากประสบการณ์ในจำนวนของรอบการทำงานซ้ำที่ผ่านมาโดยการใช้หน่วยความจำของคอมพิวเตอร์เข้ามาช่วย ซึ่งวิธีการนี้จะป้องกันไม่ให้เกิดผลลัพธ์ที่ดีที่สุดเฉพาะกลุ่ม (Local optima) โดยแนวคิดของวิธี Tabu นั้น จะใช้หน่วยความจำของคอมพิวเตอร์ ซึ่งคอมพิวเตอร์จะเรียนรู้จากการหาผลลัพธ์ของรอบการทำงานที่ผ่านมาในการบอกบริเวณพื้นที่ในการหาคำตอบที่อาจจะให้ผลลัพธ์ที่ดีที่สุดหรือเป็นผลลัพธ์ที่ดีที่สุดของรอบการทำงานถัดไป

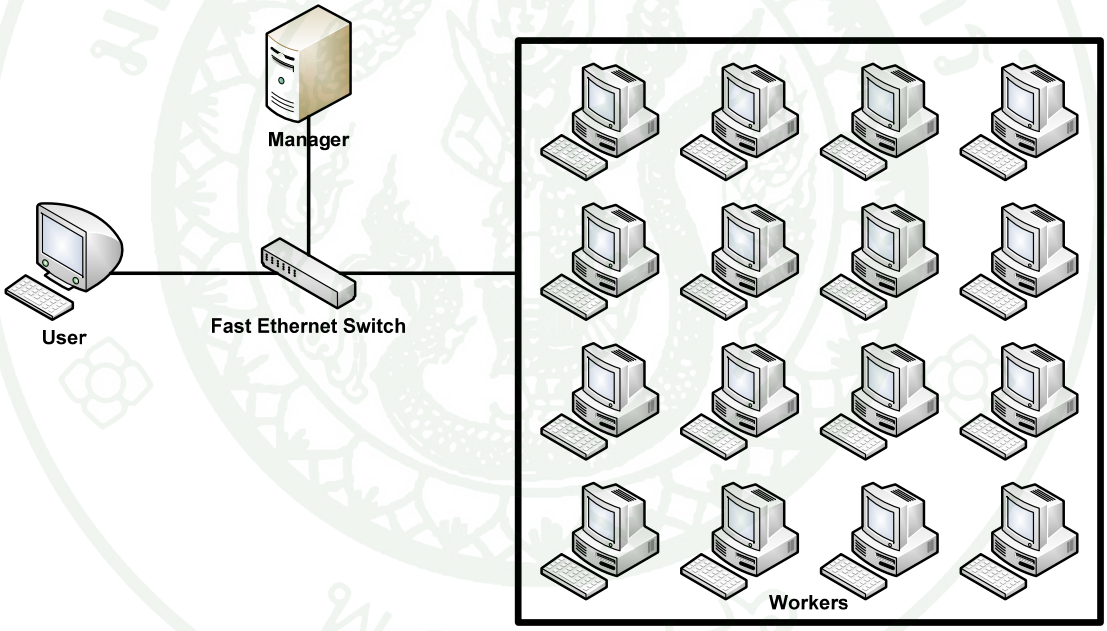
Paradiseo (Paradiseo, 2009) เป็นเฟรมเวิร์คที่รองรับกับการแก้ปัญหาการตัดสินใจ ซึ่ง Paradiseo ได้จัดเตรียมไลบรารีสำหรับการพัฒนาซอฟต์แวร์แบบขนาน สำหรับวิธีการที่เรียกว่า Metaheuristics (Alba, 2005) โดยไลบรารีที่จัดเตรียมไว้ให้จะเกี่ยวกับวิธี Genetic algorithm, Local search, Simulated annealing, Iterated local search และ Tabu search เป็นต้น ซึ่งไลบรารีเหล่านี้ได้ถูกพัฒนาโดยใช้ไลบรารีพื้นฐานจาก MPI, PVM และ PThreads ซึ่งทำให้ซอฟต์แวร์ที่พัฒนาโดย Paradiseo สามารถทำงานบนเครื่องมัลติคอร์, ระบบคลัสเตอร์ และระบบกริด แต่เนื่องจากในการพัฒนาซอฟต์แวร์เพื่อแก้ปัญหาการตัดสินใจโดยใช้ Paradiseo นั้น ผู้พัฒนาจำเป็นต้องมีความรู้ความเข้าใจในเรื่องของปัญหาการตัดสินใจ, แนวคิดของการประมวลผลแบบขนานและทักษะในการเขียนซอฟต์แวร์ ดังนั้น ในการพัฒนาซอฟต์แวร์โดยใช้ Paradiseo จึงต้องใช้ระยะเวลาของการพัฒนาซอฟต์แวร์มาก

อุปกรณ์และวิธีการ

อุปกรณ์

ฮาร์ดแวร์ (Hardware)

งานวิจัยนี้ใช้เครื่องคอมพิวเตอร์ของทางศูนย์ไทยกริดแห่งชาติ (Thai National Grid Center) ซึ่งแต่ละเครื่องจะประกอบด้วย 4 โปรเซสเซอร์ สำหรับการพัฒนาและทดสอบระบบโดยโครงสร้างการเชื่อมต่อระหว่างส่วนประกอบต่างๆ และรายละเอียดทางด้านฮาร์ดแวร์ของแต่ละส่วนประกอบในระบบ แสดงดังภาพที่ 16 และตารางที่ 1



ภาพที่ 16 โครงสร้างของการเชื่อมต่อระหว่างส่วนประกอบต่างๆ ในระบบ

ตารางที่ 1 รายละเอียดทางด้านฮาร์ดแวร์ของแต่ละส่วนประกอบในระบบ

Machine	Hardware	Operating System
1 Manager	Intel(R) Xeon(TM) 3.2 GHz, 8 GB of RAM	Windows HPC Server 2003
16 Workers	Intel(R) Xeon(TM) 3.2 GHz, 8 GB of RAM	Windows HPC Server 2003
1 User	Intel(R) Xeon(TM) 3.2 GHz, 8 GB of RAM	Windows HPC Server 2003

ตารางที่ 2 รายละเอียดทางด้านซอฟต์แวร์ที่ใช้ในระบบ

No.	Software
1	Microsoft Windows HPC Server 2003
2	Microsoft Robotics Developer Studio 2008
3	Microsoft Office 2007
4	Microsoft Visual Studio 2008
5	Microsoft .NET Framework SDK v3.0
6	MySQL Server v5.1
7	MySQL Connector v5.1.7

วิธีการ

การออกแบบและพัฒนากรอบการทำงานสมรรถนะสูง เพื่อรองรับการคำนวณแบบขนานสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของงานวิจัยนี้มีการออกแบบและพัฒนาระบบแบ่งออกเป็น 2 ส่วนดังต่อไปนี้

1. การออกแบบและพัฒนากรอบการทำงานสมรรถนะสูงเพื่อรองรับการคำนวณแบบขนานสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เป็นส่วนการดำเนินการที่อธิบายถึงการออกแบบโครงสร้างและสถาปัตยกรรมของระบบมัลติคอร์คลัสเตอร์คอมพิวเตอร์ที่รองรับการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ว่ามีส่วนประกอบอะไรบ้าง, มีการเชื่อมต่อกันอย่างไร, ลักษณะการทำงานโดยรวมของระบบ, แต่ละส่วนประกอบเป็นอย่างไร และแสดง

โครงสร้างโดยรวมของระบบการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง นอกจากนั้นยังได้กล่าวถึงโมเดลที่นำมาใช้สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งมี 2 แบบ คือ Solve and improve algorithm (SIA) และ Tabu search algorithm (TSA)

2. การแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งด้วยโมเดล SIA และ TSA บนแนวคิดของการคำนวณแบบขนานโดยใช้กรอบการทำงานแบบสมรรถนะสูง เป็นส่วนการดำเนินการที่กล่าวถึงแนวคิดที่ทำให้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สามารถเกิดการคำนวณแบบขนานได้บนโมเดลของ SIA และ TSA และกล่าวถึงการออกแบบรูปแบบการใช้งานของการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งให้กับผู้ใช้

การออกแบบและพัฒนากรอบการทำงานสมรรถนะสูงเพื่อรองรับการคำนวณแบบขนานสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

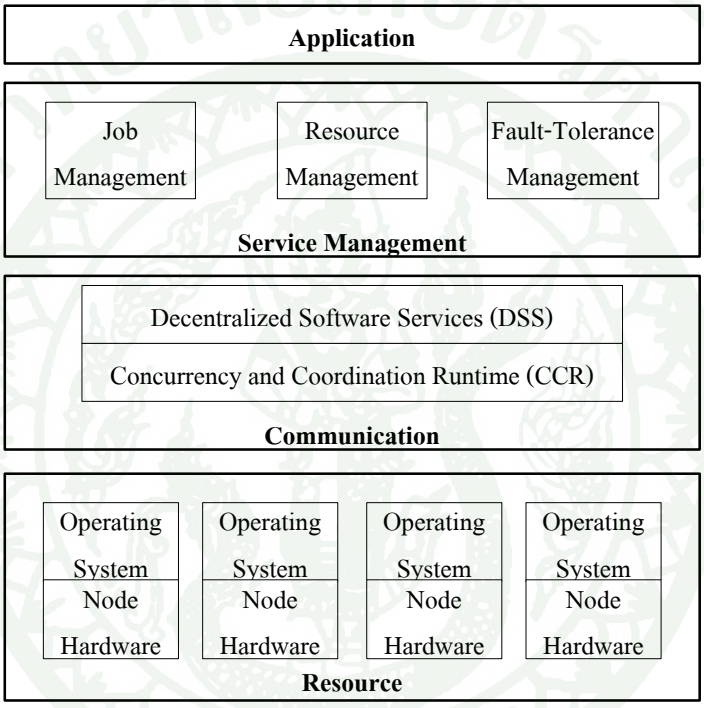
งานวิจัยนี้ทำการออกแบบกรอบการทำงานสมรรถนะสูงที่รองรับการคำนวณแบบขนานที่ผสมผสานระหว่างการทำงานแบบ Message passing และการทำงานแบบ Multithreading (Hybrid) สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง มีชื่อเรียกว่า Distributed Pickup and Delivery Problem with Time Windows Infrastructure (DPDPTWI) โดย DPDPTWI มีการออกแบบและพัฒนาระบบในแต่ละส่วนโดยใช้เทคโนโลยี Concurrency and Coordination Runtime และ Decentralized Software Services (CCR/DSS) เป็นพื้นฐาน โดย CCR ใช้สำหรับการพัฒนาการคำนวณแบบขนานแบบ Multithreading และ DSS ใช้สำหรับการพัฒนาการคำนวณแบบขนานแบบ Message passing

สถาปัตยกรรมของ DPDPTWI

สถาปัตยกรรมของ DPDPTWI มี 4 เลเยอร์ (Layer) ซึ่งมีลักษณะดังภาพที่ 17 โดยประกอบด้วย Resource Layer, Communication Layer, Service Management Layer และApplication Layer ซึ่งรายละเอียดของแต่ละเลเยอร์ มีดังต่อไปนี้

1. Resource Layer เป็นชั้นของทรัพยากรต่างๆ บนระบบเครือข่าย ซึ่งประกอบด้วยเครื่องมัลติคอร์คอมพิวเตอร์และระบบปฏิบัติการวินโดวส์

2. Communication Layer เป็นชั้นการสื่อสารของส่วนประกอบต่างๆ บน DPDPTWI ประกอบด้วย CCR ใช้สำหรับการคำนวณแบบมัลติเทรดบนเครื่องมัลติคอร์คอมพิวเตอร์ และ DSS ใช้สำหรับในการแลกเปลี่ยนข้อมูลระหว่างเครื่องมัลติคอร์คอมพิวเตอร์ในระบบ



ภาพที่ 17 สถาปัตยกรรมของ DPDPTWI

3. Service Management Layer เป็นชั้นเซอร์วิสที่ใช้ในการสร้าง DPDPTWI โดยมีเซอร์วิสที่ทำให้แอพพลิเคชั่นสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สามารถทำการคำนวณแบบขนานบน DPDPTWI ซึ่งประกอบด้วย

3.1 Resource Management เป็นเซอร์วิสที่ใช้รวบรวมทรัพยากรที่อยู่บนเครือข่าย โดยทำหน้าที่เก็บรายละเอียดและสถานะของทรัพยากรบนระบบ

3.2 Job Management เป็นเซอร์วิสที่ใช้รองรับการส่งงานจากผู้ใช้, การจัดลำดับงาน และการควบคุมงานที่ได้รับจากผู้ใช้ให้กระจายไปทำงานยังทรัพยากรบนระบบ

3.3 Fault-Tolerance Management เป็นเซอร์วิสที่ใช้ตรวจสอบความผิดพลาดของทรัพยากรต่างๆ บนระบบเพื่อให้ระบบสามารถดำเนินงานได้ ถึงแม้จะมีทรัพยากรบางส่วนเกิดความเสียหาย (เครื่องคอมพิวเตอร์หยุดการประมวลผล)

4. Application Layer เป็นชั้นของแอพพลิเคชั่นสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งพัฒนาให้สามารถคำนวณแบบขนานบน DPDPTWI ได้ โดย DPDPTWI จัดเตรียม GUI สำหรับสร้างงานและการส่งงานให้กับแอพพลิเคชั่นสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

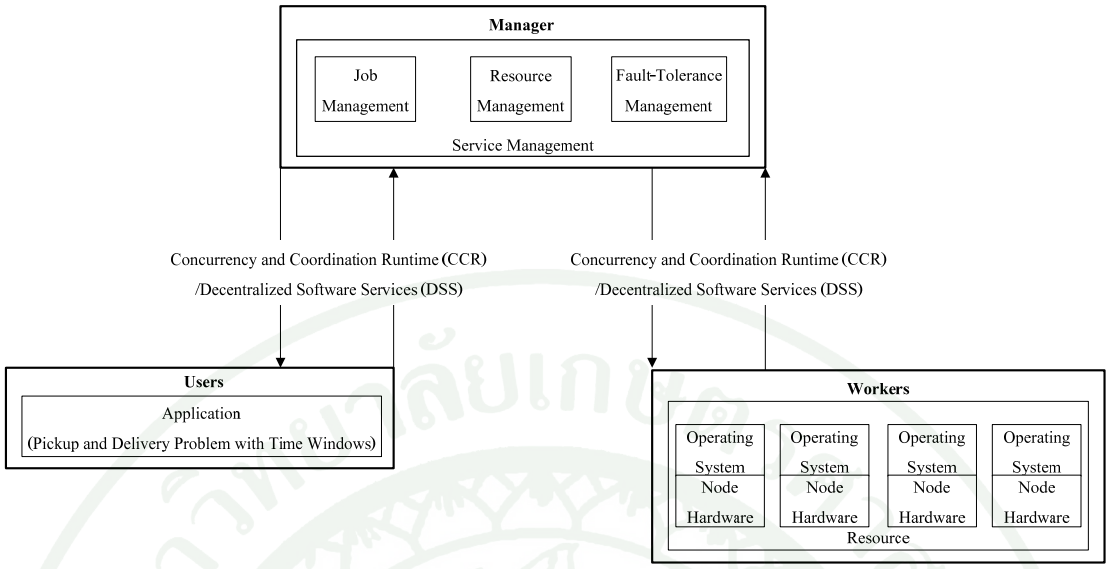
โครงสร้างของ DPDPTWI

ระบบ DPDPTWI มีส่วนประกอบ 3 ส่วน ซึ่งมีโครงสร้างและลักษณะการเชื่อมต่อแสดงดังภาพที่ 18 โดยรายละเอียดของแต่ละส่วนประกอบ มีดังต่อไปนี้

1. เมเนเจอร์ (Manager) เป็นส่วนประกอบที่ประกอบด้วยเซอร์วิสต่างๆ สำหรับใช้ในการสร้าง DPDPTWI ซึ่งประกอบด้วย การจัดการทรัพยากร (Resource management), การจัดการงาน (Job management) และการทนต่อความผิดพร่องของระบบ (Fault-torlerance management)

2. เวิร์คเกอร์ (Worker) เป็นส่วนประกอบที่ใช้ในการทำงานสำหรับงานที่ถูกส่งเข้ามาในระบบ ซึ่งเป็นเครื่องมัลติคอร์คอมพิวเตอร์ที่มีทรัพยากรทางด้านต่างๆ โดยเวิร์คเกอร์จะทำหน้าที่รอรับงานจากเมเนเจอร์ และทำงานเพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง และส่งผลลัพธ์กลับไปยังเมเนเจอร์

3. ผู้ใช้ (User) เป็นส่วนประกอบที่ต้องการใช้ทรัพยากรต่างๆ บนระบบ โดยเป็นผู้ที่ต้องการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งทำหน้าที่สร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง, ส่งงานเข้าสู่ระบบผ่านทางเมเนเจอร์ และรอรับผลลัพธ์ที่ได้จากการประมวลผลจากเครื่องมัลติคอร์คอมพิวเตอร์บนระบบ



ภาพที่ 18 โครงสร้างและลักษณะการเชื่อมต่อของ DPDPTWI

ขั้นตอนการทำงานโดยรวมของ DPDPTWI

ขั้นตอนการทำงานโดยรวมของ DPDPTWI เริ่มตั้งแต่เวิร์คเกอร์ต่างๆ และผู้ใช้ทำการลงทะเบียนไปยังเมเนเจอร์, ผู้ใช้ส่งงานเข้าสู่ระบบจนกระทั่งได้ผลลัพธ์ของงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งกลับไปยังผู้ใช้ ขั้นตอนการทำงานโดยรวมของ DPDPTWI สำหรับประมวลผลแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง แสดงดังภาพที่ 19 ซึ่งมีรายละเอียดของการทำงานในแต่ละขั้นตอน ดังต่อไปนี้

1. เวิร์คเกอร์ทำการร้องขอเพื่อลงทะเบียนกับเมเนเจอร์ซึ่งข้อมูลที่ส่งไปจะเป็นรายละเอียดของเวิร์คเกอร์นั้น

2. เมื่อเมเนเจอร์ได้รับการร้องขอจากเวิร์คเกอร์แล้วนั้น เมเนเจอร์จะทำการเพิ่มข้อมูลของเวิร์คเกอร์เข้าไปสู่ฐานข้อมูลสำหรับเก็บเวิร์คเกอร์

3. เมื่อเพิ่มข้อมูลของเวิร์คเกอร์เข้าสู่ฐานข้อมูลเสร็จเรียบร้อยแล้ว เมเนเจอร์จะส่งข้อความเพื่อยืนยันการลงทะเบียนของเวิร์คเกอร์ดังกล่าวไปยังเวิร์คเกอร์

4. ผู้ใช้ทำการร้องขอเพื่อลงทะเบียนกับเมเนเจอร์ ซึ่งข้อมูลที่ส่งไปจะเป็นรายละเอียดของผู้ใช้

5. เมเนเจอร์ได้รับการร้องขอจากผู้ใช้แล้วนั้น เมเนเจอร์จะทำการเพิ่มข้อมูลของผู้ใช้เข้าไปสู่ฐานข้อมูลสำหรับเก็บผู้ใช้

6. เมื่อเพิ่มข้อมูลของผู้ใช้เข้าสู่ฐานข้อมูลเสร็จเรียบร้อยแล้ว เมเนเจอร์จะส่งข้อความเพื่อยืนยันการลงทะเบียนของผู้ใช้

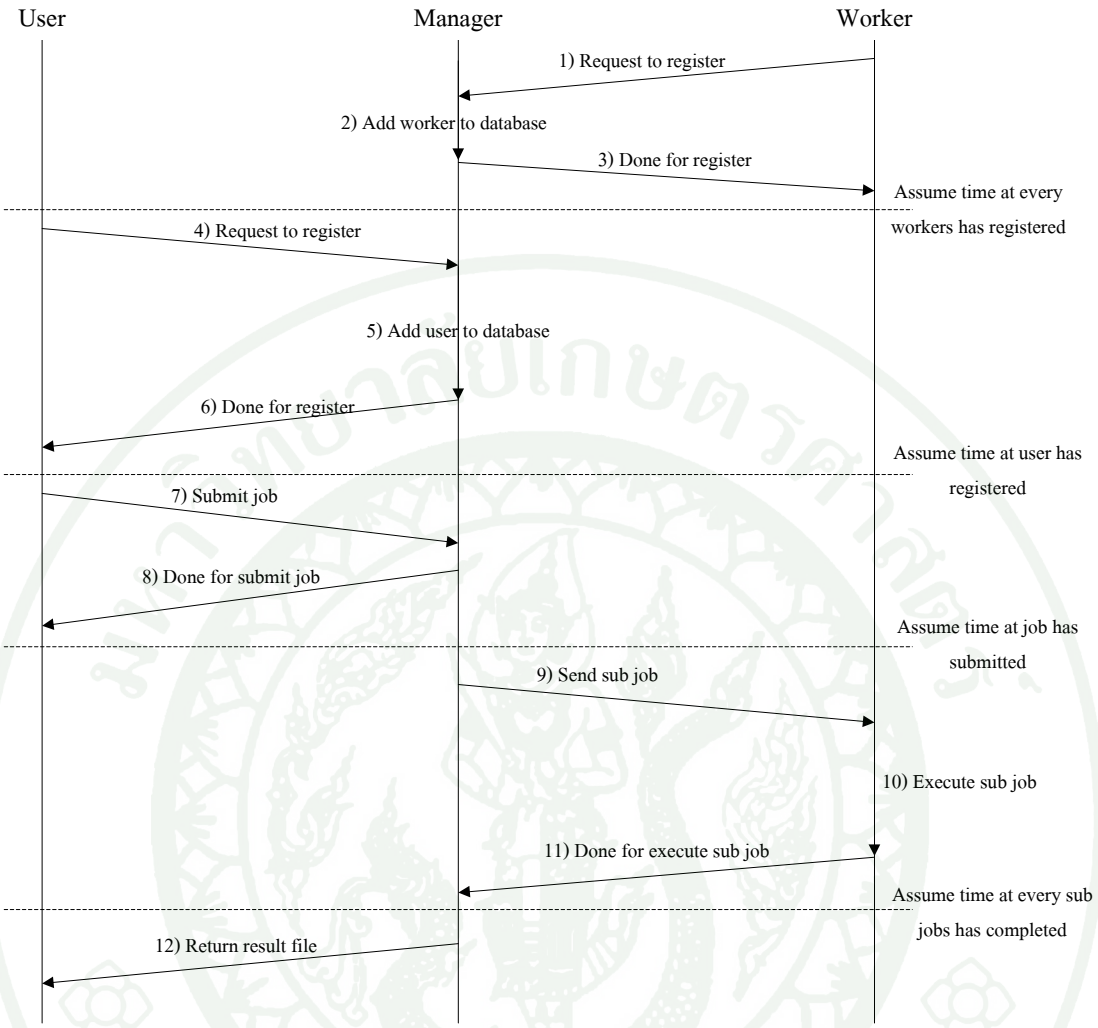
7. เมื่อผู้ใช้สร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ผ่านทางแบบฟอร์มเอ็กเซลที่ระบบจัดเตรียมไว้และทำการบันทึกงานดังกล่าวโดยการกดปุ่มบันทึก (Save) เสร็จเรียบร้อยแล้ว จากนั้นผู้ใช้ทำการส่งงานเข้าสู่ระบบโดยการระบุรายละเอียดของงาน ดังนี้ ตำแหน่งของงาน (Path) ที่ได้ทำการบันทึก, ชื่องาน, จำนวนของลูกค้า, เวลาที่ต้องการใช้สำหรับการประมวลผลของงานดังกล่าว และจำนวนของหน่วยประมวลผลที่ต้องการให้ทำงานเพื่อแก้ปัญหาดังกล่าว

8. เมื่อส่งงานไปยังเมเนเจอร์แล้ว เมเนเจอร์ทำการส่งข้อความตอบกลับไปยังผู้ใช้เพื่อยืนยันการส่งงานเสร็จเรียบร้อย

9. เมเนเจอร์ทำการกระจายงานไปยังกลุ่มของเวิร์คเกอร์ตามจำนวนของหน่วยประมวลผลที่ถูกระบุจากผู้ใช้จากข้อ 7 (เวิร์คเกอร์มีหน่วยประมวลผลมากกว่า 1 หน่วยประมวลผล) กรณีที่จำนวนของเวิร์คเกอร์ไม่เพียงพอต่อความต้องการของงาน เมเนเจอร์จะทำการกระจายงานที่เหลือไปยังกลุ่มของเวิร์คเกอร์ซ้ำตามจำนวนที่เหลือของงานนั้น

10. เวิร์คเกอร์ทำการประมวลผลงาน (ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง) ที่ได้รับจากเมเนเจอร์

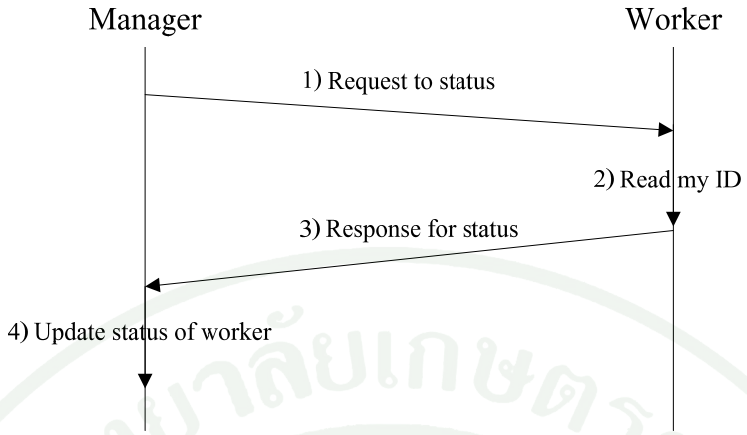
11. เมื่อเวิร์คเกอร์ประมวลผลงานเสร็จเรียบร้อยแล้ว เวิร์คเกอร์จะส่งงานกลับไปยังเมเนเจอร์



ภาพที่ 19 ขั้นตอนการทำงานโดยรวมของ DPDPTWI ของการประมวลผลแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

12. เมื่อเมเนเจอร์ได้รับผลลัพธ์ที่ได้จากการประมวลผลของเวิร์คเกอร์ ก็จะทำการส่งผลลัพธ์ไปยังผู้ใช้

สำหรับขั้นตอนการทำงานสำหรับการตรวจสอบความผิดพลาดของเวิร์คเกอร์นั้น เป็นหน้าที่ของเมเนเจอร์ โดยจะทำหน้าที่ในการส่งข้อความตามช่วงเวลาที่กำหนดเพื่อตรวจสอบสถานะของเวิร์คเกอร์แต่ละเครื่องว่าสามารถรองรับการทำงานได้หรือไม่ ซึ่งขั้นตอนการทำงานสำหรับการตรวจสอบความผิดพลาด แสดงดังภาพที่ 20



ภาพที่ 20 ขั้นตอนการทำงานสำหรับตรวจสอบความผิดพลาดของเวิร์คเกอร์

1. เมเนเจอร์สามารถตรวจสอบสถานะของเวิร์คเกอร์ได้ โดยการส่งข้อความไปยังทุกเวิร์คเกอร์เพื่อตรวจสอบสถานะของเวิร์คเกอร์ว่าขณะนี้ทำงานอยู่หรือไม่

2. เวิร์คเกอร์ทำการอ่านข้อมูลของเครื่องเวิร์คเกอร์

3. เวิร์คเกอร์ทำการส่งข้อความกลับไปยังเมเนเจอร์ กรณีที่เมเนเจอร์ส่งข้อความไปยังเวิร์คเกอร์ที่กำลังประมวลผลงานอยู่นั้น เวิร์คเกอร์จะยังไม่ส่งข้อความตอบกลับจนกระทั่งการประมวลผลงานดังกล่าวเสร็จสิ้นขณะที่เมเนเจอร์จะรอข้อความตอบกลับของเวิร์คเกอร์เช่นกัน

4. เมื่อเมเนเจอร์ได้รับข้อความตอบกลับจากเวิร์คเกอร์แล้วนั้น เมเนเจอร์จะทำการตรวจสอบข้อความว่าได้รับจากเวิร์คเกอร์เครื่องใด และหลังจากนั้นจะทำการปรับแต่งรายการของกลุ่มของเวิร์คเกอร์ กรณีที่เวิร์คเกอร์ทำงานล้มเหลวเมเนเจอร์จะทำการนำงานต่างๆ ที่เวิร์คเกอร์นั้นรับผิดชอบส่งไปยังเวิร์คเกอร์เครื่องอื่นๆ ที่สามารถทำงานได้ ซึ่งทำให้ระบบสามารถดำเนินการต่อไปได้

หมายเหตุ ขั้นตอนการทำงานสำหรับตรวจสอบความผิดพลาดของเวิร์คเกอร์นี้จะทำงานตามช่วงเวลาที่กำหนด

รายละเอียดการออกแบบของ DPDPTWI

DPDPTWI มีเซอร์วิสที่ใช้ในการสร้างระบบเพื่อรองรับการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งประกอบด้วย

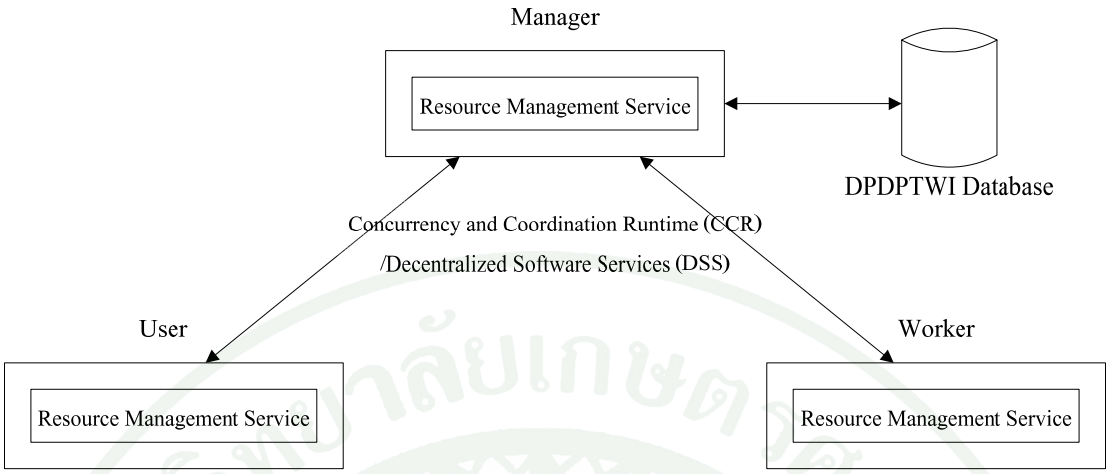
1. Resource Management Service
2. Job Management Service
3. Fault-Tolerance Management Service

รายละเอียดและขั้นตอนการทำงานของแต่ละเซอร์วิสที่ใช้ในการสร้างระบบเพื่อรองรับการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง แสดงดังต่อไปนี้

Resource Management Service

Resource Management Service เป็นเซอร์วิสที่ใช้รวบรวมทรัพยากรต่างๆ บนเครือข่าย พร้อมทั้งเก็บรายละเอียดและสถานะของทรัพยากรต่างๆ ที่ได้รวบรวมบนระบบ เพื่อใช้สำหรับทำงานการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ถูกส่งเข้าสู่ระบบจากผู้ใช้ และเซอร์วิสนี้มีหน้าที่สำหรับการจัดการกับผู้ใช้โดยผู้ที่ดูแลระบบสามารถทำการถอนการลงทะเบียน (Unregister) ของผู้ใช้ได้

การรวบรวมทรัพยากรต่างๆ ของระบบ DPDPTWI เป็นหน้าที่ของเมเนเจอร์ซึ่งทำหน้าที่ในการรวบรวมทรัพยากรต่างๆ ของระบบ โดยเมเนเจอร์ทำหน้าที่รอรับการเข้าร่วมและการออกจากระบบของกลุ่มทรัพยากรของเวิร์คเกอร์ พร้อมทั้งเก็บรายละเอียดและสถานะของเวิร์คเกอร์ ที่ถูกส่งมาจากเวิร์คเกอร์ และในส่วนของการทำหน้าที่สำหรับผู้ใช้ เมเนเจอร์จะทำการรอรับการลงทะเบียนเพื่อขอใช้งานระบบจากผู้ใช้และเมเนเจอร์สามารถทำการถอนการลงทะเบียนของผู้ใช้ได้ตามที่ผู้ดูแลระบบเห็นสมควร โดยรายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการจัดการทรัพยากรแสดงดังภาพที่ 21

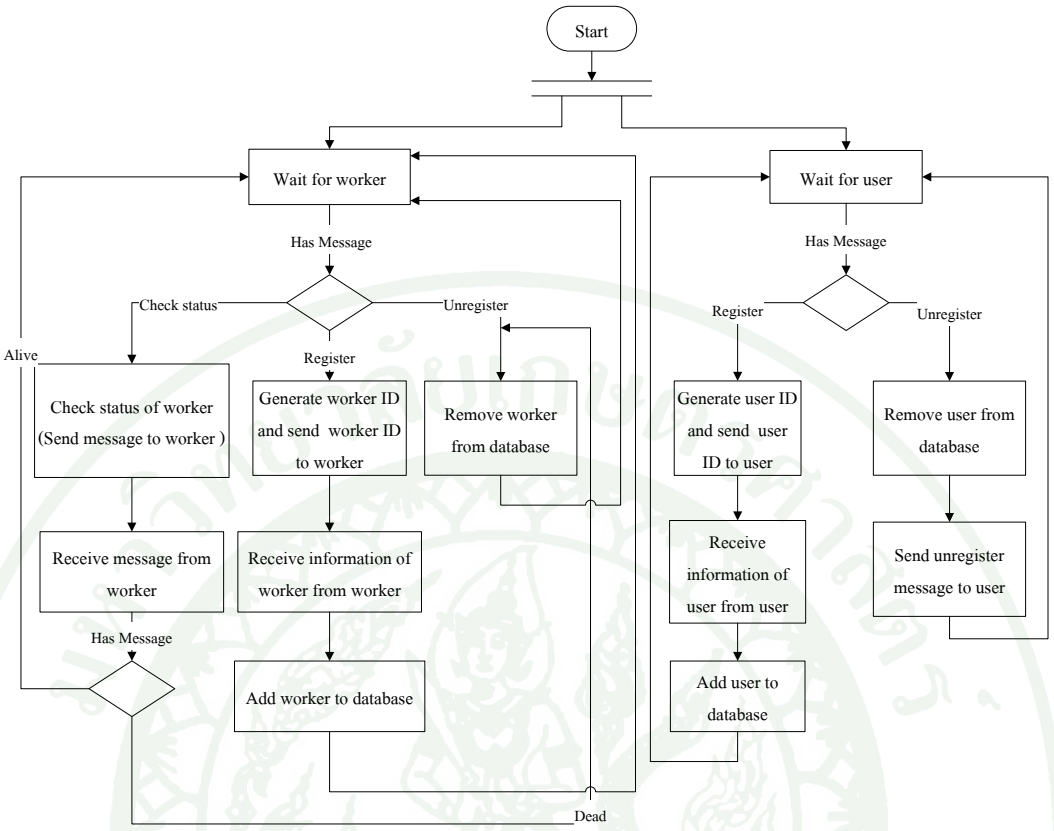


ภาพที่ 21 รายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการจัดการทรัพยากร

จากภาพที่ 21 แสดงถึงรายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการจัดการทรัพยากรซึ่งประกอบด้วย 3 ส่วน คือ เมเนเจอร์, เวิร์คเกอร์ และผู้ใช้ โดยแต่ละส่วนประกอบมีรายละเอียดการทำงานเกี่ยวกับการจัดการทรัพยากรดังต่อไปนี้

1. เมเนเจอร์ (Manager) รายละเอียดการทำงานเกี่ยวกับการจัดการทรัพยากรของเวิร์คเกอร์ที่เมเนเจอร์มีหน้าที่รับผิดชอบ แสดงดังภาพที่ 22 โดยรายละเอียดของการทำงานของเมเนเจอร์ที่ใช้จัดการทรัพยากรระบบ เป็นดังต่อไปนี้

1.1 การทำงานการจัดการทรัพยากรของเวิร์คเกอร์ เป็นการทำงานการรอคอยในการให้บริการกับเวิร์คเกอร์โดยคอยรอรับข้อความจากเวิร์คเกอร์เพื่อนำมาดำเนินการ ซึ่งประเภทของข้อความนั้นมีอยู่ 3 ประเภท ดังต่อไปนี้



ภาพที่ 22 กระบวนการทำงานของเซอร์วิสสำหรับการจัดการทรัพยากรบน DPDPTWI ของเมเนเจอร์

1.1.1 การลงทะเบียน (Register) เป็นข้อความสำหรับการขอเข้าสู่ระบบของเวิร์คเกอร์ โดยเมื่อเมเนเจอร์ได้รับข้อความจากเวิร์คเกอร์แล้ว เมเนเจอร์จะทำการสร้างรหัสสำหรับเวิร์คเกอร์ (Worker ID) (ตัวอย่างเช่น dssp.tcp://booboo:50003/computeservice/dss/notificationtarget/d445f90c-af8c-4e77-b8ef-bb7dc9 ซึ่งรหัสของเวิร์คเกอร์นี้จะนำไปใช้ติดต่อในครั้งต่อไป) แล้วเมเนเจอร์จะส่งรหัสนี้กลับไปยังเวิร์คเกอร์ จากนั้นเมเนเจอร์จะรอรับข้อมูลของเวิร์คเกอร์จากเวิร์คเกอร์เพื่อนำข้อมูลของเวิร์คเกอร์เพิ่มเข้าไปยังฐานข้อมูล

1.1.2 การถอนการลงทะเบียน (Unregister) เป็นข้อความสำหรับการขอออกจากระบบ เมื่อเมเนเจอร์ได้รับข้อความจากเวิร์คเกอร์แล้ว เมเนเจอร์จะทำการลบข้อมูลของเวิร์คเกอร์ดังกล่าวออกจากฐานข้อมูล

1.1.3 การตรวจสอบสถานะ (Check status) เป็นข้อความสำหรับการตรวจสอบสถานะของเวิร์คเกอร์โดยที่เมเนเจอร์จะทำการส่งข้อความไปยังเวิร์คเกอร์เพื่อตรวจสอบสถานะของเวิร์คเกอร์เป็นอย่างไร กรณีที่เวิร์คเกอร์ตาย (Dead) ซึ่งเวิร์คเกอร์ไม่สามารถส่งข้อความตอบกลับมาได้ ดังนั้นเมเนเจอร์จะมีช่วงเวลาในการรอรับการตอบกลับของเวิร์คเกอร์ หากเวิร์คเกอร์ไม่มีการส่งข้อความตอบกลับมายังเมเนเจอร์ภายในช่วงเวลาของการรอ เมเนเจอร์จะถือว่าเวิร์คเกอร์นั้นตายไปแล้ว และเมเนเจอร์จะทำการลบข้อมูลของเวิร์คเกอร์ดังกล่าวออกจากฐานข้อมูล

1.2 การทำงานการจัดการกับผู้ใช้ เป็นการทำงานสำหรับจัดการกับผู้ใช้งานโดยหน้าที่นี้เป็นหน้าที่ของผู้ดูแลระบบโดยการทำงานดังกล่าวจะคอยรอรับข้อความจากเวิร์คเกอร์ เพื่อนำมาดำเนินการ ซึ่งประเภทของข้อความนั้น มีอยู่ 2 ประเภท ดังต่อไปนี้

1.2.1 การลงทะเบียน (Register) เป็นข้อความสำหรับการขอเข้าใช้งานระบบของผู้ใช้ โดยเมื่อเมเนเจอร์ได้รับข้อความจากผู้ใช้ เมเนเจอร์จะทำการสร้างรหัสสำหรับผู้ใช้ (User ID) แล้วเมเนเจอร์จะส่งรหัสนี้กลับไปยังผู้ใช้ จากนั้นเมเนเจอร์จะรอรับข้อมูลของผู้ใช้เพื่อนำข้อมูลของผู้ใช้เพิ่มเข้าไปยังฐานข้อมูล

1.2.2 การถอนการลงทะเบียน (Unregister) เป็นข้อความสำหรับการขอออกจากระบบ โดยแบ่งเป็น 2 กรณี คือ กรณีที่ผู้ใช้ทำการร้องขอ และกรณีที่ผู้ดูแลระบบทำการลบผู้ใช้ออกจากระบบ ซึ่งทั้ง 2 กรณีนี้เมเนเจอร์จะทำการลบข้อมูลของผู้ใช้ออกจากฐานข้อมูลหลังจากนั้นเมเนเจอร์จะทำการส่งข้อความไปยังผู้ใช้งานเพื่อแจ้งให้กับผู้ใช้งานดังกล่าวทราบ

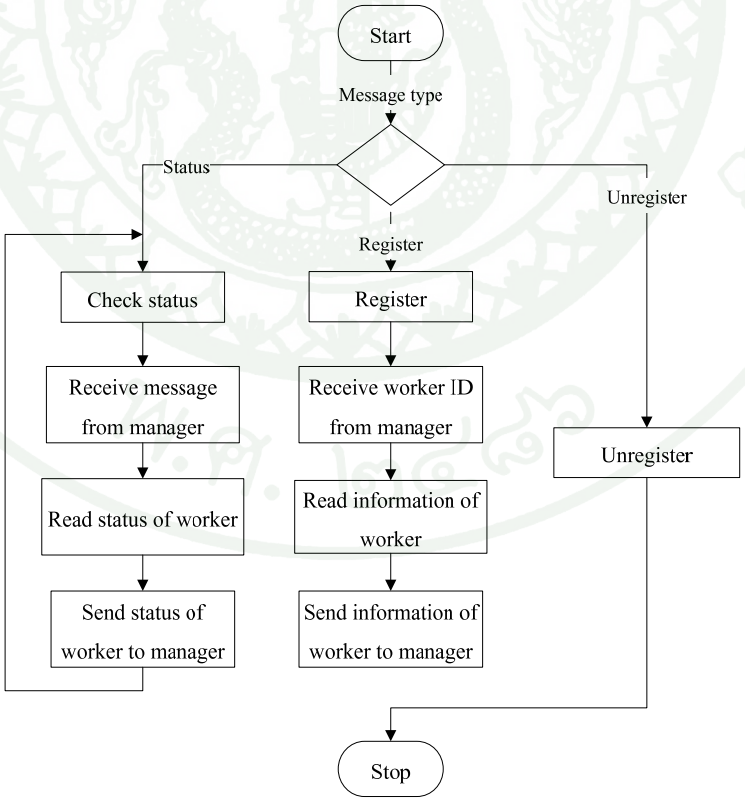
ฐานข้อมูลของ DPDPTWI ที่ใช้ในส่วนของการจัดการทรัพยากรของระบบมีรายละเอียดดังภาพที่ 23

Worker_Info	
PK	**Worker_ID**
	Hostname IP_Address MAC_Address Network_Speed OS_Name Memory_Capacity Storage_Size Processor_Capacity Processor_Count

Client_Info	
PK	**Client_ID**
	Hostname IP_Address MAC_Address Network_Speed OS_Name Memory_Capacity Storage_Size Processor_Capacity Processor_Count

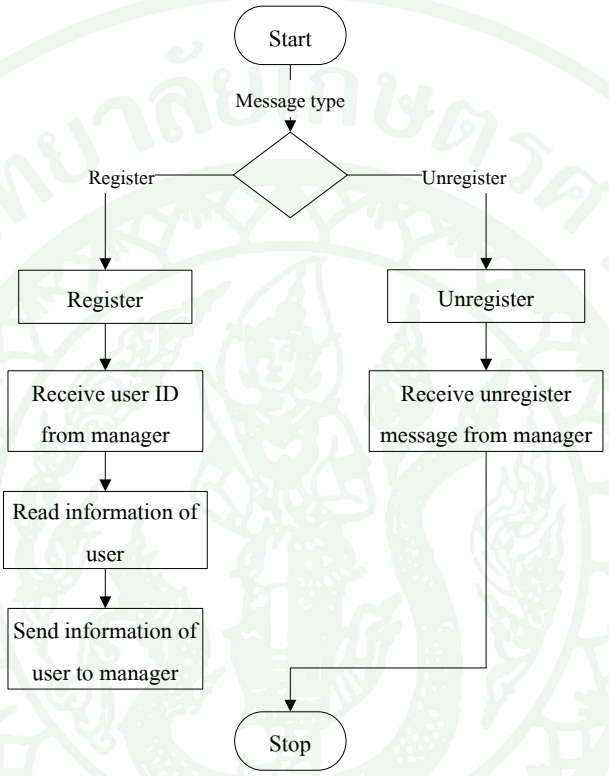
ภาพที่ 23 รายละเอียดฐานข้อมูลของ DPDPTWI ที่ใช้ในส่วนของการจัดการทรัพยากรของระบบ

2. เวิร์คเกอร์ (Worker) รายละเอียดการทำงานสำหรับการจัดการทรัพยากรบนเวิร์คเกอร์ ซึ่งกระบวนการทำงานภายในเวิร์คเกอร์ประกอบด้วย การลงทะเบียนเข้าสู่ระบบ, การตรวจสอบสถานะ และการถอนการลงทะเบียนจากระบบ ซึ่งการทำงานเหล่านี้แสดงดังภาพที่ 24



ภาพที่ 24 การทำงานบนเวิร์คเกอร์สำหรับการจัดการทรัพยากร

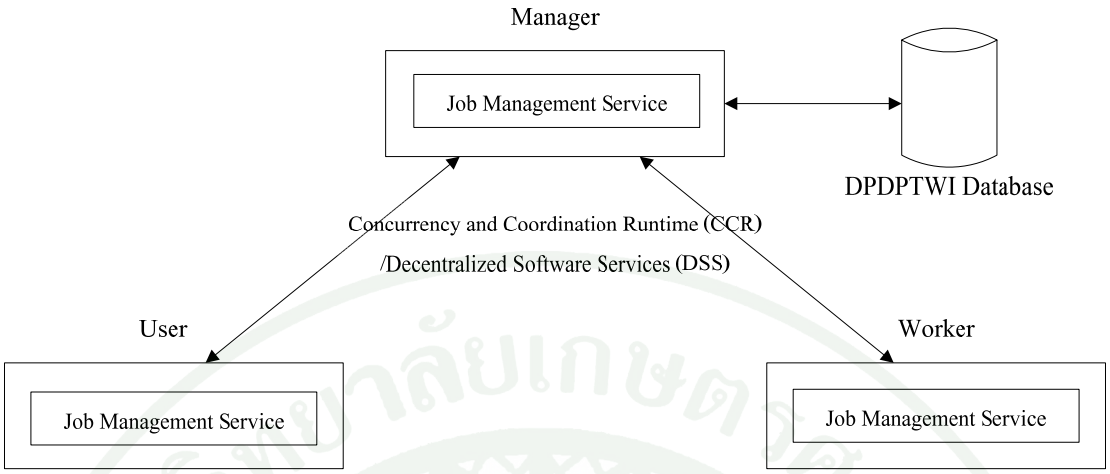
3. ผู้ใช้ (User) การทำงานของเซอร์วิสสำหรับการจัดการทรัพยากรบนผู้ใช้ มีลักษณะดังภาพที่ 25 ซึ่งกระบวนการทำงานในส่วนของผู้ใช้ประกอบด้วย การลงทะเบียนเพื่อขอเข้าใช้ระบบ และการถอนการลงทะเบียน โดยสำหรับการถอนการลงทะเบียนนั้นมี 2 กรณี คือ ผู้ใช้ร้องขอเพื่อทำการออกจากระบบ และผู้ใช้ออกจากระบบโดยจากผู้ดูแลระบบทำการถอดออกจากระบบ



ภาพที่ 25 การทำงานบนผู้ใช้สำหรับการจัดการทรัพยากร

Job Management Service

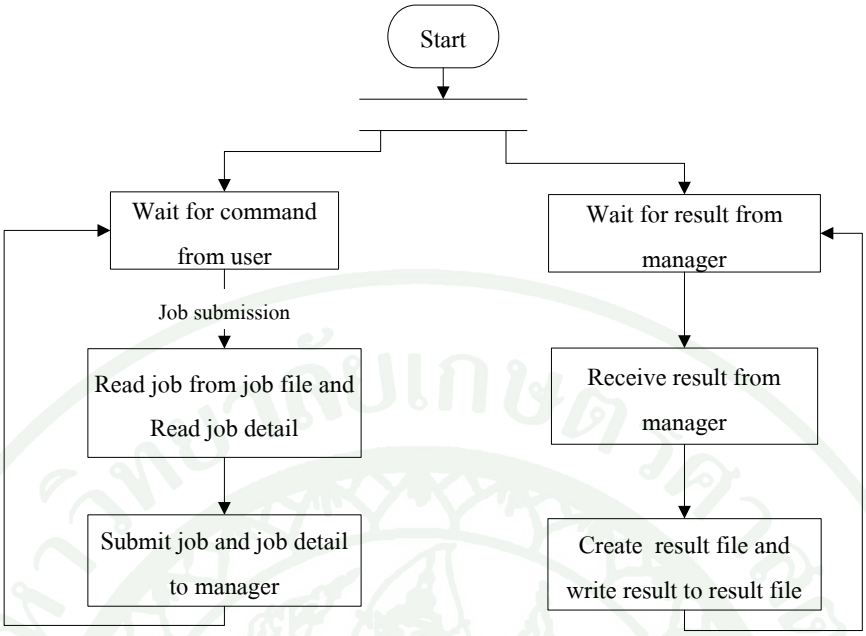
Job Management Service เป็นเซอร์วิสที่ใช้รองรับการส่งงาน, การจัดลำดับงาน และควบคุมงานที่ได้รับจากผู้ใช้ให้กระจายไปทำงานยังทรัพยากรต่างๆ บนระบบ โดยการจัดการงานของ DPDPTWI นั้น ทำโดยมีเมเนเจอร์เป็นส่วนที่ทำหน้าที่ในการรอรับงานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จากผู้ใช้ที่ต้องการส่งงานเข้ามาในระบบ โดยรายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการจัดการงาน แสดงดังภาพที่ 26



ภาพที่ 26 รายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการจัดการงาน

จากภาพที่ 26 การจัดการงานบน DPDPTWI มีส่วนประกอบที่เกี่ยวข้องอยู่ 3 ส่วน ประกอบด้วย ผู้ใช้, เมเนเจอร์ และเวิร์คเกอร์ ซึ่งแต่ละส่วนประกอบมีหน้าที่และการทำงานสำหรับการจัดการงานบน DPDPTWI ดังต่อไปนี้

1. ผู้ใช้ (User) เป็นผู้สร้างและส่งงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เข้าสู่ระบบ โดยข้อมูลของงานที่ผู้ใช้ส่งเข้าสู่ระบบนั้นมีด้วยกัน 2 ส่วน คือ 1) ไฟล์งาน (Job file) เป็นส่วนของไฟล์ข้อมูล (Text file) ที่มีรูปแบบตามกำหนด (การสร้างไฟล์งานนั้นเกิดจากการใช้แบบฟอร์มเอ็กเซล (Excel template) ที่ระบบได้จัดเตรียมไว้) พร้อมทั้งรายละเอียดของข้อมูลของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ตามรูปแบบของแบบฟอร์มของระบบ และ 2) รายละเอียดของงาน (Job detail) เป็นส่วนของรายละเอียดที่ใช้ในการดำเนินงานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ประกอบด้วย ชื่องาน, จำนวนของร้านค้า (ลูกค้า), เวลา (หน่วยเป็นนาที) และจำนวนของหน่วยประมวลผล ซึ่งในการส่งงานเข้าสู่ระบบของ DPDPTWI นั้น ผู้ใช้สามารถส่งงานเข้าสู่ระบบโดยการระบุข้อมูลของงานทั้ง 2 ส่วนที่ได้กล่าวมาโดยผ่านทาง GUI ที่ระบบได้จัดเตรียมไว้ โดยการทำงานที่ใช้จัดการกับงานนี้ทำหน้าที่ในการติดต่อกับเมเนเจอร์เพื่อดำเนินการส่งงานให้กับเมเนเจอร์ต่อไป โดยกระบวนการทำงานสำหรับการจัดการงานของผู้ใช้แสดงดังภาพที่ 27



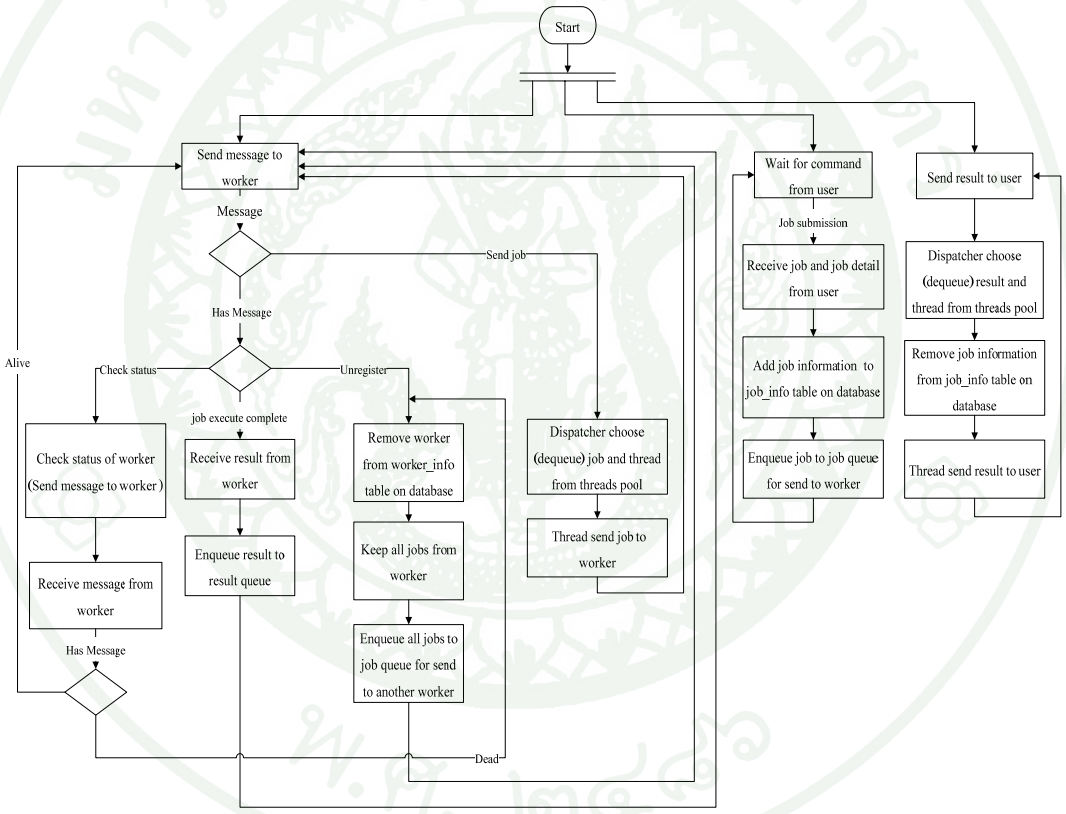
ภาพที่ 27 การทำงานสำหรับการจัดการงานของผู้ใช้

จากภาพที่ 27 การทำงานสำหรับการจัดการงานของผู้ใช้มีหน้าที่รอรับคำสั่งสำหรับการจัดการกับงานจากผู้ใช้งานโดยมีคำสั่ง คือ

การส่งงานเข้าสู่ระบบ (Job submission) ซึ่งเมื่อมีการส่งงานจากผู้ใช้เข้าสู่ระบบผ่านทางการทำงานสำหรับการจัดการงานของผู้ใช้แล้วนั้น การทำงานสำหรับการจัดการงานของผู้ใช้ จะทำการติดต่อไปยังเมเนเจอร์เพื่อส่งงานเข้าสู่ระบบ ซึ่งการทำงานสำหรับการจัดการงานจะอ่านข้อมูลงานจากไฟล์งานที่ผู้ใช้ได้สร้างไว้แล้วและอ่านรายละเอียดของงานที่ผู้ใช้ระบุ โดยใช้ชื่อของงานที่ผู้ใช้ระบุเป็นรหัสของงาน (Work ID) นั้น จากนั้นส่งงานและรายละเอียดของงานไปยังเมเนเจอร์เพื่อให้เมเนเจอร์กระจายงานไปยังกลุ่มของเวิร์คเกอร์

โดยเมื่อผู้ใช้ได้ทำการส่งงานและรายละเอียดของงานไปยังเมเนเจอร์แล้วนั้น ผู้ใช้สามารถที่จะส่งงานอื่นๆ เข้าสู่ระบบได้ในช่วงเวลาของการรอนั้น และเมื่อการประมวลผลเสร็จเรียบร้อยแล้วกลุ่มของเวิร์คเกอร์จะส่งผลลัพธ์กลับไปยังเมเนเจอร์ และเมเนเจอร์จะส่งผลลัพธ์ดังกล่าวกลับมายังผู้ใช้ ซึ่งเมื่อผู้ใช้ได้รับผลลัพธ์จากเมเนเจอร์แล้วนั้น การทำงานสำหรับการจัดการงานของผู้ใช้จะทำการสร้างไฟล์ผลลัพธ์และนำผลลัพธ์เขียนลงในไฟล์นี้ ซึ่งผู้ใช้สามารถเปิดดูได้จากแบบฟอร์มเอ็กเซลที่ระบบจัดเตรียมไว้ให้

2. เมเนเจอร์ (Manager) เป็นส่วนที่ทำหน้าที่ในการให้บริการกับผู้ใช้สำหรับการส่งงาน, การลบงาน และการจัดลำดับของงาน โดยเมื่อมีงานถูกส่งเข้ามาในระบบจากผู้ใช้แล้วนั้นเมเนเจอร์จะทำหน้าที่นำงานที่ได้รับเก็บไว้ที่คิวงาน (Job queue) เพื่อเตรียมส่งให้กับเวิร์คเกอร์ต่อไปซึ่งงานที่อยู่บนคิวงานเหล่านี้ถูกรับผิดชอบโดยกลุ่มของเทรด (Thread pool) ที่สร้างไว้บนเมเนเจอร์ โดยแต่ละงานจะมีเทรดรับผิดชอบในการส่งงานไปยังเวิร์คเกอร์ซึ่งแต่ละเทรดจะทำงานขนานกันไป ซึ่งการแจกจ่ายงานให้กับกลุ่มของเวิร์คเกอร์นั้น จะคำนึงถึงปริมาณจำนวนของงานบนแต่ละเวิร์คเกอร์ที่รับผิดชอบ เพื่อให้แต่ละเวิร์คเกอร์มีปริมาณจำนวนของงานใกล้เคียงกันมากที่สุด โดยกระบวนการทำงานของเซอร์วิสสำหรับการจัดการงานบน DPDPTWI ของเมเนเจอร์ แสดงดังภาพที่ 28



ภาพที่ 28 กระบวนการทำงานของเซอร์วิสสำหรับการจัดการงานบน DPDPTWI ของเมเนเจอร์

จากภาพที่ 28 กระบวนการทำงานของเซอร์วิสสำหรับการจัดการงานบน DPDPTWI ของเมเนเจอร์ มีการทำงานประกอบด้วย 3 งานหลักซึ่งทำงานอยู่ตลอดเวลา โดยรายละเอียดของการทำงานของแต่ละงานแสดงดังต่อไปนี้

2.1 การทำงานการจัดการงานให้กับผู้ใช้ (Wait for command from user) เป็นการทำงานให้กับผู้ใช้สำหรับการส่งงาน โดยกระบวนการทำงานของการทำงานนี้เป็นการตอบสนองต่อคำสั่งของผู้ใช้ที่ทำการสั่งงานผ่านการจัดการงานของผู้ใช้เพื่อนำงานเข้าสู่ระบบ ซึ่งมีรายละเอียดที่แตกต่างจากการจัดการงานของผู้ใช้ คือ เมื่อผู้ใช้ส่งงานมายังเมเนเจอร์ เมเนเจอร์จะรอรับงานและรายละเอียดของงานจากผู้ใช้แล้วทำการเพิ่มข้อมูลของงานเข้าสู่ฐานข้อมูลที่ตำแหน่งตารางของงาน (Job_info) จากนั้นนำงานเข้าไปในคิวงานเพื่อที่จะรอส่งต่อให้กับเวิร์คเกอร์

2.2 การทำงานการจัดการงานให้กับเวิร์คเกอร์ (Send message to worker) เป็นการทำงานให้กับเวิร์คเกอร์สำหรับการส่งงานไปยังเวิร์คเกอร์, การรับผลลัพธ์จากเวิร์คเกอร์, การตรวจสอบสถานะของเวิร์คเกอร์ และการถอนการลงทะเบียนกรณีที่เวิร์คเกอร์ไม่สามารถประมวลผลได้ โดยมีรายละเอียดดังต่อไปนี้

2.2.1 การส่งงานไปยังเวิร์คเกอร์ โดยคิวงานจะมี Dispatcher ซึ่งทำหน้าที่ในการเลือกงานและเลือกเทรดจากกลุ่มเทรด (Thread pool) ขึ้นมาเพื่อที่ให้เทรดทำงาน คือ การส่งงานไปยังเวิร์คเกอร์ ซึ่งการทำงานของเทรดจะทำงานขนานกันและการส่งงานไปยังเวิร์คเกอร์จะคำนึงถึงปริมาณงานแต่ละเวิร์คเกอร์ที่รับผิดชอบ โดยจะส่งงานไปยังเวิร์คเกอร์ที่มีปริมาณงานที่รับผิดชอบน้อยที่สุดเป็นลำดับแรกเสมอ

2.2.2 การรับผลลัพธ์จากเวิร์คเกอร์ เมเนเจอร์รับผลลัพธ์จากเวิร์คเกอร์ที่ส่งกลับมา โดยเมื่อเวิร์คเกอร์ทำงานเสร็จจะส่งผลลัพธ์กลับมายังเมเนเจอร์ และเมเนเจอร์จะนำผลลัพธ์เก็บไว้ในคิวผลลัพธ์ (Result queue) แล้ว Dispatcher จะนำผลลัพธ์และทำการเลือกเทรดเพื่อทำการส่งผลลัพธ์ให้กับผู้ใช้ต่อไป

2.2.3 การตรวจสอบสถานะของเวิร์คเกอร์ เป็นการทำงานที่รับผิดชอบการตรวจสอบสถานะของเวิร์คเกอร์ว่ามีสถานะเป็นอย่างไร (กำลังทำงาน หรือไม่สามารถทำงานได้) โดยการส่งข้อความเพื่อใช้สำหรับการตรวจสอบซึ่งจะส่งข้อความเป็นระยะๆ ตามที่กำหนดซึ่งกรณีที่เวิร์คเกอร์ไม่สามารถทำงานได้แล้ว เมเนเจอร์จะทำการลบข้อมูลของเวิร์คเกอร์ออกจากฐานข้อมูลที่ตำแหน่งตารางเวิร์คเกอร์ (Worker_info) และจะนำงานทั้งหมดที่เวิร์คเกอร์ดังกล่าวรับผิดชอบ ส่งกลับเข้าไปยังคิวงานเพื่อส่งงานเหล่านี้ไปยังเวิร์คเกอร์อื่นต่อไป

2.2.4 การถอนการลงทะเบียนกรณีที่เวิร์คเกอร์ไม่สามารถประมวลผลได้ โดยเมื่อตรวจสอบสถานะของเวิร์คเกอร์แล้วพบว่าเวิร์คเกอร์ไม่สามารถทำงานได้ เมเนเจอร์จะทำการลบข้อมูลของเวิร์คเกอร์ออกจากฐานข้อมูลที่ตำแหน่งตารางเวิร์คเกอร์ และจะนำงานทั้งหมดที่เวิร์คเกอร์ดังกล่าวรับผิดชอบส่งกลับเข้าไปยังคิวงานเพื่อส่งงานเหล่านี้ไปยังเวิร์คเกอร์อื่นต่อไป

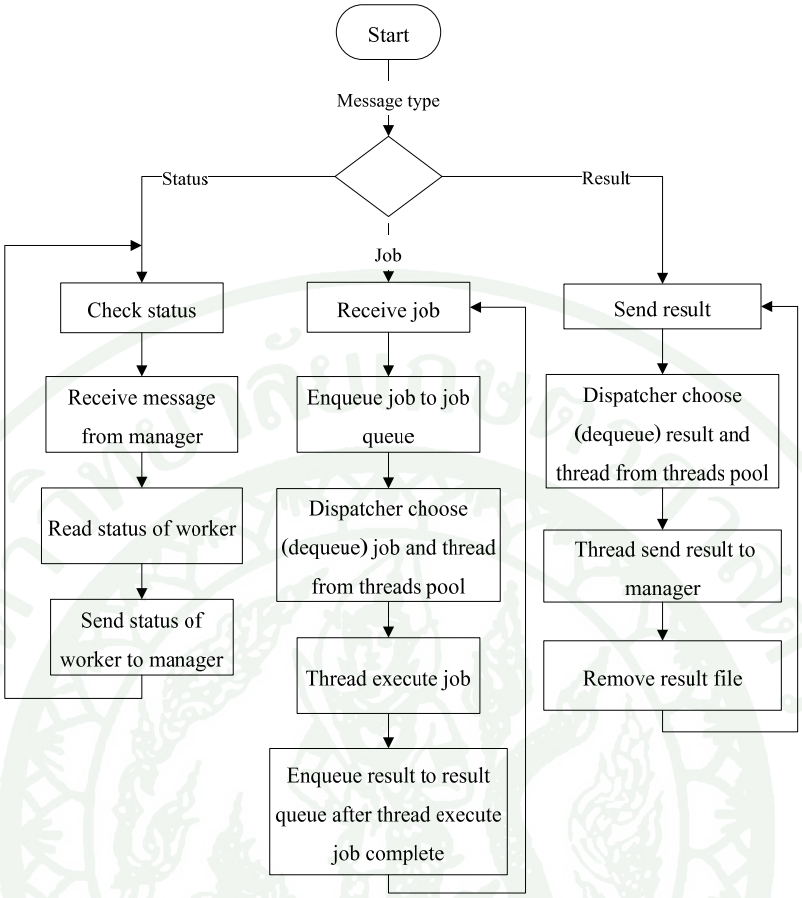
2.3 การทำงานการจัดการงานสำหรับผู้ใช้สำหรับการส่งผลลัพธ์ไปยังผู้ใช้ (Send result to user) เป็นการทำงานเพื่อส่งผลลัพธ์กลับไปยังผู้ใช้ โดยเมื่อมีผลลัพธ์เข้ามายังคิวผลลัพธ์ (Result queue) การทำงานการส่งผลลัพธ์จะเริ่มทำงานโดยการนำผลลัพธ์ออกจากคิวผลลัพธ์ (Dispatcher ทำหน้าที่เลือกงานออกจากคิวผลลัพธ์และเลือกเทรดจากกลุ่มเทรด) และทำการลบข้อมูลของงานที่ตรงกับผลลัพธ์ที่นำออกจากคิวผลลัพธ์ออกจากฐานข้อมูลที่ตำแหน่งตารางของงาน ต่อจากนั้นจะทำการส่งผลลัพธ์ไปยังผู้ใช้ต่อไป

ฐานข้อมูลของ DPDPTWI ที่ใช้ในส่วนของการจัดการงานของระบบมีรายละเอียดดังภาพที่ 29

Job_Info	
PK	**Job_ID**
	Job_Name Client_Request Worker_Response Processor_Uses Time_Span

ภาพที่ 29 รายละเอียดฐานข้อมูลของ DPDPTWI ที่ใช้ในส่วนของการจัดการงานของระบบ

3. เวิร์คเกอร์ (Worker) เป็นส่วนที่ทำหน้าที่ในการประมวลผลให้กับงานต่างๆ ที่เกิดขึ้นในระบบ โดยมีการทำงานสำหรับการจัดการงานบนเวิร์คเกอร์ ซึ่งติดตั้งอยู่บนเวิร์คเกอร์ต่างๆ ในระบบ เป็นผู้ดำเนินการสำหรับการประมวลผลงาน, การแจ้งสถานะของเวิร์คเกอร์ และการส่ง ผลลัพธ์กลับไปยังเมเนเจอร์ โดยกระบวนการทำงานสำหรับการจัดการงานบนเวิร์คเกอร์ แสดงดังภาพที่ 30



ภาพที่ 30 กระบวนการทำงานสำหรับการจัดการงานบนเวิร์คเกอร์

จากภาพที่ 30 แสดงรายละเอียดกระบวนการทำงานสำหรับการจัดการงานบนเวิร์คเกอร์ ซึ่งจะมีการทำงานดังต่อไปนี้

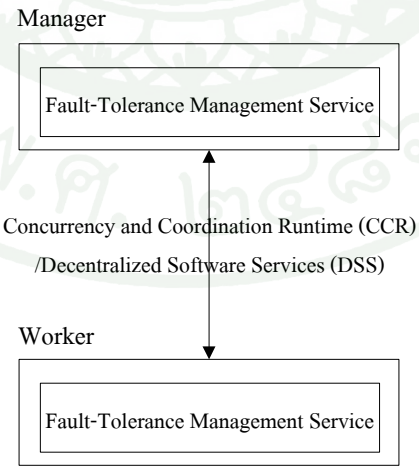
3.1 การประมวลผลงาน เป็นการประมวลผลงาน (การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง) ที่ได้รับจากเมเนเจอร์โดยจะนำงานที่ได้รับเข้าไปยังคิวงาน ซึ่ง Dispatcher จะรับผิดชอบในการนำงานออกจากคิวงานและเลือกเทรดที่จะรับผิดชอบประมวลผลงานดังกล่าวจากกลุ่มเทรด ซึ่งการประมวลผลงานสามารถประมวลผลงานได้พร้อมๆ กันตามจำนวนเทรดที่อยู่ในกลุ่มเทรด เมื่อทำการประมวลผลงานเสร็จสิ้นแล้วจะทำการส่งผลลัพธ์เข้าไปที่คิวผลลัพธ์เพื่อรอส่งให้กับเมเนเจอร์ต่อไป

3.2 การแจ้งสถานะของเวิร์คเกอร์ เป็นการแจ้งสถานะของเวิร์คเกอร์กลับไปยังเมเนเจอร์ เมื่อเวิร์คเกอร์ได้รับข้อความจากเมเนเจอร์ เวิร์คเกอร์จะทำการอ่านข้อมูลสถานะของตนเอง (กรณีที่เวิร์คเกอร์ตายข้อมูลจะไม่ถูกส่งไปยังเมเนเจอร์) และส่งให้กับเมเนเจอร์ ซึ่งกรณีที่เวิร์คเกอร์ทำการประมวลผลงานอยู่นั้น การแจ้งสถานะของเวิร์คเกอร์จะรอให้เวิร์คเกอร์ประมวลผลงานดังกล่าวจนเสร็จก่อนแล้วจึงส่งข้อมูลสถานะของเวิร์คเกอร์ให้กับเมเนเจอร์ต่อไป

3.3 การส่งผลลัพธ์ เป็นการส่งผลลัพธ์ไปยังเมเจอร์ ซึ่งจะนำผลลัพธ์ที่เก็บอยู่ในคิวของผลลัพธ์ (เป็นหน้าที่ของ Dispatcher) และทำการเลือกเทรดจากกลุ่มเทรดแล้วทำการส่งผลลัพธ์ไปยังเมเนเจอร์ ต่อจากนั้นจะทำการลบไฟล์ผลลัพธ์นั้นบนเวิร์คเกอร์

Fault-Tolerance Management Service

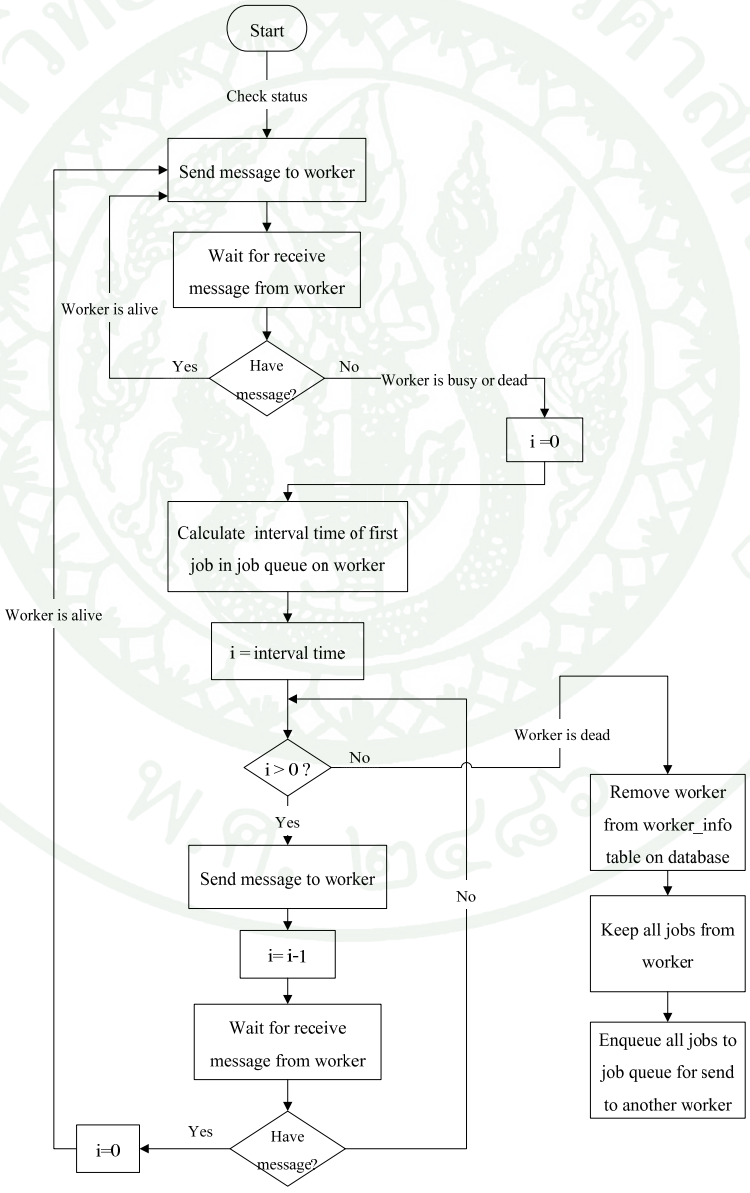
Fault-Tolerance Management Service เป็นเซอร์วิสที่ใช้รองรับต่อความผิดพลาดของเวิร์คเกอร์ในกลุ่มทรัพยากรเวิร์คเกอร์ ซึ่งมีเมเนเจอร์เป็นส่วนที่ทำหน้าที่ในการตรวจสอบสถานะการทำงานของแต่ละเวิร์คเกอร์ซึ่งทำให้ระบบสามารถดำเนินการต่อไปได้ เมื่อเกิดความความผิดพลาดของเวิร์คเกอร์ใดๆ ระบบการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ทนต่อความผิดพลาดของเวิร์คเกอร์ในระบบ มีรายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการทนต่อความผิดพลาดของเวิร์คเกอร์ แสดงดังภาพที่ 31



ภาพที่ 31 รายละเอียดส่วนประกอบของ DPDPTWI ที่เกี่ยวข้องกับการทนต่อความผิดพลาด

จากภาพที่ 31 การทนต่อความผิดพลาดบน DPDPTWI มีส่วนประกอบที่เกี่ยวข้องอยู่ 2 ส่วน ประกอบด้วย เมเนเจอร์ และเวิร์คเกอร์ ซึ่งแต่ละส่วนประกอบมีหน้าที่และการทำงานเพื่อรองรับการทนต่อความผิดพลาดบน DPDPTWI ดังต่อไปนี้

1. เมเนเจอร์ (Manager) เป็นส่วนที่ทำหน้าที่ในการตรวจสอบสถานะของเวิร์คเกอร์ ด้วยวิธีการส่งข้อความไปยังทุกเวิร์คเกอร์ตามช่วงเวลาที่กำหนด เพื่อตรวจสอบสถานะ โดยกระบวนการทำงานของเซอร์วิสที่ทนต่อความผิดพลาดบน DPDPTWI ของเมเนเจอร์แสดงดังภาพที่ 32



ภาพที่ 32 กระบวนการทำงานของเซอร์วิสที่ทนต่อความผิดพลาดบน DPDPTWI ของเมเนเจอร์

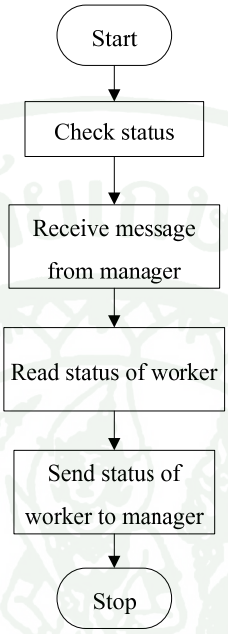
จากภาพที่ 32 กระบวนการทำงานของเซอร์วิสที่ทนต่อความผิดพลาดบน DPDPTWI ของเมเนเจอร์ มีการทำงานคือ การตรวจสอบสถานะของเวิร์คเกอร์ ซึ่งทำงานทุกช่วงเวลาที่กำหนด โดยรายละเอียดของการทำงานการตรวจสอบสถานะของเวิร์คเกอร์ใช้วิธีการส่งข้อความไปยังเวิร์คเกอร์และรอรับข้อความจากเวิร์คเกอร์ กรณีที่มีข้อความจากเวิร์คเกอร์ส่งกลับมายังเมเนเจอร์ แสดงว่าเวิร์คเกอร์ดังกล่าวสามารถทำงานได้ ส่วนในกรณีที่ไม่มีข้อความจากเวิร์คเกอร์ตอบกลับเป็นไปได้ 2 สาเหตุ คือ เวิร์คเกอร์กำลังประมวลผลงานหรือเวิร์คเกอร์ไม่สามารถทำงานได้ โดยจะรู้สาเหตุได้นั้น ต้องทำการตรวจสอบปริมาณงานที่เวิร์คเกอร์รับผิดชอบ (รายละเอียดของงานจะถูกเก็บไว้ขณะที่เมเนเจอร์ทำการส่งงานไปยังเวิร์คเกอร์)

กรณีเวิร์คเกอร์มีงานที่รับผิดชอบแสดงว่าเวิร์คเกอร์กำลังประมวลผลงาน ซึ่งจะทำการรอตามช่วงเวลา (Interval time of job) ของงานนั้น (รอจนกว่าการประมวลผลงานดังกล่าวเสร็จ) ซึ่งค่าของช่วงเวลาของงานหาได้จาก (เวลาที่ผู้ใช้ต้องการประมวลผลของงานนั้น (นาที) คูณด้วย 60) แล้วหารด้วยช่วงเวลาของการตรวจสอบสถานะที่กำหนด โดยเมเนเจอร์จะรอจนกว่าค่าช่วงเวลาของงานนั้นหมด (ค่าเป็น 0) ซึ่งขณะที่รอ หากเกิดกรณีที่มีข้อความจากเวิร์คเกอร์ส่งกลับมายังเมเนเจอร์แสดงว่าเวิร์คเกอร์ยังสามารถทำงานได้

กรณีค่าช่วงเวลาของงานหมดและไม่มีข้อความจากเวิร์คเกอร์ส่งกลับมายังเมเนเจอร์แสดงว่าเวิร์คเกอร์ทำงานล้มเหลว (ไม่สามารถทำงานต่อไปได้) จากนั้นทำการลบเวิร์คเกอร์นั้นออกจากระบบแล้วนำงานทั้งหมดที่เวิร์คเกอร์ดังกล่าวรับผิดชอบส่งเข้าไปยังคิวงานเพื่อรอการกระจายไปยังเวิร์คเกอร์อื่นต่อไป

กรณีที่เวิร์คเกอร์ไม่มีงานที่รับผิดชอบและไม่มีการตอบกลับแสดงว่าเวิร์คเกอร์ไม่สามารถทำงานได้และจะทำการลบเวิร์คเกอร์นั้นออกจากระบบ

2. เวิร์คเกอร์ (Worker) เป็นส่วนที่ทำหน้าที่ในการส่งข้อมูลสถานะของเวิร์คเกอร์ไปยังเมเนเจอร์เมื่อมีการร้องขอจากเมเนเจอร์โดยกระบวนการทำงานแสดงดังภาพที่ 33

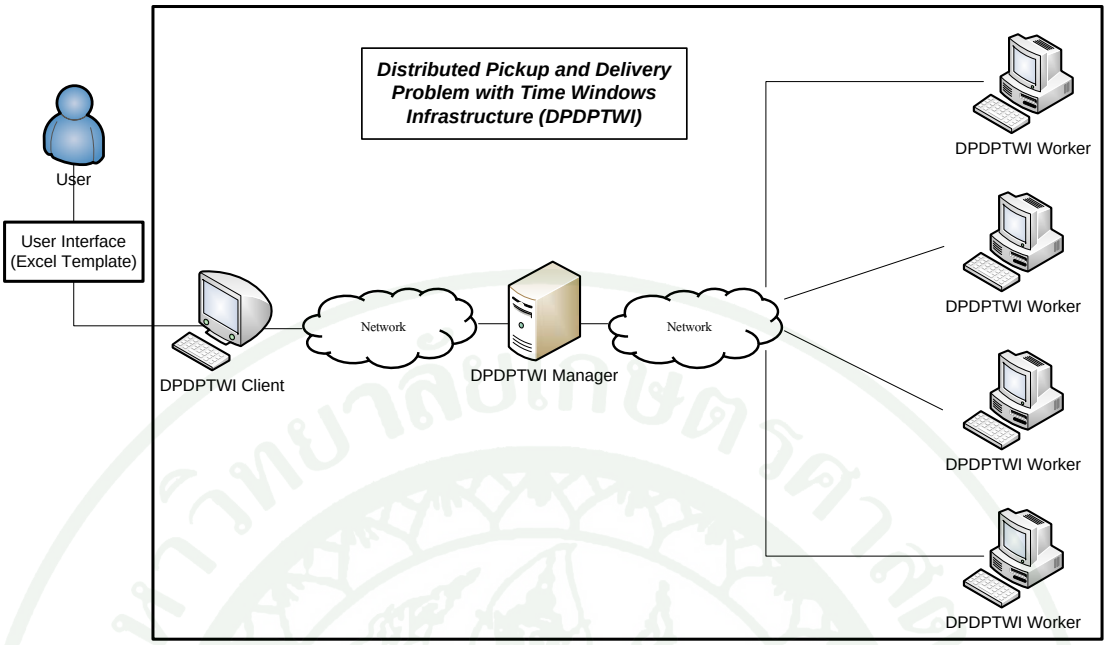


ภาพที่ 33 การทำงานการส่งข้อมูลสถานะของเวิร์คเกอร์ไปยังเมเนเจอร์

จากภาพที่ 33 แสดงการทำงานการส่งข้อมูลสถานะของเวิร์คเกอร์ไปยังเมเนเจอร์ โดยเมื่อรับข้อความจากเมเนเจอร์เพื่อร้องขอข้อมูลสถานะของเวิร์คเกอร์ จากนั้นเวิร์คเกอร์จะอ่านค่าสถานะของตนเองและส่งกลับไปยังเมเนเจอร์ กรณีที่เวิร์คเกอร์กำลังประมวลผลงานอยู่นั้นระบบจะรอจนกว่าประมวลผลงานดังกล่าวเสร็จ จึงจะส่งข้อมูลสถานะของเวิร์คเกอร์ไปยังเมเนเจอร์ต่อไป

โครงสร้างโดยรวมของระบบการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

การออกแบบและพัฒนากรอบการทำงานสมรรถนะสูงเพื่อรองรับการคำนวณแบบขนานสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งของงานวิจัยนี้มีโครงสร้างโดยรวมของงานวิจัยแสดงดังภาพที่ 34

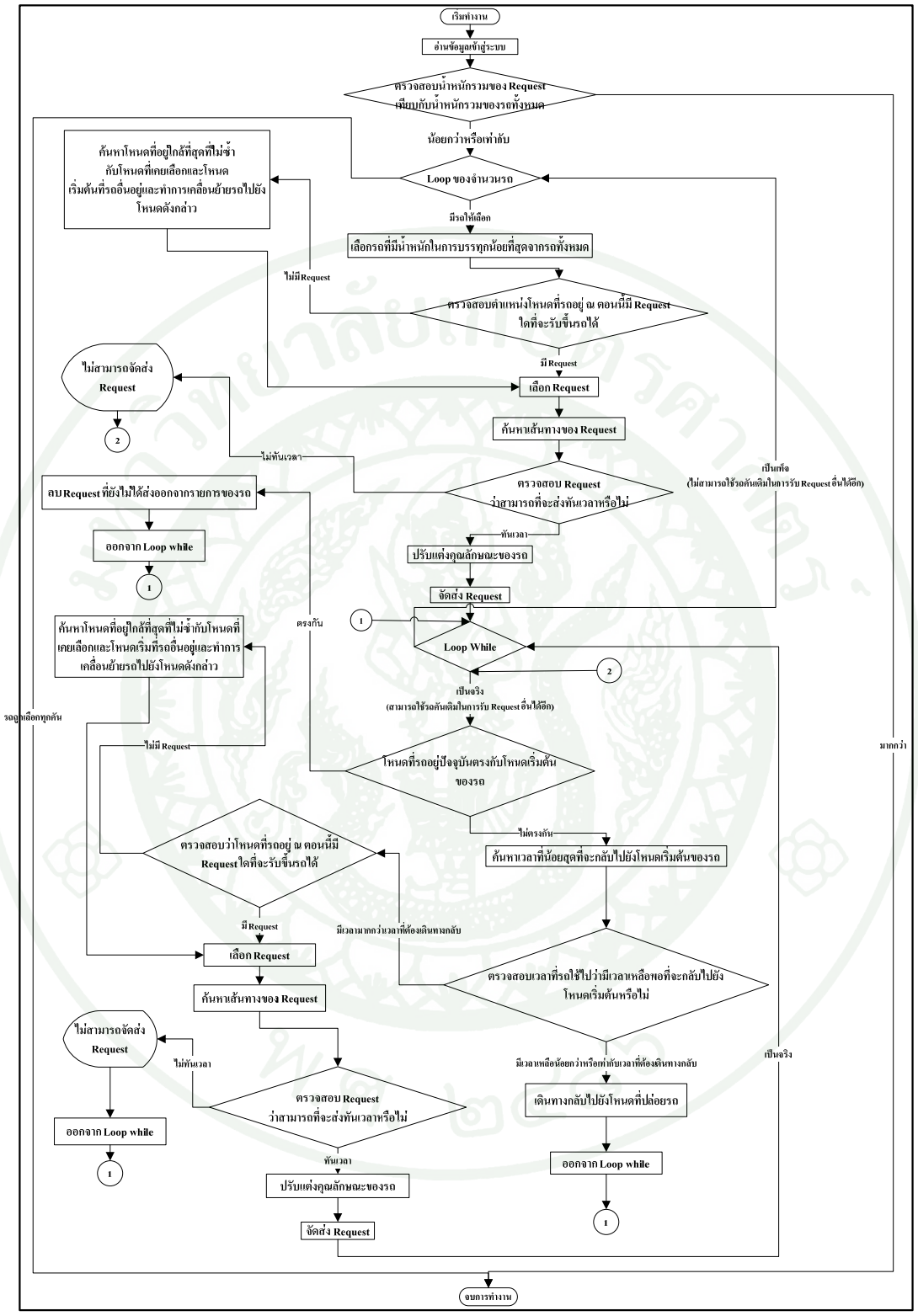


ภาพที่ 34 โครงสร้างโดยรวมของระบบการคำนวณแบบขนานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

โมเดลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วง เวลาในการรับ/ส่ง (Model for PDPTW)

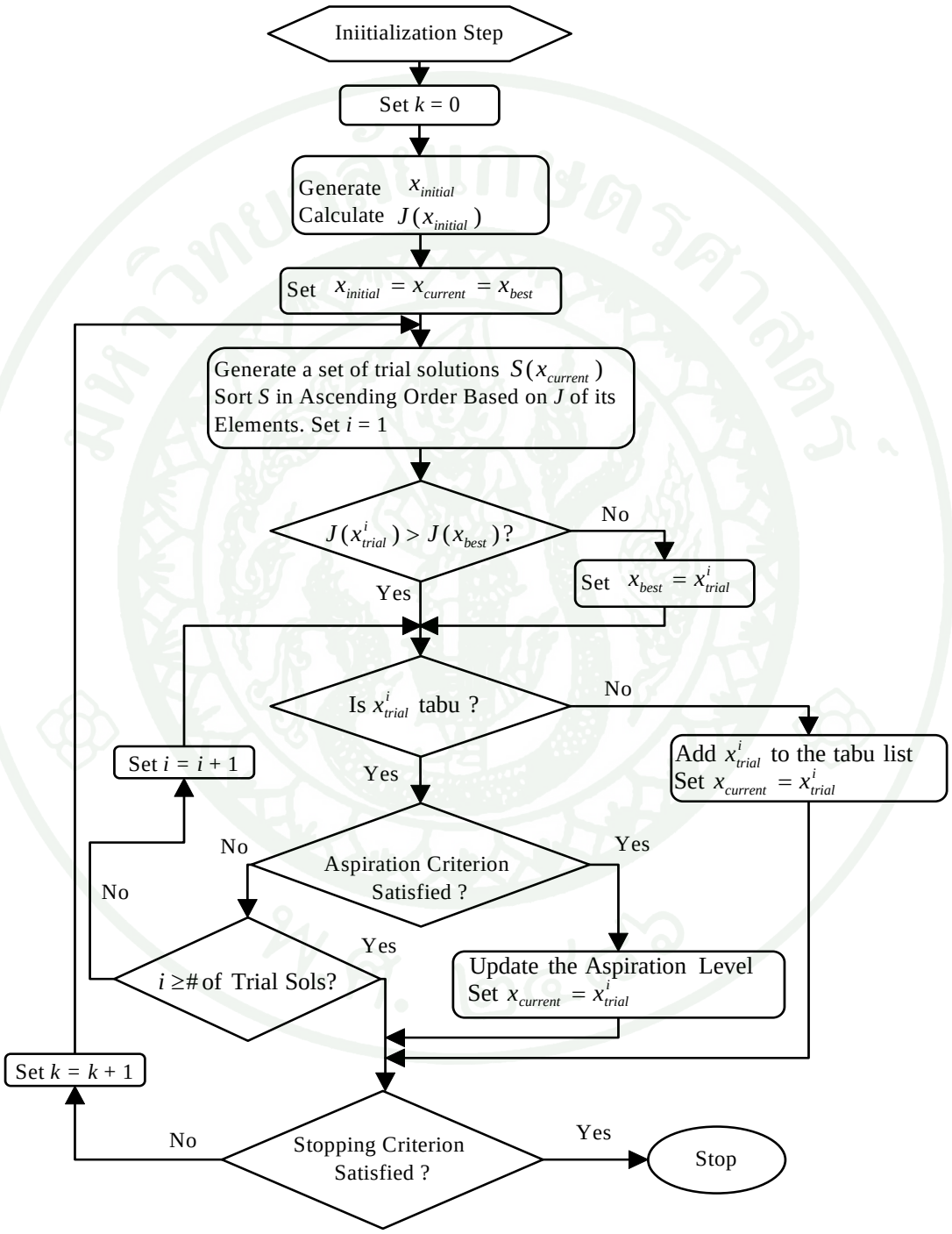
งานวิจัยนี้ผู้วิจัยเลือกใช้โมเดล 2 แบบ คือ Solve and improve algorithm (SIA) และ Tabu search algorithm (TSA) (Tabu search algorithm, 1993) สำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เนื่องจากโมเดลแต่ละแบบมีวิธีการหาคำตอบที่แตกต่างกัน, คำตอบที่ได้เป็นที่น่าพอใจต่อผู้ใช้แตกต่างกัน และการนำคำตอบไปใช้กับปัญหาการจัดเส้นทางยานพาหนะบนสถานการณ์จริง ดังนั้นเพื่อตอบสนองต่อผู้ใช้และสามารถนำคำตอบไปใช้งานได้จริง ผู้วิจัยพิจารณาเลือกใช้โมเดล 2 แบบเพื่อหาคำตอบจากแต่ละโมเดลว่าโมเดลใดให้คำตอบที่ดีที่สุดแก่ผู้ใช้ ซึ่งรายละเอียดของโมเดลมีลักษณะการทำงานดังนี้

1. กระบวนการทำงานของโมเดล SIA สำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง



ภาพที่ 35 ลักษณะการทำงานของโมเดล SIA สำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

2. กระบวนการทำงานของโมเดล TSA สำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง



ภาพที่ 36 ลักษณะการทำงานของโมเดล TSA สำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

The general algorithm of Tabu Search can be described in steps as follows:

Step 1: Set the iteration counter k=0 and randomly generate an initial solution $x_{initial}$. Set this solution as the current solution as well as the best solution, x_{best}, i.e. $x_{initial}= x_{current}=x_{best}$.

Step 2: Randomly generate a set of trial solutions x_{trial}s in the neighborhood of the current solution, i.e. create $S(x_{current})$. Sort the elements of S based on their objective function values in ascending order as the problem is a minimization one. Let us define x_{trial}^{i} as the i^{th} trial solution in the sorted set, 1 i nt. Here, x_{trial}^{1} represents the best trial solution in S in terms of objective function value associated with it.

Step 3: Set i=1. If $J(x_{trial}^{i})$ $J(x_{best})$ go to Step 4, else set $x_{best}= x_{trial}^{i}$ and go to Step 4.

Step 4: Check the tabu status of x_{trial}^{i}. If it is not in the tabu list then put it in the tabu list, set $x_{current}=x_{trial}^{i}$, and go to Step 7. If it is in tabu list go to Step 5.

Step 5: Check the aspiration criterion of x_{trial}^{i}. If satisfied then override the tabu restrictions, update the aspiration level, set $x_{current}=x_{trial}^{i}$, and go to Step 7. If not set $i=i+1$ and go to Step 6.

Step 6: If $i>nt$ go to Step 7, else go back to Step 4.

Step 7: Check the stopping criteria. If one of them is satisfied then stop, else set $k=k+1$ and go back to Step 2.

ภาพที่ 36 (ต่อ)

การแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จากโมเดลทั้ง 2 แบบนั้นจะเห็นว่าในการคำนวณเพื่อจัดเส้นทางที่เหมาะสมให้กับยานพาหนะจำเป็นต้องใช้ระยะเวลาสำหรับการจัดเส้นทางที่สูง เนื่องจากความซับซ้อนของการตัดสินใจ เพราะจำนวนของตัวแปรที่ใช้ในการตัดสินใจมีจำนวนมากและความซับซ้อนของวิธีหาคำตอบเพื่อให้ได้คำตอบดีที่สุดหรือเป็นที่น่าพอใจต่อผู้ใช้ จึงส่งผลให้ต้องใช้พลังการประมวลผลมากขึ้นเช่นกัน และจะเห็นได้ว่าเพื่อให้ได้ผลลัพธ์ที่มีความแตกต่างและเหมาะสมสำหรับเป็นทางเลือกให้แก่ผู้ใช้งานสามารถนำไปพิจารณาเพื่อตัดสินใจในการเลือกผลลัพธ์ที่ดีที่สุดต่อผู้ใช้งานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ดังนั้นแนวคิดของการคำนวณแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด

โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งของงานวิจัยนี้ คือ แบ่งอินพุตสำหรับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ออกเป็นส่วนๆ แล้วกระจายไปคำนวณยังเครื่องคอมพิวเตอร์ต่างๆ ที่มีโมเดลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งนั้นอยู่ หรือกล่าวได้ว่าเป็นการคำนวณแบบขนานในลักษณะ Domain decomposition

การแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งด้วยโมเดล SIA และ TSA บนแนวคิดของการคำนวณแบบขนาน โดยใช้กรอบการทำงานแบบสมรรถนะสูง

งานวิจัยนี้ใช้เครื่องคอมพิวเตอร์ที่ประกอบด้วย จำนวนของหน่วยประมวลผลที่มีมากกว่า 1 หน่วยประมวลผล ดังนั้น เพื่อเพิ่มประสิทธิภาพของการประมวลผลแบบขนานให้มีประสิทธิภาพมากยิ่งขึ้น ผู้วิจัยจึงนำแนวคิดของการประมวลผลแบบมัลติเทรดเข้ามาใช้เพิ่มประสิทธิภาพการคำนวณแบบขนานเพื่อช่วยลดระยะเวลาในการประมวลผลของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่มีปริมาณมากบนแต่ละเครื่องคอมพิวเตอร์ โดยใช้แนวคิดการพัฒนาโปรแกรมแบบ CCR ซึ่งรองรับการทำงานแบบมัลติเทรด และใช้แนวคิดการพัฒนาโปรแกรมแบบ DSS สำหรับการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ (Message passing) (รายละเอียดของ CCR และ DSS ได้กล่าวไว้ในหัวข้อ ความรู้พื้นฐาน)

ดังนั้น ระบบการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จึงมีความสามารถในการประมวลผลแบบขนาน 2 รูปแบบ คือ

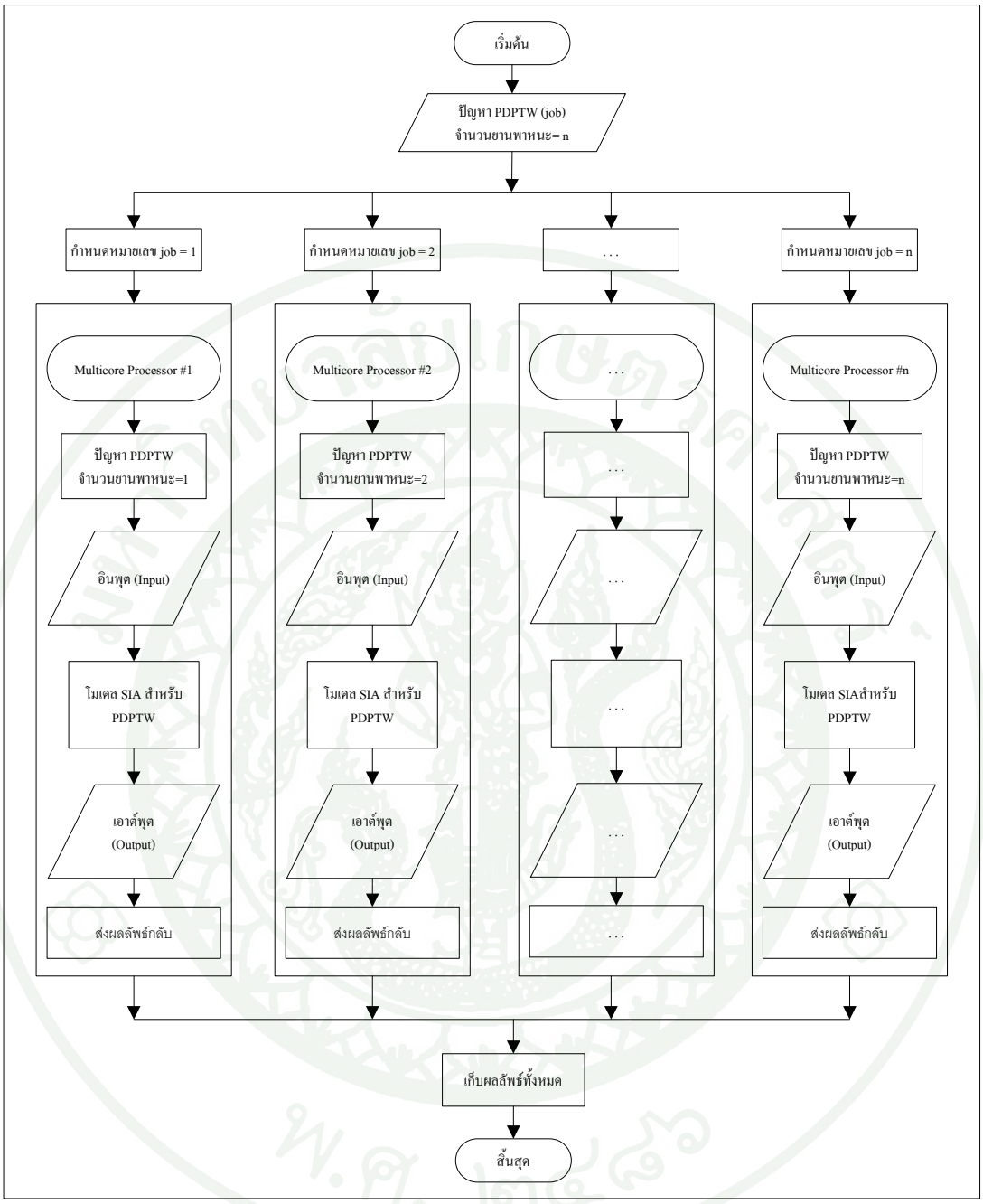
1. สามารถคำนวณแบบขนานด้วยวิธีการกระจายปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ไปยังกลุ่มของเครื่องคอมพิวเตอร์โดยใช้แนวคิดของการพัฒนาโปรแกรมแบบ DSS

2. สามารถคำนวณแบบขนานด้วยวิธีการกระจายปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ไปยังกลุ่มของเทรดบนเครื่องคอมพิวเตอร์ โดยใช้แนวคิดของการพัฒนาโปรแกรมแบบ CCR

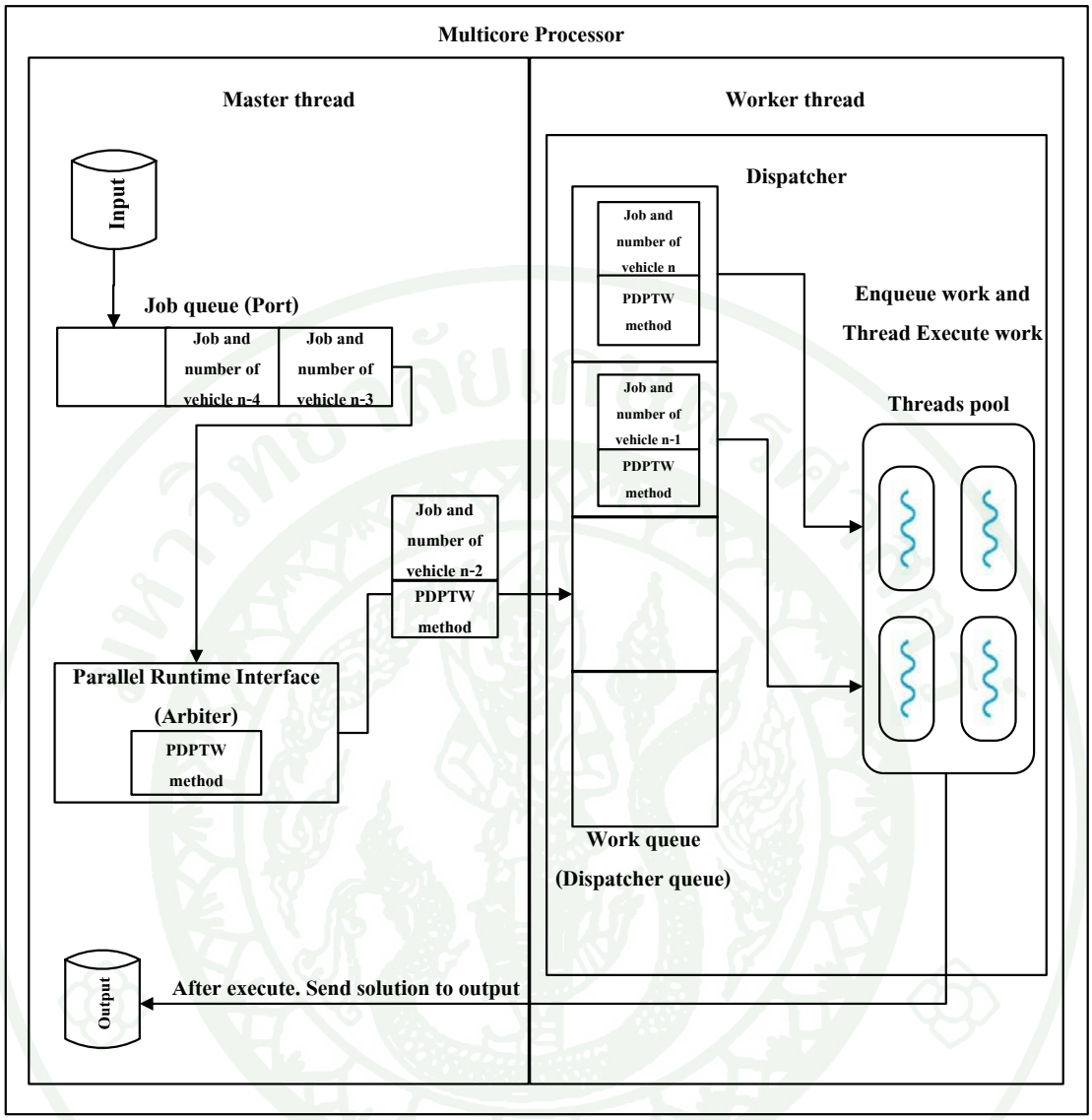
คุณสมบัติทั้ง 2 รูปแบบนี้จะช่วยเพิ่มประสิทธิภาพของการประมวลผลได้ดียิ่งขึ้นและสามารถรองรับปริมาณงานจำนวนมากได้

โดยรายละเอียดของแนวคิดการคำนวณแบบขนานในการแบ่งข้อมูลของอินพุตสำหรับทั้ง 2 โมเดลมีรูปแบบที่แตกต่างกันดังนี้

1. แนวคิดการคำนวณแบบขนานของการแบ่งข้อมูลของอินพุตสำหรับโมเดล SIA กล่าวคือ การแบ่งข้อมูลของอินพุตสำหรับโมเดล SIA ผู้วิจัยได้พิจารณาการแบ่งข้อมูลของอินพุตที่ตำแหน่งจำนวนของยานพาหนะ (Number of Vehicles) ที่ผู้ใช้ระบุ โดยแต่ละเครื่องคอมพิวเตอร์จะได้รับจำนวนของยานพาหนะที่แตกต่างกัน เพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ตัวอย่างเช่น เมื่อผู้ใช้ระบุจำนวนของยานพาหนะมีค่าเท่ากับ 25 ระบบจะทำการกระจายจำนวนของยานพาหนะตั้งแต่ค่า 1 ถึง 25 ไปยังกลุ่มของเครื่องคอมพิวเตอร์ที่ใช้ประมวลผล (อินพุตของปัญหาเหมือนกันแต่ต่างกันที่จำนวนของยานพาหนะ) เป็นต้น และบนเครื่องคอมพิวเตอร์ที่ใช้ประมวลผลของแต่ละเครื่องสามารถประมวลผลแบบมัลติเทรด โดยการส่งงานเข้าสู่ระบบการคำนวณแบบมัลติเทรดตามแนวคิดแบบ CCR โดยที่ผลลัพธ์ของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จากแต่ละเครื่องคอมพิวเตอร์จะเป็นอิสระต่อกัน โดยแนวคิดของการคำนวณแบบขนานแบบ Message passing และ Multithreading สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สำหรับโมเดล SIA แสดงดังภาพที่ 37 และภาพที่ 38



ภาพที่ 37 แนวคิดการคำนวณแบบขนานแบบ Message passing สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งสำหรับโมเดล SIA



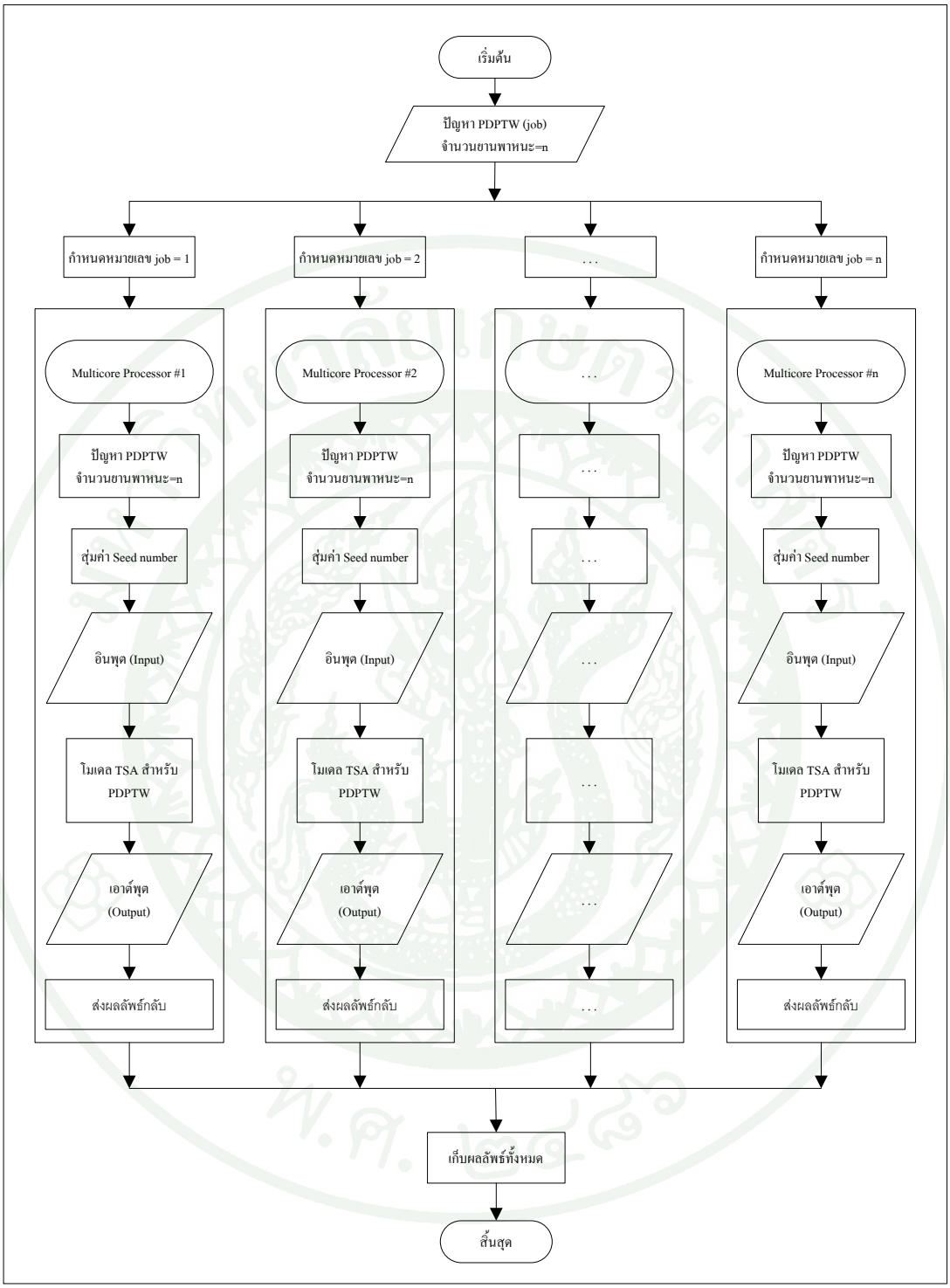
ภาพที่ 38 แนวคิดการคำนวณแบบขนานแบบ Multithreading ในรูปแบบของ CCR สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สำหรับโมเดล SIA บนแต่ละเครื่องคอมพิวเตอร์

2. แนวคิดการคำนวณแบบขนานของการแบ่งข้อมูลของอินพุตสำหรับโมเดล TSA กล่าวคือ การแบ่งข้อมูลของอินพุตสำหรับโมเดล TSA ผู้วิจัยได้พิจารณาการแบ่งข้อมูลของอินพุตโดยการกำหนดค่าของ Seed number ที่แตกต่างกัน โดยที่แต่ละเครื่องคอมพิวเตอร์จะใช้วิธีการสุ่ม (Random) ค่า Seed ขึ้นมาเพื่อใช้สำหรับนำไปสร้างคำตอบเริ่มต้น (Initial solution) ซึ่งการสร้างเซตคำตอบเริ่มต้นจากวิธีการสุ่มตามค่าเริ่มต้นของ Seed ซึ่งค่า Seed จะถูกสุ่มค่าขึ้นมาใหม่ทุกๆครั้งสำหรับแต่ละอินพุตที่เข้ามา ซึ่งทำให้ผลลัพธ์ที่ได้จากโมเดล TSA มีผลลัพธ์ที่แตกต่างกัน เพราะค่าเริ่มต้นของ

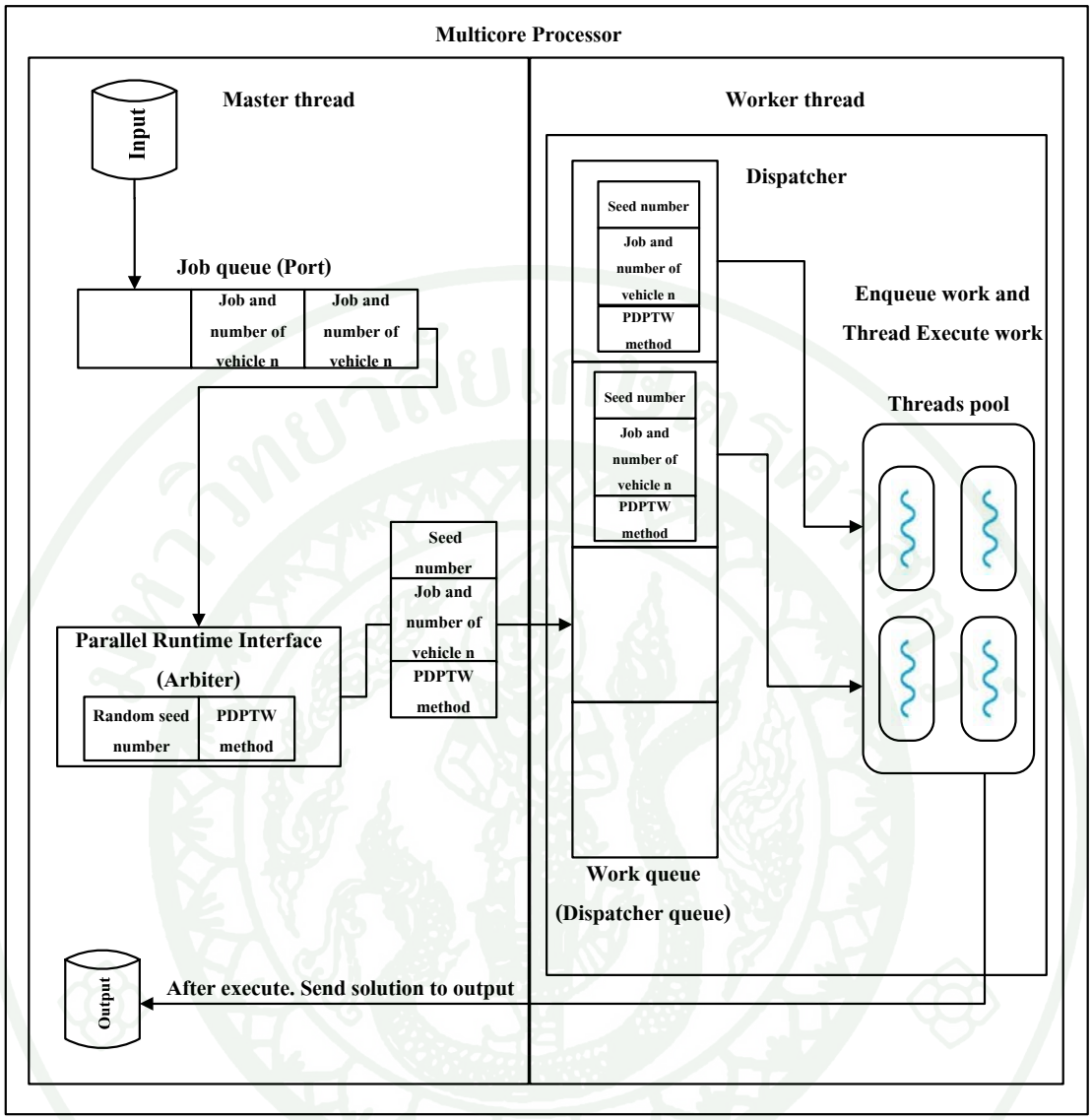
Seed number ต่างกัน และบนเครื่องคอมพิวเตอร์แต่ละเครื่องสามารถประมวลผลแบบมัลติเทรด โดยการส่งงานเข้าสู่ระบบการคำนวณแบบมัลติเทรดตามแนวคิดแบบ CCR โดยแนวคิดของการคำนวณแบบขนานแบบ Message passing และ Multithreading สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สำหรับโมเดล TSA แสดงดังภาพที่ 39 และภาพที่ 40

จากภาพที่ 38 และภาพที่ 40 แสดงการคำนวณแบบขนานแบบ Multithreading ในรูปแบบของ CCR มีกระบวนการทำงาน ดังนี้คือ

เมื่องาน (Job) การจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง และจำนวนของยานพาหนะ (Number of vehicle) ถูกส่งเข้ามาสู่แต่ละเครื่องคอมพิวเตอร์ งานที่เข้ามาจะถูกส่งไปยังคิวของงาน (Job queue) โดยคิวของงานนี้ได้ลงทะเบียนไปยัง Parallel Runtime Interface (Arbiter) เพื่อแจ้งให้ Arbiter ทราบถึงข้อมูลที่อยู่ในคิวของงาน ซึ่งภายใน Arbiter ประกอบด้วย การทำงานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง (PDPTW method) (กรณีระบบใช้โมเดล TSA ภายใน Arbiter จะมีการทำงานของการสุ่มค่า Seed (Random seed number) เพิ่มเข้ามา ดังภาพที่ 39) โดย Arbiter ทำหน้าที่ในการนำงานจากคิวของงาน (จำนวนงานที่นำออกจากคิวของงานจะขึ้นกับนโยบายที่กำหนดไว้) และการทำงานต่างๆสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง นำเข้าสู่คิวการทำงาน (Work queue) คิวการทำงานนี้เป็นคิวที่ใช้สำหรับรับงานและการทำงานต่างๆ เพื่อเตรียมพร้อมสำหรับการประมวลผล ซึ่งในการนำงานและการทำงานต่างๆ ออกจากคิวการทำงานนั้นเป็นหน้าที่ของ Dispatcher โดย Dispatcher จะทำการเลือกเทรดจากกลุ่มเทรด (Thread pool) นำไปจับคู่กับงานและการทำงานต่างๆ ที่ Dispatcher ได้ทำการเลือกออกมาจากคิวการทำงาน ถึงตรงนี้เทรดมีหน้าที่การทำงาน คือ การทำงานสำหรับแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง (PDPTW method) และข้อมูลที่ใช้ในการทำงาน คือ งานที่ถูกเลือกจากคิวการทำงาน ต่อจากนั้นเทรดจะทำการประมวลผลงานที่ได้รับมอบหมาย หลังจากที่เทรดทำงานเสร็จสมบูรณ์แล้ว ผลลัพธ์ของการแก้ปัญหาจะถูกส่งไปยังเอาต์พุต ต่อจากนั้นเทรดจะกลับเข้าสู่สถานะรอในกลุ่มของเทรดเพื่อรอรับงานชิ้นต่อไป ซึ่งการทำงานของแต่ละเทรดสามารถทำงานได้พร้อมๆ กัน



ภาพที่ 39 แนวคิดการคำนวณแบบขนานแบบ Message passing สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งสำหรับโมเดล TSA

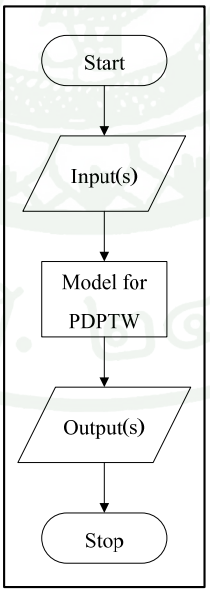


ภาพที่ 40 แนวคิดการคำนวณแบบขนานแบบ Multithreadingในรูปแบบของ CCR สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สำหรับโมเดล TSA บนแต่ละเครื่องคอมพิวเตอร์

ผลและวิจารณ์

ซอฟต์แวร์ที่พัฒนาขึ้นตามกรอบการทำงานแบบสมรรถนะสูง

ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เป็นปัญหาการให้บริการการขนส่งสินค้าจากการร้องขอจากจุดรับ/ส่งสินค้า โดยใช้ยานพาหนะที่มีอยู่จำกัด ซึ่งแต่ละการร้องขอจะมีความเกี่ยวข้องกับการเคลื่อนย้าย โดยวัตถุประสงค์ของการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง คือ การใช้จำนวนยานพาหนะได้อย่างมีประสิทธิภาพ ซึ่งผลลัพธ์ที่ได้จากการแก้ปัญหา คือ เส้นทางของยานพาหนะแต่ละคันที่ต้องรับผิดชอบต่อการรับและส่งสินค้า และนำผลลัพธ์ของการแก้ปัญหาไปใช้ต่อไป โดยทั่วไปผู้ใช้งานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง มีการระบุรายละเอียดอินพุต (Input) ของปัญหาจากนั้นจะนำอินพุตของปัญหาเข้าสู่โมเดลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เพื่อให้ได้เอาต์พุต (Output) ที่ได้รับจากโมเดลของปัญหาดังกล่าว เพื่อนำไปใช้สำหรับการจัดเส้นทางของยานพาหนะจริง ลักษณะทั่วไปของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง แสดงดังภาพที่ 41



ภาพที่ 41 ลักษณะทั่วไปของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

รายละเอียดแต่ละส่วนประกอบของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง (งานวิจัยนี้ใช้แบบฟอร์มเอกสารเอ็กเซลที่ทางระบบจัดเตรียมไว้สำหรับรับอินพุตและแสดงเอาต์พุตผ่านทางแบบฟอร์มนี้) แสดงดังต่อไปนี้

อินพุต (Input)

รายละเอียดและรูปแบบอินพุตของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ผู้ใช้ระบุเพื่อนำอินพุตเข้าสู่โมเดลของการแก้ปัญหา แสดงดังภาพที่ 42 ซึ่งประกอบด้วยข้อมูลดังนี้

Number of Customers หมายถึง จำนวนร้านค้าทั้งหมด

Number of Vehicles หมายถึง จำนวนของยานพาหนะทั้งหมด

Capacity หมายถึง จำนวนความจุของยานพาหนะแต่ละคัน

Speed หมายถึง ความเร็วของยานพาหนะแต่ละคัน

Customer หมายถึง หมายเลขลำดับของร้านค้า โดย 0 หมายถึงคลังพัสดุ (Depot) สำหรับเป็นจุดปล่อยรถ กรณีอื่นๆ เป็นร้านค้า

X Coordination หมายถึง ตำแหน่งที่ตั้งของร้านค้าในแนวแกน X

Y Coordination หมายถึง ตำแหน่งที่ตั้งของร้านค้าในแนวแกน Y

Demand/Load หมายถึง ความต้องการในการบรรทุกสินค้า (Pickup) หรือส่งสินค้า (Delivery) ซึ่งค่า เป็น 0 หมายถึง จุดปล่อยรถ, ค่าเป็นบวก หมายถึง จุดบรรทุกสินค้า และค่าเป็นลบ หมายถึง จุดส่งสินค้า

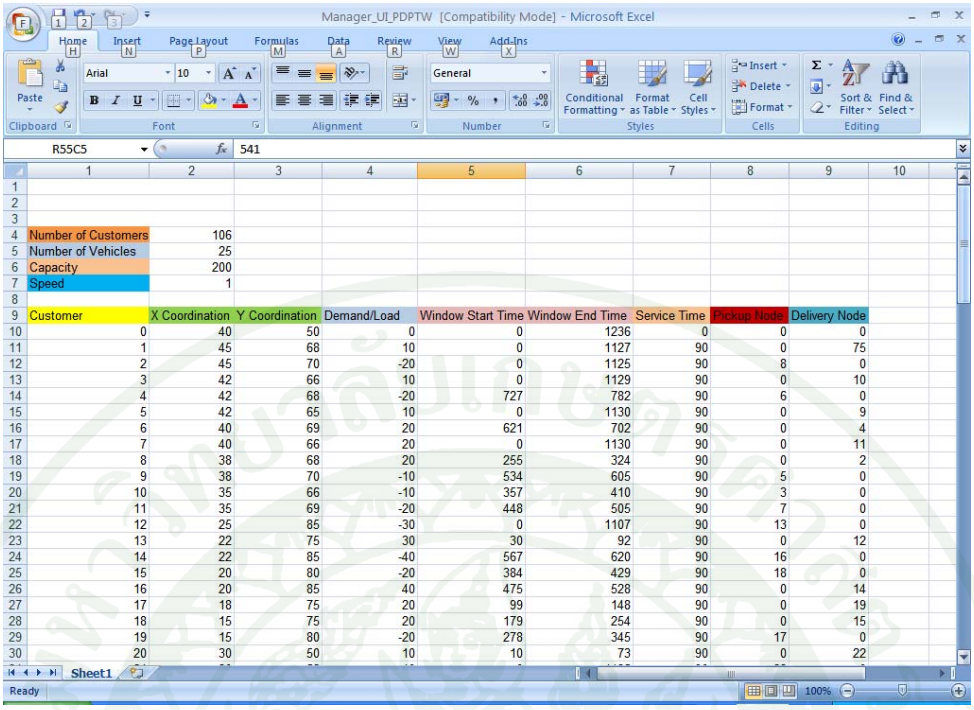
Window Start Time หมายถึง เวลาเปิดรับ/ส่งสินค้าของร้านค้า

Window End Time หมายถึง เวลาปิดรับ/ส่งสินค้าของร้านค้า

Service Time หมายถึง ระยะเวลาของการบริการ (การรับ/ส่งสินค้า)

Pickup Node หมายถึง จุดรับสินค้าซึ่งค่าเป็น 0 หมายถึง ไม่ใช่จุดรับสินค้าและกรณีอื่นๆ หมายถึง หมายเลขของร้านค้าที่ต้องไปรับสินค้า

Delivery Node หมายถึง จุดส่งสินค้าซึ่งค่าเป็น 0 หมายถึง ไม่ใช่จุดส่งสินค้าและกรณีอื่นๆ หมายถึง หมายเลขของร้านค้าที่ต้องไปส่งสินค้า



Number of Customers	106	
Number of Vehicles	25	
Capacity	200	
Speed	1	

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

ภาพที่ 42 รายละเอียดอินพุตของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

เอาต์พุต (Output)

รายละเอียดเอาต์พุตของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ผู้ใช้งานได้รับหลังจากผ่านโมเดลของการแก้ปัญหาเมื่อการแก้ปัญหาเสร็จสิ้น โดยแสดงรายการของร้านค้าต่างๆที่ยานพาหนะต้องให้บริการ แสดงดังภาพที่ 43 ซึ่งประกอบด้วยข้อมูลดังนี้

Index Vehicle หมายถึง ลำดับของยานพาหนะ

Visited Node หมายถึง หมายเลขลำดับของร้านค้า

Pickup at หมายถึง หมายเลขลำดับของร้านค้าที่ไปรับสินค้า กรณีค่าเป็น 0 หมายถึง ไม่มีการรับสินค้า

Delivery at หมายถึง หมายเลขลำดับของร้านค้าที่ไปส่งสินค้า กรณีค่าเป็น 0 หมายถึง ไม่มีการส่งสินค้า

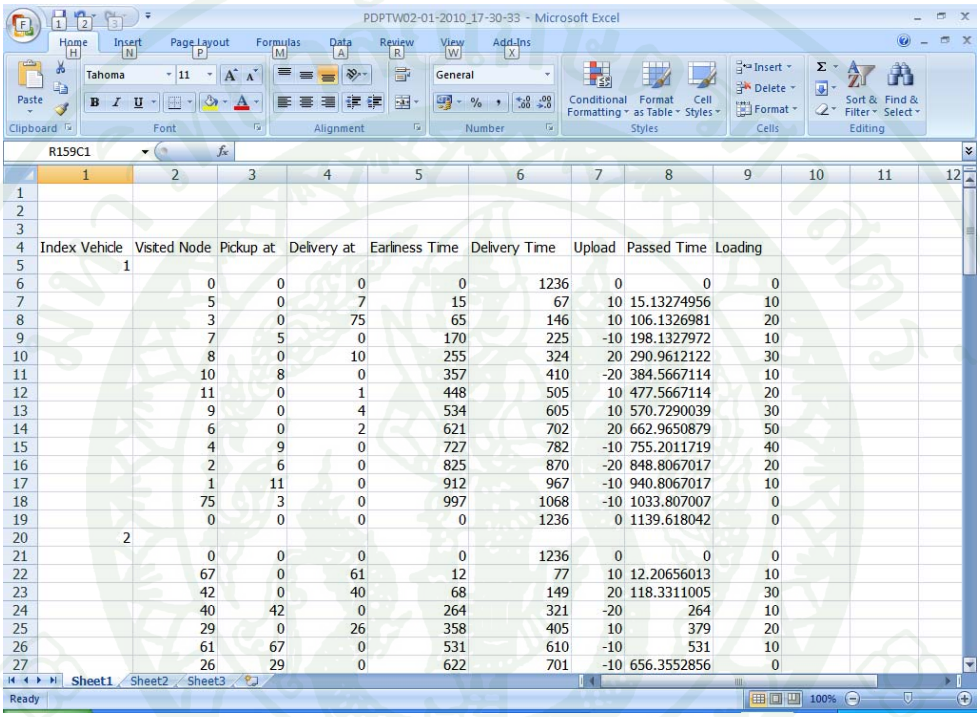
Earliness Time หมายถึง เวลาเปิดรับ/ส่งสินค้าของร้านค้า

Delivery Time หมายถึง เวลาปิดรับ/ส่งสินค้าของร้านค้า

Upload หมายถึง ปริมาณน้ำหนักของสินค้า กรณีค่าเป็นบวก หมายถึง ปริมาณน้ำหนักของสินค้าที่ขนขึ้นยานพาหนะ กรณีเป็นลบ หมายถึง ปริมาณน้ำหนักของสินค้าที่ขนลงจากยานพาหนะ

Passed Time หมายถึง เวลาทั้งหมดที่ยานพาหนะใช้ไป

Loading หมายถึง ปริมาณน้ำหนักขณะที่อยู่บนยานพาหนะ



	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4	Index Vehicle	Visited Node	Pickup at	Delivery at	Earliness Time	Delivery Time	Upload	Passed Time	Loading			
5	1											
6		0	0	0	0	1236	0	0	0			
7		5	0	7	15	67	10	15.13274956	10			
8		3	0	75	65	146	10	106.1326981	20			
9		7	5	0	170	225	-10	198.1327972	10			
10		8	0	10	255	324	20	290.9612122	30			
11		10	8	0	357	410	-20	384.5667114	10			
12		11	0	1	448	505	10	477.5667114	20			
13		9	0	4	534	605	10	570.7290039	30			
14		6	0	2	621	702	20	662.9650879	50			
15		4	9	0	727	782	-10	755.2011719	40			
16		2	6	0	825	870	-20	848.8067017	20			
17		1	11	0	912	967	-10	940.8067017	10			
18		75	3	0	997	1068	-10	1033.807007	0			
19		0	0	0	0	1236	0	1139.618042	0			
20	2											
21		0	0	0	0	1236	0	0	0			
22		67	0	61	12	77	10	12.20656013	10			
23		42	0	40	68	149	20	118.3311005	30			
24		40	42	0	264	321	-20	264	10			
25		29	0	26	358	405	10	379	20			
26		61	67	0	531	610	-10	531	10			
27		26	29	0	622	701	-10	656.3552856	0			

ภาพที่ 43 รายละเอียดเอาต์พุตของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

การทดสอบประสิทธิภาพ

งานวิจัยนี้ทำการทดสอบระบบการทำงานของ DPDPTWI โดยแบ่งการทดสอบออกเป็น 2 แบบ คือ 1) การทดสอบประสิทธิภาพการทำงานของระบบ DPDPTWI โดยใช้โมเดล SIA (แสดงดังภาพที่ 35) สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง และ 2) การทดสอบการทำงานของระบบ DPDPTWI โดยใช้โมเดล TSA (แสดงดังภาพที่ 36) สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งระบบ DPDPTWI ที่ใช้โมเดล TSA นี้ เป็นการพัฒนาต่อเนื่องจากระบบ DPDPTWI ที่ใช้โมเดล SIA ซึ่งโมเดล TSA นี้เป็นโมเดลที่ตอบสนองต่อความต้องการของผู้ใช้ได้ดีกว่าการใช้โมเดล SIA

การทดสอบประสิทธิภาพการทำงานของระบบ DPDPTWI ได้ทำการทดสอบวัดเวลาของการประมวลผลซ้ำ 3 ครั้ง และการวัดประสิทธิภาพของระบบการคำนวณแบบขนานวัด โดยใช้ Speedup และ Efficiency

รายละเอียดของการทดสอบระบบ DPDPTWI ที่ใช้โมเดลแต่ละแบบแสดงดังต่อไปนี้

การทดสอบประสิทธิภาพการทำงานของระบบ DPDPTWI โดยใช้โมเดล SIA สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

การทดสอบประสิทธิภาพการทำงานของระบบ DPDPTWI โดยใช้โมเดล SIA สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งนั้น ได้ทำการทดสอบประสิทธิภาพโดยแบ่งเป็น 2 การทดสอบ คือ

1. การทดสอบประสิทธิภาพบนเครื่องเวิร์คเกอร์ที่ประกอบด้วย 2 โปรเซสเซอร์ ซึ่งแต่ละโปรเซสเซอร์ประกอบด้วย 4 คอร์ (Two quad core processors) จำนวน 2 เวิร์คเกอร์

ตารางที่ 3 รายละเอียดของฮาร์ดแวร์และซอฟต์แวร์สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI

Machine	Hardware	Operating System	Software
1 Manager	AMD Athlon XP 2500+, 512 MB of RAM	Windows XP Professional	Manager, MySQL Server for Manager
2 Workers	Quad-core Intel Xeon 2.0 GHz, 1 GB of RAM	Windows XP Professional	Worker

การทดสอบประสิทธิภาพในหัวข้อนี้แบ่งการทดสอบเป็น 2 ส่วนคือ

1.1 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 1 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 1, 2, 4, 8, 16, 32, 64, 128, 256 และ 512 เทรด

1.2 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 2 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 2, 4, 8, 16, 32, 64, 128, 256 และ 512 เทรด ซึ่งสำหรับการประมวลผลปัญหาบนเวิร์คเกอร์ 2 เครื่องนี้ ทางผู้วิจัยใช้วิธีการแบ่งเทรดให้กับแต่ละเครื่องดังนี้

- วิธีการแบ่งจำนวนเทรดให้กับแต่ละเครื่องมีจำนวนเทรดเท่ากัน ซึ่งเรียกวิธีนี้ว่า Half-half strategy (h-h)

- วิธีการแบ่งจำนวนเทรดให้กับแต่ละเครื่อง โดยแบ่งจำนวนเทรดให้กับเครื่องลำดับที่หนึ่งเท่ากับจำนวนคอร์ของเครื่องและจำนวนเทรดที่เหลือแบ่งให้กับเครื่องลำดับที่สอง ซึ่งเรียกวิธีนี้ว่า Fill-spill strategy (f-s)

ผลการทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเวิร์คเกอร์ 1 เครื่องและ 2 เครื่อง แสดงดังนี้

ตารางที่ 4 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 1 เครื่อง

Problem sizes	Number of threads									
(customers)	1	2	4	8	16	32	64	128	256	512
212	0.33	0.16	0.09	0.05	0.05	0.05	0.05	0.05	0.06	0.06
530	16.93	8.53	4.34	2.28	2.26	2.29	2.40	2.41	2.43	2.54
848	146.29	78.59	43.61	31.64	24.63	19.25	19.47	19.69	19.86	20.09

จากผลการทดสอบตารางที่ 4 แสดงให้เห็นว่าเมื่อจำนวนของเทรดเพิ่มขึ้นทำให้เวลาที่ใช้ในการประมวลผลลดลง ซึ่งกรณีที่ดีที่สุดของแต่ละปัญหาจากการทดสอบเห็นได้ว่า

กรณีที่ดีที่สุดของปัญหาที่มีขนาดเท่ากับ 212 ลูกค้า (Customers) จากการคำนวณด้วย 1 เทรดใช้เวลา 0.33 นาที เหลือเพียง 0.05 นาที เมื่อใช้เทรดจำนวน 8 เทรด

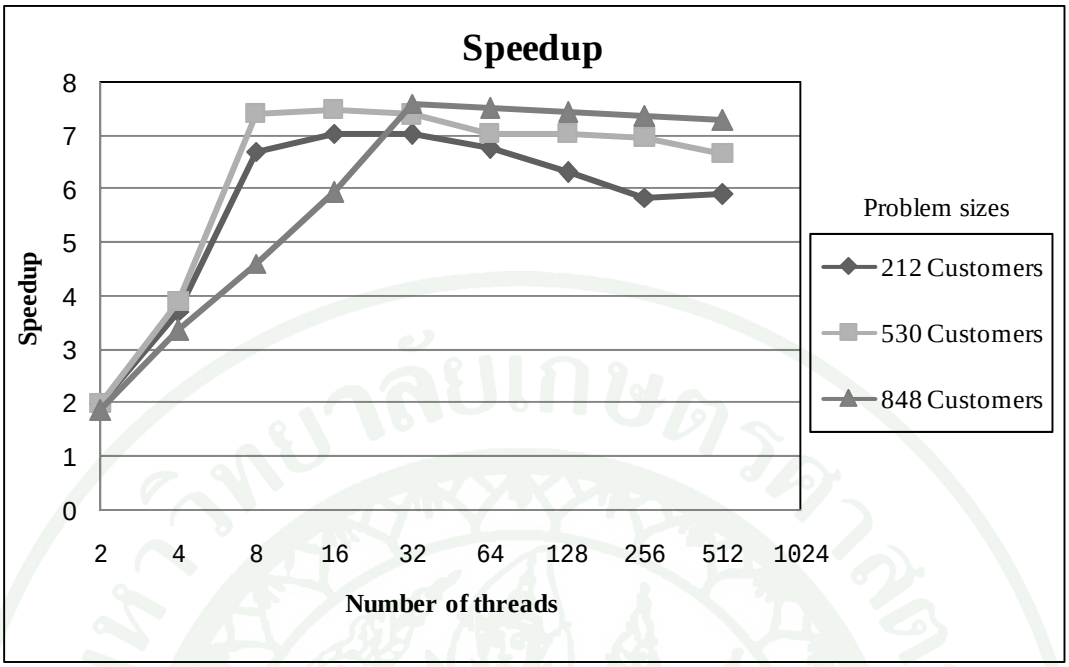
กรณีที่ดีที่สุดของปัญหาที่มีขนาดเท่ากับ 530 ลูกค้า จากการคำนวณด้วย 1 เทรดใช้เวลา 16.93 นาที เหลือเพียง 2.26 นาที เมื่อใช้เทรดจำนวน 16 เทรด

กรณีที่ดีที่สุดของปัญหาที่มีขนาดเท่ากับ 848 ลูกค้า จากการคำนวณด้วย 1 เทรดใช้เวลา 146.29 นาที เหลือเพียง 19.25 นาที เมื่อใช้เทรดจำนวน 32 เทรด

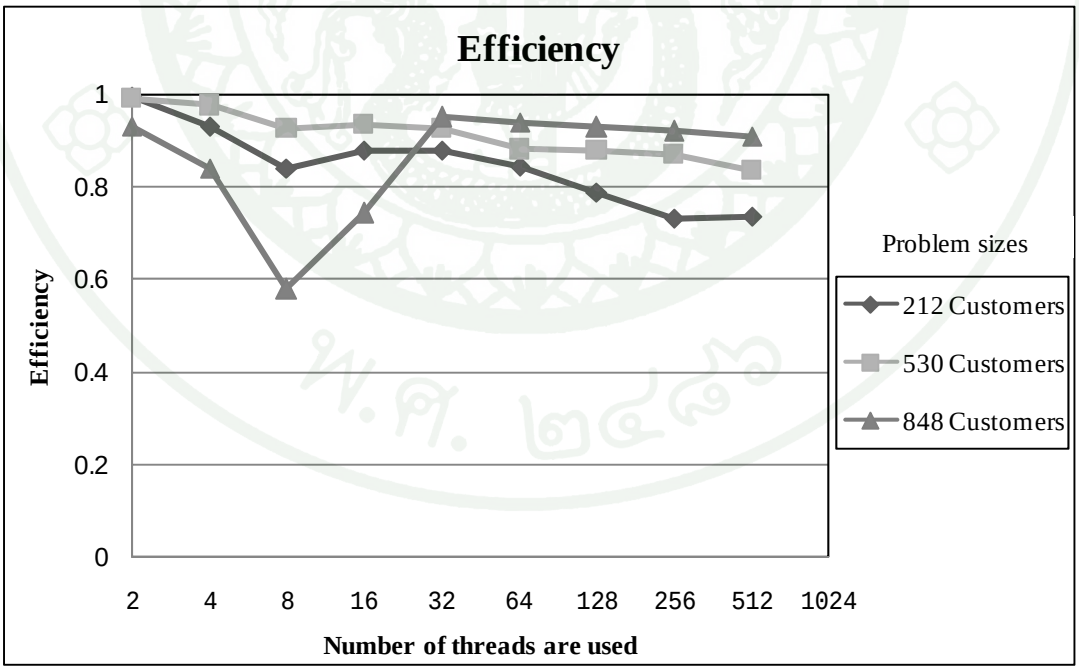
ตารางที่ 5 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 2 เครื่อง

Threads	Problem sizes (customers)					
	212		530		848	
	h-h	f-s	h-h	f-s	h-h	f-s
2	0.17	0.16	8.57	8.53	73.81	78.59
4	0.09	0.09	4.32	4.34	37.01	43.61
8	0.05	0.05	2.2	2.28	18.65	31.64
16	0.02	0.02	1.13	1.13	9.6	9.6
32	0.02	0.02	1.13	1.13	9.72	9.46
64	0.03	0.03	1.16	1.17	9.85	9.97
128	0.03	0.03	1.18	1.18	9.88	10.05
256	0.03	0.03	1.23	1.12	10.04	10.22
512	0.03	0.04	1.23	1.21	10.18	10.33

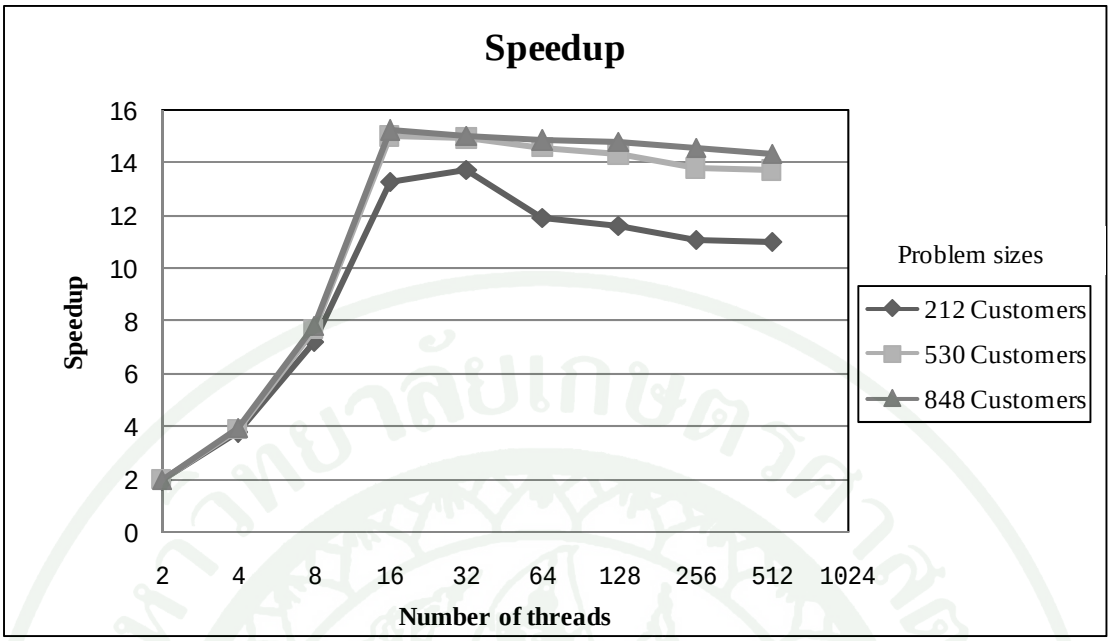
จากผลการทดสอบตารางที่ 5 แสดงให้เห็นว่า วิธีการแบ่งจำนวนเทรดให้กับแต่ละเครื่องและการเพิ่มจำนวนเทรดมีผลกระทบต่อเวลาที่ใช้ในการประมวลผล ซึ่งเห็นได้ว่าการแบ่งจำนวนเทรดให้กับแต่ละเวิร์คเกอร์แบบ Half-half (h-h) มีผลทำให้เวลาที่ใช้ในการประมวลผลลดลงมากกว่าการแบ่งจำนวนเทรดให้กับแต่ละเวิร์คเกอร์แบบ Fill-spill (f-s)



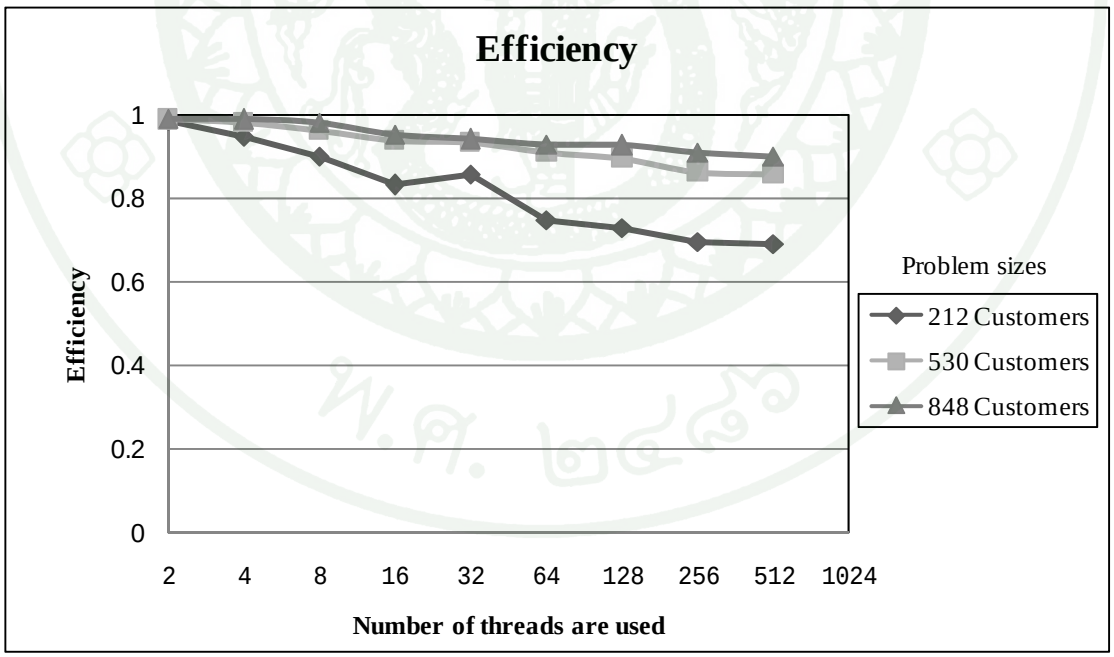
ภาพที่ 44 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์



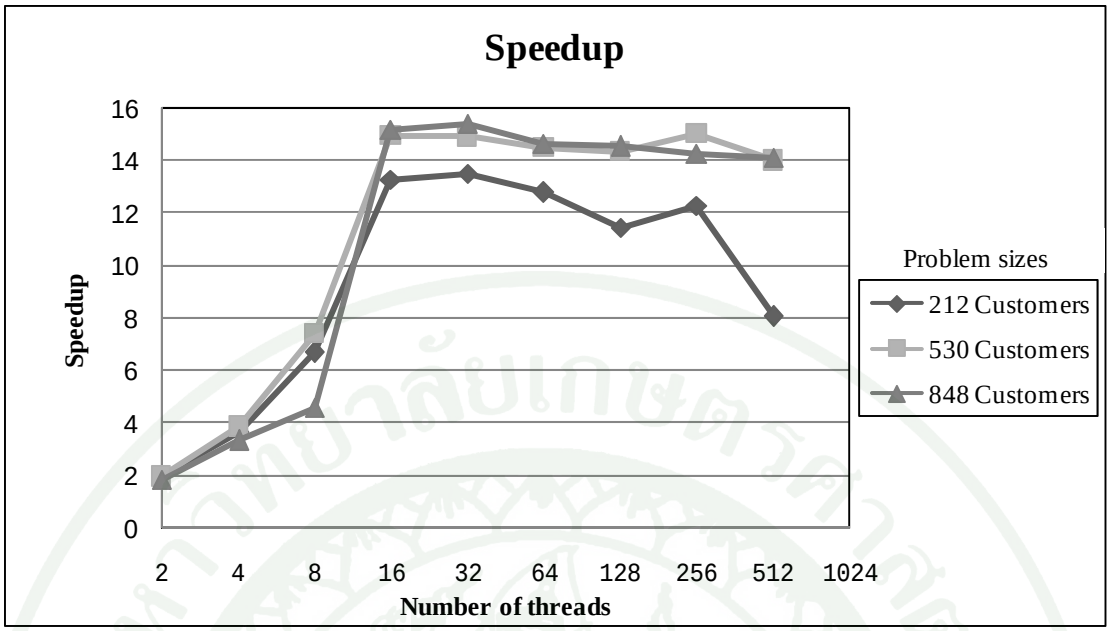
ภาพที่ 45 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์



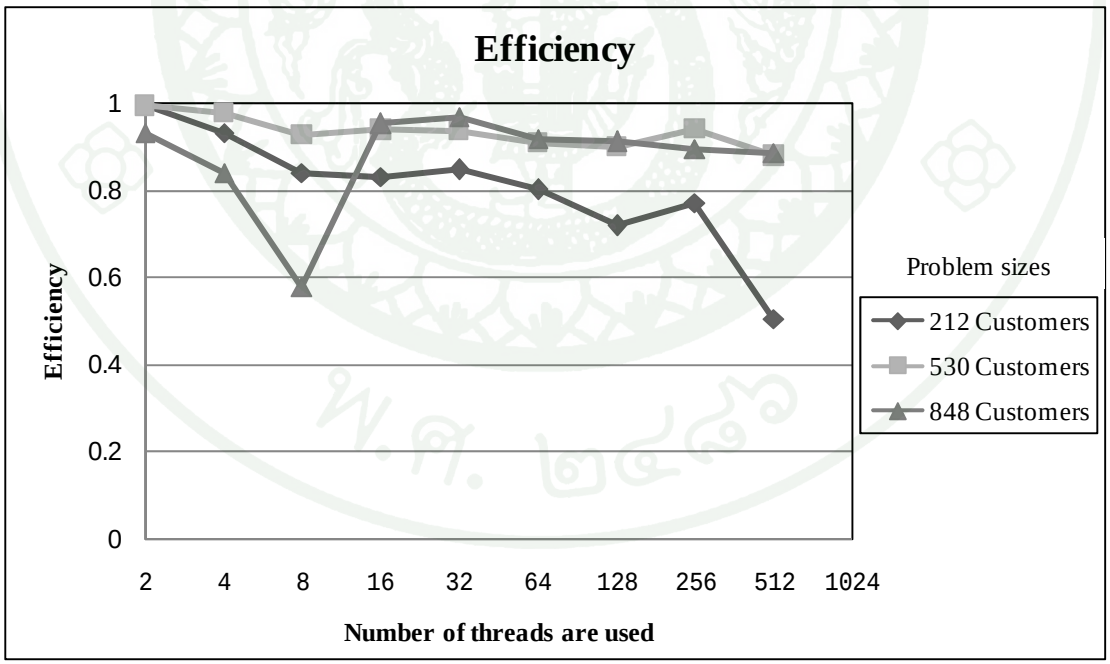
ภาพที่ 46 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์ที่ใช้วิธีการแบ่งจำนวนเทรดแบบ Half-half (h-h)



ภาพที่ 47 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์ที่ใช้วิธีการแบ่งจำนวนเทรดแบบ Half-half (h-h)



ภาพที่ 48 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์ที่ใช้วิธีการแบ่งจำนวนเทรดแบบ Fill-spill (f-s)



ภาพที่ 49 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์ที่ใช้วิธีการแบ่งจำนวนเทรดแบบ Fill-spill (f-s)

จากผลการทดสอบประสิทธิภาพของระบบ DPDPTWI ที่แสดงในข้างต้น สามารถวิเคราะห์ประสิทธิภาพของระบบ DPDPTWI ได้ดังต่อไปนี้

1. จากผลการทดสอบเวลาในการประมวลผลสำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์และ 2 เวิร์คเกอร์ ซึ่งแสดงดังตารางที่ 4 และตารางที่ 5 ตามลำดับ แสดงให้เห็นว่า สำหรับแต่ละขนาดของปัญหานั้นเวลาที่ใช้ในการประมวลผลมีค่าลดลง เมื่อจำนวนของเทรดและจำนวนของเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้น สาเหตุที่เป็นเช่นนี้สามารถแยกพิจารณาได้ 2 กรณีคือ

1.1 กรณีการประมวลผลบน 1 เวิร์คเกอร์มีการเพิ่มจำนวนของเทรดทำให้งานสามารถถูกแบ่งออกเป็นงานย่อยๆ ส่งผลให้งานนั้น ถูกกระจายไปประมวลผลยังแต่ละเทรดต่างๆ ที่เพิ่มขึ้นได้ในเวลาเดียวกัน ทำให้เวลาที่ใช้ในการประมวลผลมีค่าลดลง ซึ่งจากผลการทดสอบจากตารางที่ 4 แสดงให้เห็นว่า เมื่อจำนวนของเทรดเพิ่มขึ้นทำให้เวลาที่ใช้ในการประมวลผลลดลง ซึ่งกรณีที่ดีที่สุดจากการทดสอบ เห็นได้ว่า กรณีที่จำนวนของปัญหามีขนาดเท่ากับ 848 ลูกค้า จากการคำนวณด้วย 1 เทรดใช้เวลา 146.29 นาที เหลือเพียง 19.25 นาที เมื่อใช้เทรดจำนวน 32 เทรด อย่างไรก็ตาม การเพิ่มจำนวนเทรดก็ไม่ได้ ทำให้เวลาที่ใช้ในการประมวลผลมีค่าลดลงตามจำนวนของเทรดที่เพิ่มขึ้น (เวลาไม่ได้ลดลงเท่ากับจำนวนเท่าของเทรดที่เพิ่มขึ้น) เช่น จากผลการทดสอบพบว่า การเพิ่มจำนวนเทรดที่ใช้ในการประมวลผลเป็น 2 เทรด ส่งผลให้เวลาที่ใช้ในการประมวลผลมีค่าลดลงเกือบ 2 เท่าเมื่อเทียบกับเวลาที่ใช้ในการประมวลผลจาก 1 เทรด แต่เมื่อเพิ่มจำนวนเทรดที่ใช้ในการประมวลผลเป็น 16 เทรดกลับไม่ส่งผลให้เวลาที่ใช้ในการประมวลผลมีค่าใกล้เคียง 16 เท่าเมื่อเทียบกับเวลาที่ใช้ในการประมวลผลจาก 1 เทรด เป็นต้น โดยสาเหตุที่เป็นเช่นนี้เนื่องมาจาก 2 สาเหตุ คือ 1) งานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งประมวลผลบนเครื่องเวิร์คเกอร์ที่มีจำนวนคอร์เท่ากับ 8 คอร์ทำให้ประสิทธิภาพสูงสุดในการประมวลผลที่เครื่องเวิร์คเกอร์ทำได้คือ 8 เท่าเมื่อเทียบกับ 1 คอร์ (1 คอร์ทำงาน 1 เทรด) และ 2) งานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งมีงานบางส่วนที่ไม่สามารถประมวลผลแบบมัลติเทรดได้ (เป็นส่วนที่ทำการประมวลผลแบบลำดับ) ดังนั้น เวลาที่ใช้ในการประมวลผลงานตรงส่วนนี้จึงไม่สามารถลดลงได้ ด้วยการประมวลผลแบบมัลติเทรด และอีกสิ่งหนึ่งที่น่าสนใจ คือ Overhead ที่เกิดขึ้นจากการเพิ่มจำนวน เทรดมีผลกระทบต่อเวลาในการประมวลผล

1.2 กรณีการประมวลผลบน 2 เวิร์คเกอร์ และแต่ละเวิร์คเกอร์มีการเพิ่มจำนวนเทรด ทำให้เวลาที่ใช้ในการประมวลผลมีค่าลดลง ซึ่งจากผลการทดสอบจากตารางที่ 5 แสดงให้เห็นว่า จำนวนของเทรดเพิ่มขึ้นทำให้เวลาที่ใช้ในการประมวลผลลดลง กรณีที่ดีที่สุดจากการทดสอบ เห็นได้ว่า ที่จำนวนของปัญหามีขนาดเท่ากับ 848 ลูกค้า จากการคำนวณด้วย 1 เทรดใช้เวลา 146.29 นาที (ประมวลผลบน 1 เวิร์คเกอร์) เหลือเพียง 9.6 นาที เมื่อใช้เทรดจำนวน 16 เทรด (ประมวลผลบน 2 เวิร์คเกอร์โดยใช้วิธีการแบ่งจำนวนเทรดแบบ Half-half) ซึ่งจากการทดสอบจากตารางที่ 5 แสดงให้เห็นว่าวิธีการแบ่งจำนวนเทรดมีผลต่อประสิทธิภาพโดยรวมของการประมวลผล และสิ่งหนึ่งที่น่าสนใจจากการทดสอบนี้คือ Overhead ที่เกิดจากการประมวลผลบนเครื่องเวิร์คเกอร์ที่มากกว่า 1 เครื่อง กล่าวคือ จากตารางที่ 4 และตารางที่ 5 เมื่อนำมาเปรียบเทียบผลการทดสอบของทั้ง 2 ตาราง (เปรียบเทียบด้วยจำนวนเทรดที่เท่ากัน) จะเห็นได้ว่า กรณีที่ปัญหามีขนาด 848 ลูกค้า เวลาของการประมวลผลตั้งแต่ 2 ถึง 512 เทรดบน 1 เวิร์คเกอร์ใช้เวลาประมวลผลมากกว่าเวลาของการประมวลผลตั้งแต่ 2 ถึง 512 เทรดบน 2 เวิร์คเกอร์ (วิธีการแบ่งเทรดแบบ Half-half) ซึ่งแสดงให้เห็นว่า เมื่อปัญหามีขนาดเพิ่มมากขึ้นมีผลทำให้เวลาที่ใช้ในการประมวลผลสูงขึ้น ซึ่งการแบ่งปัญหาไปยังแต่ละเวิร์คเกอร์ เพื่อช่วยกันประมวลผลทำให้สามารถลดระยะเวลาในการประมวลผลได้ ถึงแม้ว่า กระบวนการของการกระจายปัญหาไปยังแต่ละเวิร์คเกอร์จะมีเวลาของการส่งงานไปยังเวิร์คเกอร์เพิ่มเข้ามาแต่เวลาโดยรวมของการประมวลผลงานก็ยังน้อยกว่าการประมวลผลงานบน 1 เวิร์คเกอร์

2. การประเมินประสิทธิภาพของระบบการคำนวณแบบขนานด้วย Speedup ซึ่งเป็นการบ่งบอกถึงความสามารถในการประมวลผลแบบขนานได้เร็วขึ้นเป็นกี่เท่า เมื่อเทียบกับการประมวลผลแบบลำดับ ซึ่ง Speedup เป็นอัตราส่วนระหว่างเวลาที่ใช้ในการประมวลผลแบบลำดับต่อเวลาที่ใช้ในการประมวลผลแบบขนาน โดยจากผลการทดสอบเวลาในการประมวลผลงานจากตารางที่ 4 และตารางที่ 5 สามารถคำนวณหา Speedup ซึ่งผลการทดสอบ Speedup ของแต่ละการทดสอบแสดงดังนี้

2.1 ผลการทดสอบ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์ แสดงดังภาพที่ 44 ซึ่งแสดงให้เห็นว่าสำหรับแต่ละขนาดปัญหาของงานนั้น Speedup มีค่าเพิ่มขึ้น เมื่อจำนวนเทรดที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้นจนกระทั่งจำนวนเทรดมีค่าเท่ากับ 32 เทรด ซึ่งให้ค่า Speedup สูงสุด หลังจากนั้นค่า Speedup จะมีค่าลดลงอย่างช้าๆ เมื่อเพิ่มจำนวนเทรดเกิน 32 เทรด ซึ่งจากผลการทดสอบ เห็นได้ว่า ในกรณีที่ดีที่สุดจากการทดสอบนั้น Speedup สำหรับการประมวลผลของงานที่มีขนาดปัญหาเท่ากับ 848 ลูกค้า โดยใช้จำนวนเทรดในการประมวลผล

32 เทรดนั้น มีค่าถึง 7.60 เท่า เมื่อเทียบกับการประมวลผลจาก 1 เทรด อย่างไรก็ตามถึงแม้ว่า Speedup มีค่าเพิ่มขึ้น เมื่อจำนวนเทรดที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้น แต่ Speedup ที่มีค่าเพิ่มขึ้นนั้นไม่ได้มีค่าเพิ่มขึ้นตามจำนวนเทรดที่ใช้ในการประมวลผลที่เพิ่มขึ้น และ Speedup มีอัตราการเพิ่มที่ลดลงเมื่อเทียบกับจำนวนเทรดที่ใช้ในการประมวลผลที่มีจำนวนเพิ่มขึ้น ซึ่งสาเหตุก็เนื่องจากงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง มีงานบางส่วนที่ไม่สามารถประมวลผลแบบมัลติเทรดได้ และการประมวลผลแบบมัลติเทรดมี Overhead ที่เกิดขึ้นจากจำนวนเทรดที่ใช้ในการประมวลผลเพิ่มขึ้น

2.2 ผลการทดสอบ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์ ซึ่งแบ่งเป็นผลการทดสอบ 2 ผลการทดสอบ คือ ผลการทดสอบ Speedup โดยใช้วิธีการแบ่งจำนวนเทรดแบบ Half-half และผลการทดสอบ Speedup โดยใช้วิธีการแบ่งจำนวนเทรดแบบ Fill-spill ซึ่งแสดงผลการทดสอบดังภาพที่ 46 และภาพที่ 48 ตามลำดับ

จากภาพที่ 46 และภาพที่ 48 แสดงให้เห็นว่า เมื่อจำนวนเทรดที่ใช้ในการประมวลผลเพิ่มขึ้นจนถึงจุดหนึ่ง ทำให้ Speedup มีค่าสูงสุดและเมื่อทำการเพิ่มจำนวนเทรดมากกว่าเดิมก็ไม่สามารถทำให้ค่าของ Speedup มีค่าเพิ่มขึ้นได้ ตัวอย่างเช่น จากผลการทดสอบภาพที่ 46 กรณีที่ปัญหามีขนาด 212 ลูกค้าที่ทำการเพิ่มจำนวนของเทรดจาก 32 เทรดเป็น 64 เทรด ทำให้ Speedup มีค่าลดลง เป็นต้น เนื่องจากในกรณีเช่นนี้ Overhead ที่เกิดจากการเพิ่มจำนวนเทรดที่ใช้ในการประมวลผลมีค่าเพิ่มขึ้นมากเกินไป ซึ่งเมื่อเปรียบเทียบกับเวลาที่ใช้ในการประมวลผลของงานนั้น มีความแตกต่างกันน้อยมาก จึงส่งผลให้ Speedup มีค่าไม่เพิ่มขึ้นแต่กลับมีค่าลดลงเมื่อจำนวนของเทรดเพิ่มขึ้น และสิ่งหนึ่งที่น่าสนใจจากผลการทดสอบ คือ Speedup ของการประมวลผลสำหรับงานที่มีขนาดปัญหาใหญ่กว่า มีค่าสูงกว่า Speedup ของการประมวลผลสำหรับงานที่มีขนาดปัญหาเล็กกว่า เมื่อจำนวนเทรดที่ใช้ในการประมวลผลคงที่ เนื่องจากงานที่มีขนาดปัญหาใหญ่กว่า มีอัตราส่วนระหว่างเวลาที่ใช้ในการประมวลผลจริงต่อ Overhead สูงกว่างานที่มีขนาดปัญหาเล็กกว่า จึงส่งผลให้ Speedup มีค่ามากกว่าทำให้กล่าวได้ว่า ระบบสามารถทำงานได้ Speedup ที่สูงขึ้น เมื่องานมีขนาดปัญหาที่ใหญ่ขึ้น

3. การประเมินประสิทธิภาพของระบบการคำนวณแบบขนานด้วย Efficiency โดย Efficiency เป็นค่าที่แสดงให้เห็นว่าระบบการคำนวณแบบขนานถูกใช้ไปกับการประมวลผลแบบขนาน เพื่อแก้ปัญหางานนั้นได้เต็มประสิทธิภาพหรือไม่ (เปอร์เซ็นต์) เนื่องจากการประมวลผลแบบขนานมี Overhead อีกทั้ง มีงานบางส่วนที่ไม่สามารถประมวลผลแบบขนานได้ ดังนั้นทำให้ระบบการคำนวณแบบขนานจึงไม่สามารถทำการประมวลผลแบบขนาน เพื่อแก้ปัญหางานนั้นได้เต็มประสิทธิภาพ

(100 เปอร์เซ็นต์) โดย Efficiency นั้นหาได้จากค่า Speedup หารด้วยจำนวนโปรเซสเซอร์ (จำนวนคอร์) ที่ใช้ในการประมวลผล และมีค่าอยู่ระหว่าง 0 ถึง 1 (หากคิดเป็นเปอร์เซ็นต์จะคูณด้วย 100) โดยระบบการคำนวณแบบขนานที่มีประสิทธิภาพที่ดีนั้น Efficiency มีค่าเข้าใกล้ 1 ซึ่งจากผลการทดสอบ Efficiency สามารถแสดงดังภาพที่ 45, ภาพที่ 47 และภาพที่ 49 แสดงให้เห็นว่า สำหรับแต่ละขนาดปัญหางานนั้น Efficiency มีค่าสูงเมื่อจำนวนเทรดที่ใช้ในการประมวลผลมีจำนวนน้อย เพราะ Overhead มีค่าต่ำ และค่า Efficiency มีอัตราการลดลงที่เพิ่มขึ้น เมื่อเทรดที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้น เนื่องจาก Overhead ที่มีค่าเพิ่มขึ้นตามจำนวนเทรดที่ใช้ในการประมวลผลที่เพิ่มขึ้น โดยจากผลการทดสอบยังพบว่า Efficiency ของการประมวลผลสำหรับงานที่มีขนาดปัญหาใหญ่มีค่าสูงกว่า Efficiency ของการประมวลผลสำหรับงานที่มีขนาดปัญหาเล็ก เมื่อจำนวนเทรดที่ใช้ในการประมวลผลคงที่

หมายเหตุ ผลการทดสอบ Efficiency ของภาพที่ 45 และภาพที่ 49 กรณีปัญหามีขนาด 848 ลูกค้านั้น ภาพที่ 45 ที่ช่วงจำนวนเทรดจาก 2 ถึง 16 เทรด และภาพที่ 49 ช่วงจำนวนเทรดจาก 2 ถึง 8 เทรด ค่าของ Efficiency ของปัญหาที่มีขนาดใหญ่กว่าให้ค่าต่ำกว่า Efficiency ของปัญหาที่มีขนาดเล็กกว่าซึ่งอาจเกิดความผิดพลาดในการทดสอบระบบโดยการทดสอบประสิทธิภาพในหัวข้อถัดไป (การทดสอบประสิทธิภาพบนเครื่องเวิร์คเกอร์ที่ประกอบด้วย 4 โปรเซสเซอร์จำนวน 16 เวิร์คเกอร์) เป็นข้อพิสูจน์ถึงความผิดพลาดของการทดสอบนี้

2. การทดสอบประสิทธิภาพบนเครื่องเวิร์คเกอร์ที่ประกอบด้วย 4 โปรเซสเซอร์ (Four processors) จำนวน 16 เวิร์คเกอร์

ตารางที่ 6 รายละเอียดของฮาร์ดแวร์และซอฟต์แวร์สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI

Machine	Hardware	Operating System	Software
1 Manager	Intel(R) Xeon(TM) 3.2 GHz, 8 GB of RAM	Windows HPC Server 2003	Manager, MySQL Server for Manager
16 Workers	Intel(R) Xeon(TM) 3.2 GHz, 8 GB of RAM	Windows HPC Server 2003	Worker

การทดสอบประสิทธิภาพในหัวข้อนี้แบ่งการทดสอบเป็น 5 ส่วนคือ

2.1 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 1 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 และ 1024 เทรด

2.2 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 2 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 2, 4, 8, 16, 32, 64, 128, 256, 512 และ 1024 เทรด ซึ่งการแบ่งจำนวนเทรดให้กับแต่ละเวิร์คเกอร์จะใช้วิธี Half-half

2.3 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 4 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 4, 8, 16, 32, 64, 128, 256, 512, 1024 และ 2048 เทรด ซึ่งการแบ่งจำนวนเทรดให้กับแต่ละเวิร์คเกอร์จะใช้วิธี Half-half

2.4 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 8 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 8, 16, 32, 64, 128, 256, 512, 1024, 2048 และ 4096 เทรด ซึ่งการแบ่งจำนวนเทรดให้กับแต่ละเวิร์คเกอร์จะใช้วิธี Half-half

2.5 การทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเครื่องเวิร์คเกอร์ 16 เครื่อง โดยการประมวลผลปัญหาดังกล่าวด้วยจำนวนเทรดที่แตกต่างกัน คือ 16, 32, 64, 128, 256, 512, 1024 และ2048 เทรด ซึ่งการแบ่งจำนวนเทรดให้กับแต่ละเวิร์คเกอร์จะใช้วิธี Half-half

ผลการทดสอบประสิทธิภาพของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง บนเวิร์คเกอร์ 1, 2, 4, 8 และ 16 เครื่อง แสดงดังนี้

ตารางที่ 7 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 1 เครื่อง

Problem sizes	Number of threads										
(customers)	1	2	4	8	16	32	64	128	256	512	1024
212	0.42	0.21	0.11	0.11	0.11	0.11	0.11	0.12	0.18	0.21	0.24
530	19.68	9.88	5.01	5.04	5.05	5.05	5.16	5.32	5.37	5.41	5.47
848	158.72	79.7	40.73	40.34	40.38	40.39	40.7	40.91	41.21	41.34	41.21

ตารางที่ 8 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 2 เครื่อง

Problem sizes	Number of threads									
(customers)	2	4	8	16	32	64	128	256	512	1024
212	0.22	0.11	0.06	0.06	0.06	0.06	0.06	0.06	0.09	0.1
530	9.98	5	2.54	2.54	2.56	2.58	2.68	2.73	2.71	2.74
848	79.96	40.23	20.32	20.36	20.37	20.41	20.46	20.55	20.84	20.85

ตารางที่ 9 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 4 เครื่อง

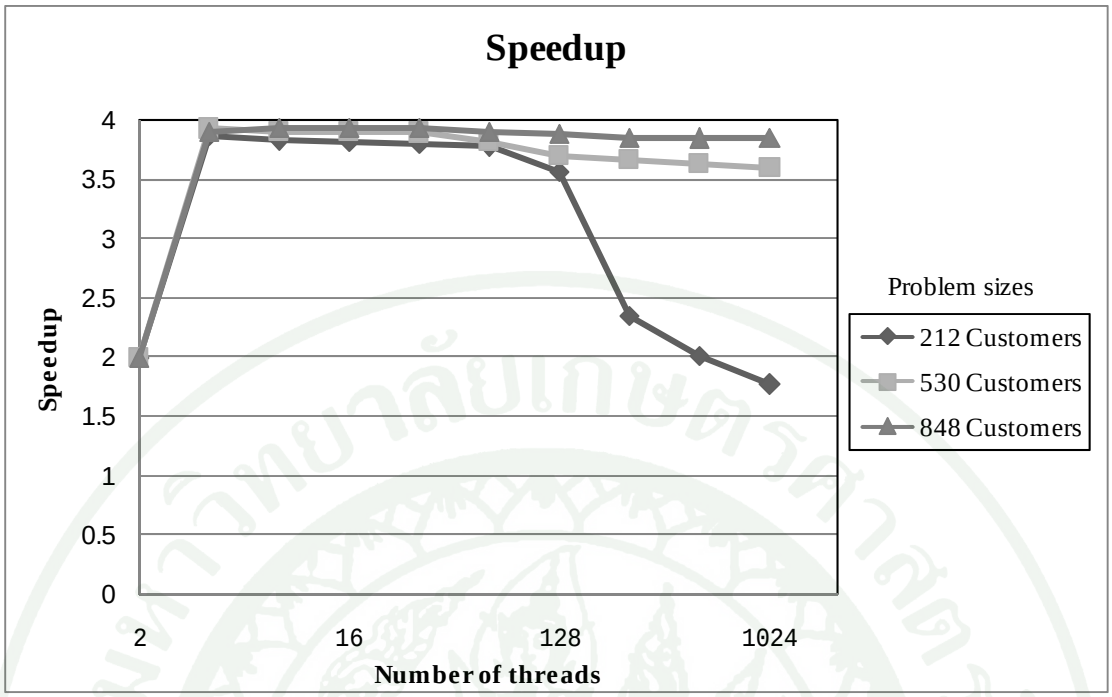
Problem sizes	Number of threads									
(customers)	4	8	16	32	64	128	256	512	1024	2048
212	0.11	0.06	0.03	0.03	0.03	0.03	0.04	0.04	0.03	0.03
530	5.1	2.56	1.31	1.3	1.3	1.31	1.35	1.39	1.42	1.43
848	40.65	20.37	10.3	10.29	10.29	10.34	10.42	10.48	10.53	10.61

ตารางที่ 10 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 8 เครื่อง

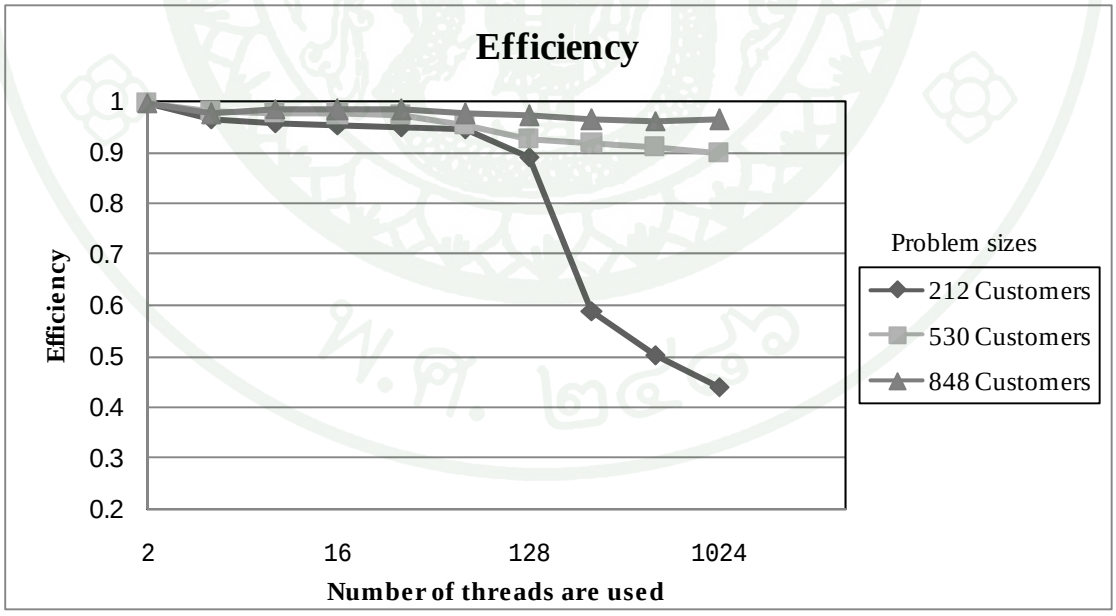
Problem sizes	Number of threads									
(customers)	8	16	32	64	128	256	512	1024	2048	4096
212	0.06	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03
530	2.65	1.34	0.68	0.68	0.68	0.69	0.7	0.71	0.73	0.76
848	20.88	10.67	5.29	5.29	5.3	5.31	5.35	5.39	5.47	5.44

ตารางที่ 11 เวลา (นาที) ในการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ของระบบ DPDPTWI บนเวิร์คเกอร์ 16 เครื่อง

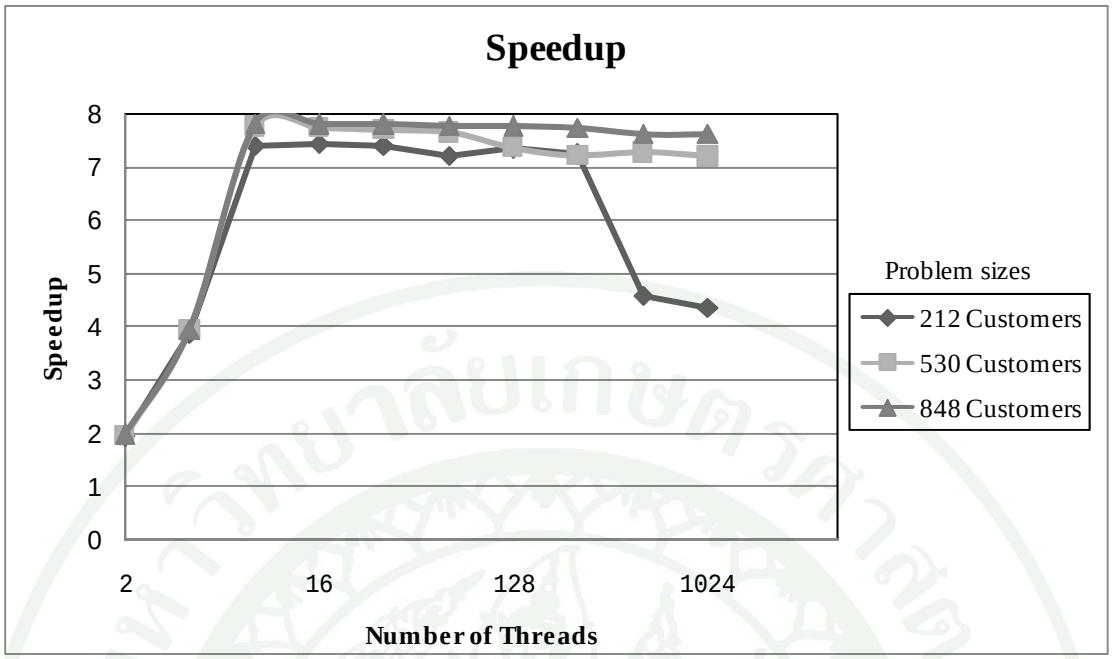
Problem sizes	Number of threads							
(customers)	16	32	64	128	256	512	1024	2048
212	0.04	0.02	0.01	0.01	0.01	0.02	0.01	0.01
530	1.44	0.73	0.37	0.37	0.37	0.38	0.38	0.4
848	11.02	5.54	2.83	2.81	2.8	2.82	2.88	2.82



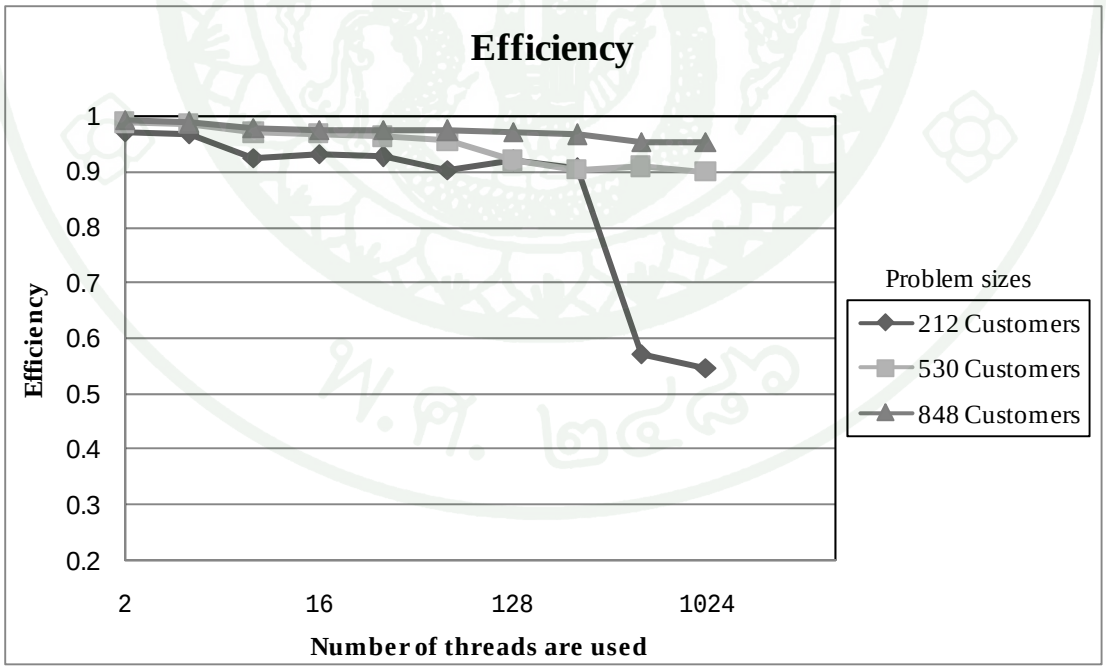
ภาพที่ 50 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์



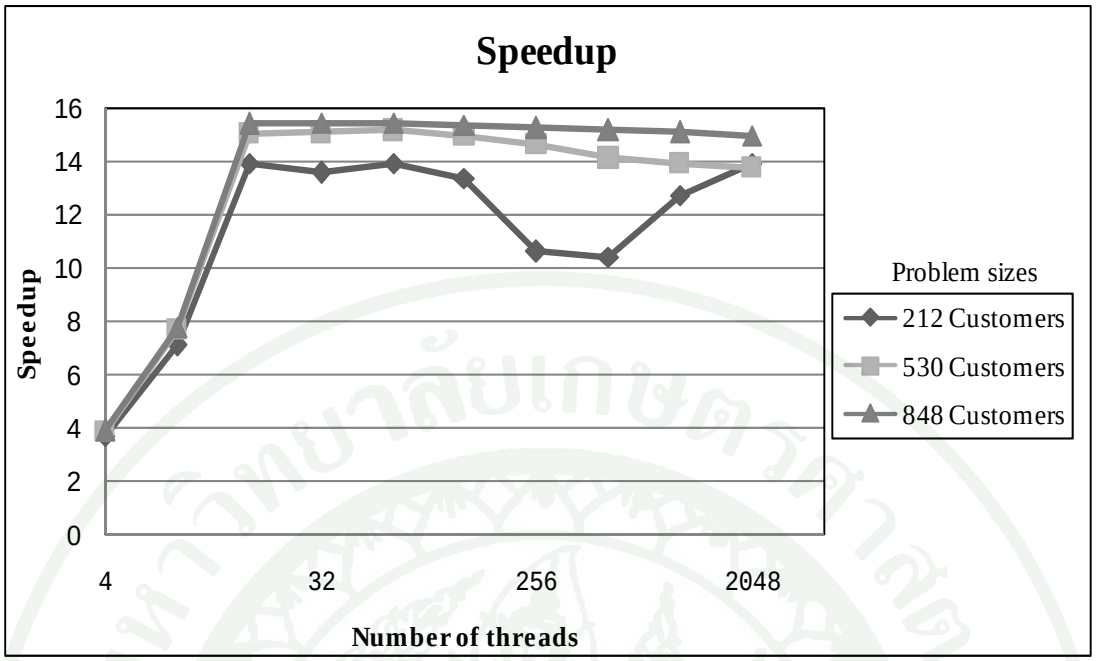
ภาพที่ 51 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์



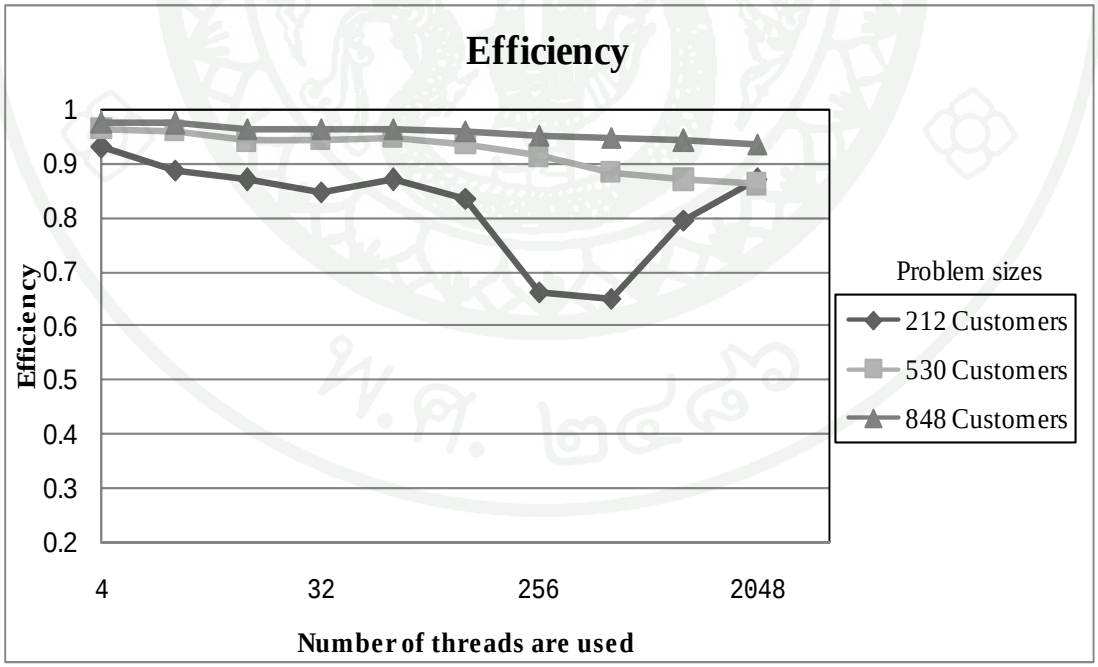
ภาพที่ 52 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์



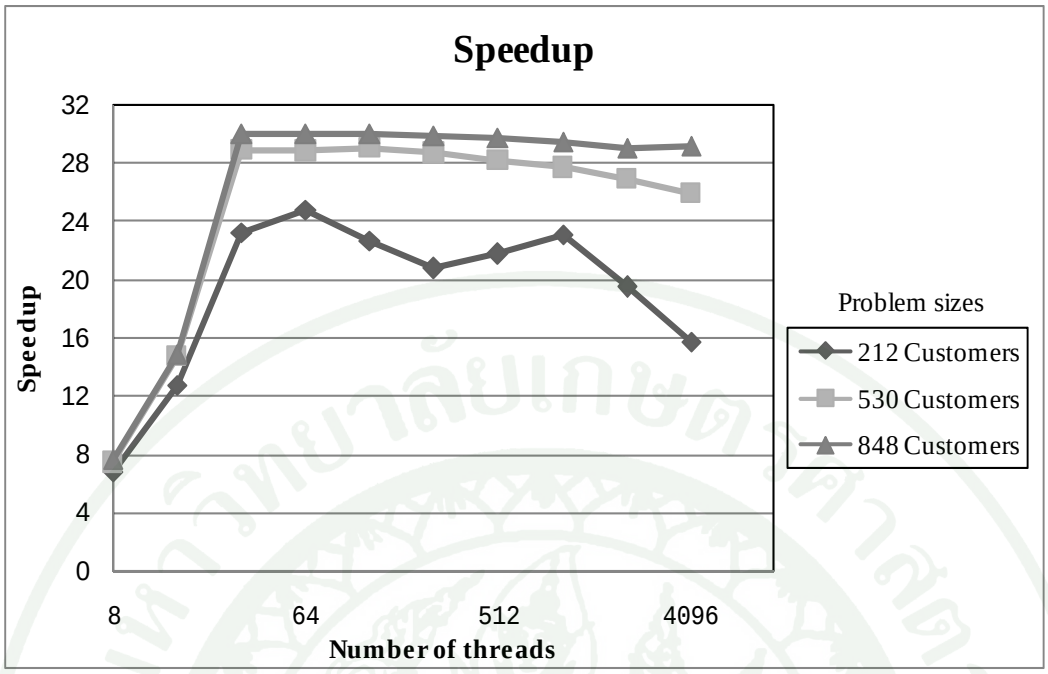
ภาพที่ 53 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 2 เวิร์คเกอร์



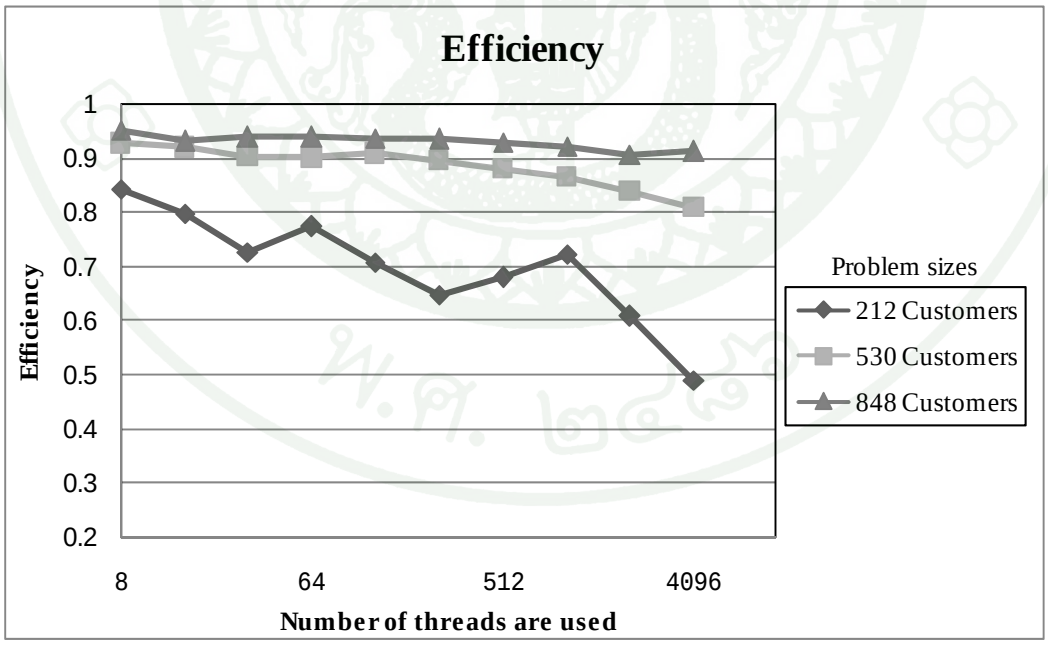
ภาพที่ 54 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 4 เวิร์คเกอร์



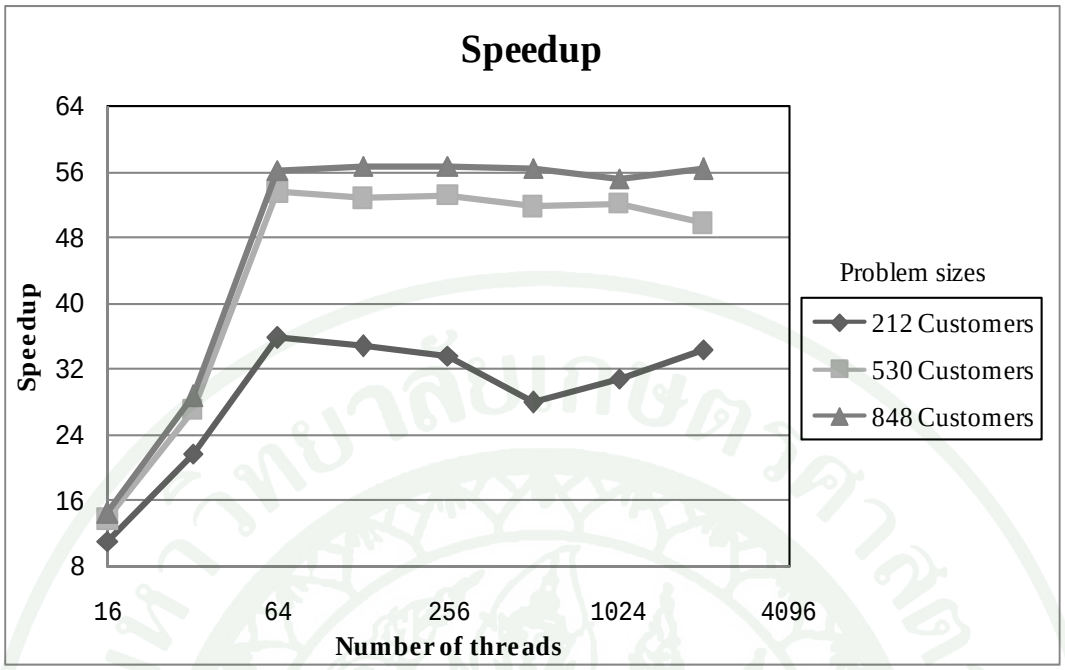
ภาพที่ 55 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 4 เวิร์คเกอร์



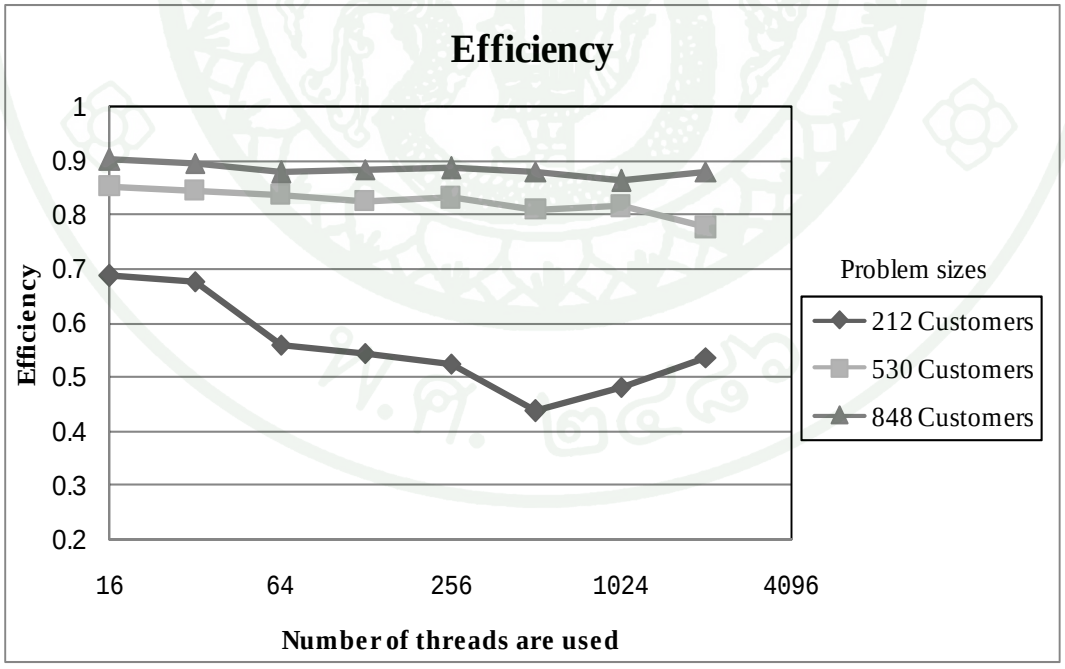
ภาพที่ 56 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 8 เวิร์คเกอร์



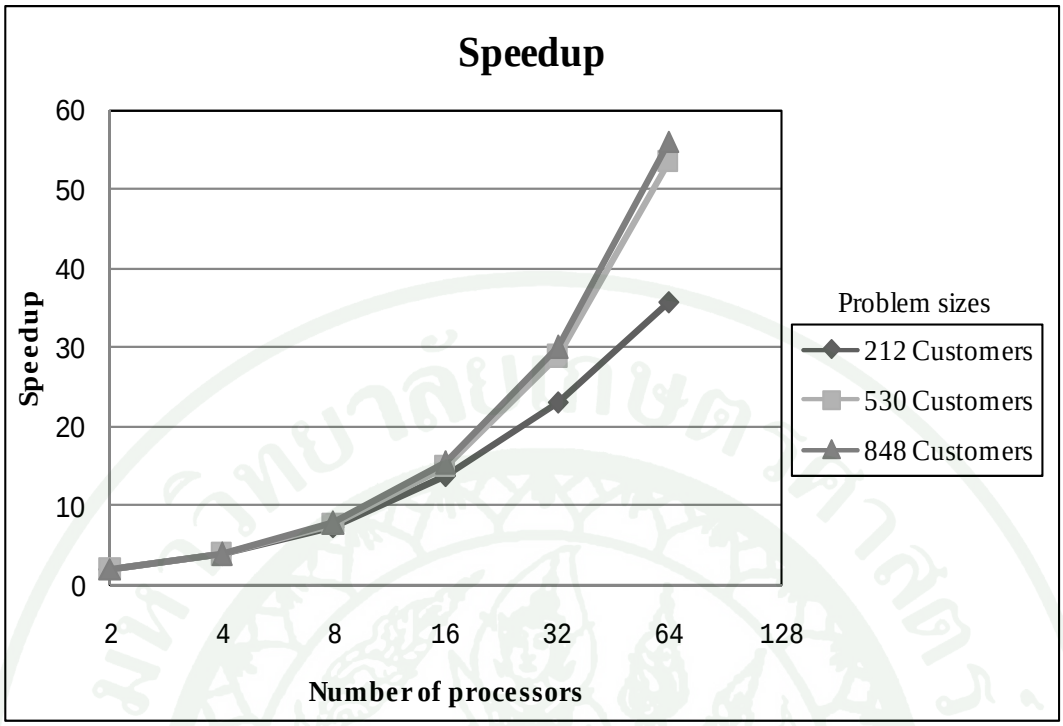
ภาพที่ 57 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 8 เวิร์คเกอร์



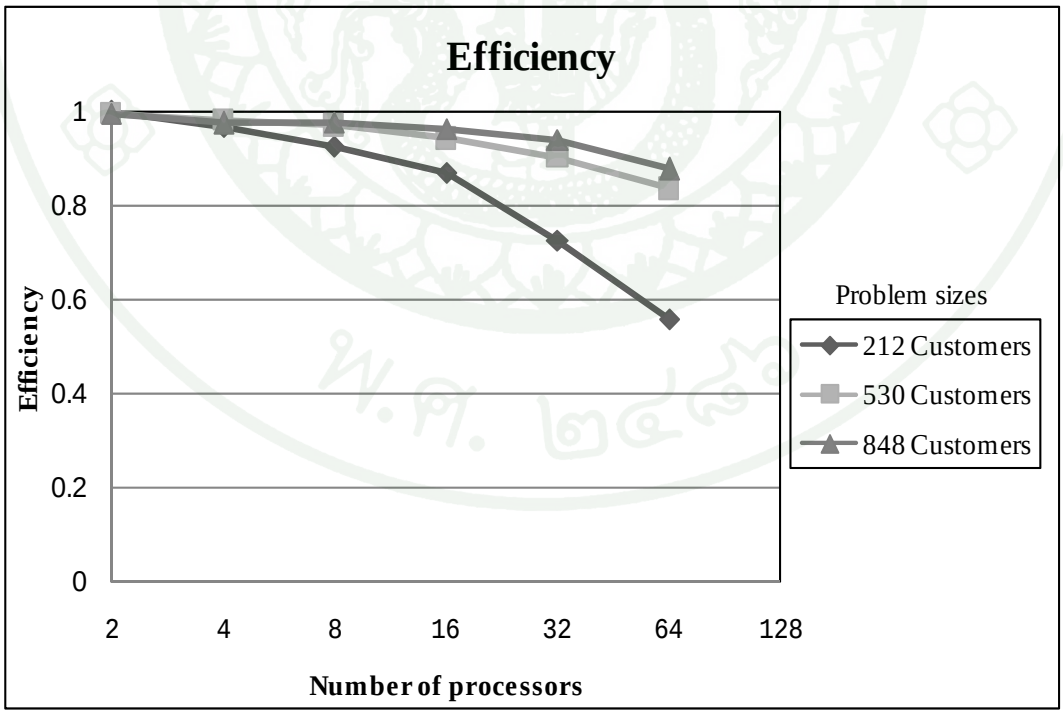
ภาพที่ 58 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 16 เวิร์คเกอร์



ภาพที่ 59 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 16 เวิร์คเกอร์



ภาพที่ 60 กราฟของ Speedup สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI



ภาพที่ 61 กราฟของ Efficiency สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI

จากผลการทดสอบประสิทธิภาพของระบบ DPDPTWI ที่แสดงในข้างต้นสามารถวิเคราะห์ประสิทธิภาพของระบบ DPDPTWI ได้ดังต่อไปนี้

1. จากผลการทดสอบเวลาในการประมวลผลสำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI บน 1 เวิร์คเกอร์, 2 เวิร์คเกอร์, 4 เวิร์คเกอร์, 8 เวิร์คเกอร์ และ 16 เวิร์คเกอร์ ซึ่งแสดงดังตารางที่ 7, ตารางที่ 8, ตารางที่ 9, ตารางที่ 10 และตารางที่ 11 ตามลำดับ แสดงให้เห็นว่า สำหรับแต่ละขนาดของปัญหานั้น เวลาที่ใช้ในการประมวลผลมีค่าลดลง เมื่อจำนวนของเทรดและจำนวนของเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้นสาเหตุที่เป็นเช่นนี้ สามารถแยกพิจารณาได้ 2 กรณีคือ

1.1 กรณีการเพิ่มจำนวนเทรดบนเวิร์คเกอร์ ทำให้เวลาที่ใช้ในการประมวลผลมีค่าลดลง ซึ่งจากผลการทดสอบจากตารางที่ 7 แสดงให้เห็นว่า เมื่อจำนวนของเทรดเพิ่มขึ้นทำให้เวลาที่ใช้ในการประมวลผลลดลง ซึ่งกรณีที่ดีที่สุดจากการทดสอบเห็นได้ว่า กรณีที่จำนวนของปัญหามีขนาดเท่ากับ 848 ลูกค้า จากการคำนวณด้วย 1 เทรดใช้เวลา 158.72 นาทีเหลือเพียง 40.34 นาที เมื่อใช้เทรดจำนวน 8 เทรด อย่างไรก็ตามการเพิ่มจำนวนเทรดก็ไม่ได้ทำให้เวลาที่ใช้ในการประมวลผลมีค่าลดลงตามจำนวนของเทรดที่เพิ่มขึ้น (เวลาไม่ได้ลดลงเท่ากับจำนวนเท่าของเทรดที่เพิ่มขึ้น) โดยสาเหตุที่เป็นเช่นนี้เนื่องมาจากงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ประมวลผลบนเครื่องเวิร์คเกอร์ที่มีจำนวนโปรเซสเซอร์เท่ากับ 4 โปรเซสเซอร์ทำให้ประสิทธิภาพสูงสุดในการประมวลผลที่เครื่องเวิร์คเกอร์ทำได้ คือ 4 เท่า เมื่อเทียบกับ 1 โปรเซสเซอร์ (1 โปรเซสเซอร์ทำงาน 1 เทรด) และงานการแก้ปัญหาดังกล่าวมีงานบางส่วนที่ไม่สามารถประมวลผลแบบมัลติเทรดได้ ดังนั้นเวลาที่ใช้ในการประมวลผลงานตรงส่วนนี้จึงไม่สามารถลดลงได้ด้วยการประมวลผลแบบมัลติเทรด และอีกสิ่งหนึ่งที่น่าสนใจ คือ Overhead ที่เกิดขึ้นจากการเพิ่มจำนวนเทรดซึ่งมีผลกระทบต่อเวลาในการประมวลผล ซึ่งจากผลการทดสอบจากตารางที่ 7 แสดงให้เห็นว่า เมื่อจำนวนของเทรดเพิ่มขึ้นเวลาในการประมวลผลมีแนวโน้มที่สูงขึ้น ตัวอย่างเช่น กรณีที่จำนวนของปัญหามีขนาดเท่ากับ 530 ลูกค้า การประมวลผลด้วย 1 เทรดใช้เวลา 19.68 นาที, เมื่อจำนวนเทรดเพิ่มเป็น 2 เทรดใช้เวลา 9.88 นาที, เมื่อจำนวนเทรดเพิ่มเป็น 4 เทรดใช้เวลา 5.01 นาที แต่เมื่อจำนวนเทรดเพิ่มเป็น 8 เทรด ใช้เวลา 5.04 นาทีและเมื่อจำนวนเทรดเพิ่มขึ้นเรื่อยๆ เวลาที่ใช้ในการประมวลผลเพิ่มสูงขึ้นด้วยเช่นกัน เป็นต้น

1.2 กรณีการเพิ่มจำนวนเวิร์คเกอร์ จากผลการทดสอบแสดงให้เห็นว่า สำหรับแต่ละขนาดปัญหางานนั้นเวลาที่ใช้ในการประมวลผลมีค่าลดลงเมื่อจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้น สาเหตุที่เป็นเช่นนี้เนื่องจาก เมื่อเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้นทำให้งานสามารถถูกแบ่งออกเป็นงานย่อยๆ ได้ในจำนวนที่เพิ่มขึ้นตามจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลที่เพิ่มขึ้นส่งผลให้งานนั้นถูกกระจายไปประมวลผลยังเวิร์คเกอร์ต่างๆ ที่เพิ่มขึ้นได้ในเวลาเดียวกันทำให้เวลาที่ใช้ในการประมวลผลมีค่าลดลง ซึ่งเมื่อพิจารณาผลการทดสอบจากตารางที่ 7, ตารางที่ 8, ตารางที่ 9, ตารางที่ 10 และตารางที่ 11 (เมื่อ 1 เวิร์คเกอร์ประกอบด้วย 4 โปรเซสเซอร์ ดังนั้น แต่ละเวิร์คเกอร์รองรับเทรดได้ 4 เทรด ซึ่งการพิจารณาจะพิจารณาตรงที่แต่ละเวิร์คเกอร์ใช้จำนวนเทรดเท่ากับ 4 เทรด เช่น 1 เวิร์คเกอร์ทำงาน 4 เทรด และ 2 เวิร์คเกอร์ทำงาน 8 เทรด เป็นต้น) ที่จำนวนเทรดเท่ากับ 4 เทรด, 8 เทรด, 16 เทรด, 32 เทรด และ 64 เทรด ตามลำดับ แสดงให้เห็นว่าเมื่อจำนวนของปัญหามีขนาดคงที่เวลาที่ใช้ในการประมวลผลมีค่าลดลง เมื่อจำนวนของเวิร์คเกอร์เพิ่มขึ้น ดังจะเห็นได้จากกรณีที่ขนาดปัญหามีขนาดเท่ากับ 848 ลูกค้า เมื่อทำงานบน 1 เวิร์คเกอร์, 2 เวิร์คเกอร์, 4 เวิร์คเกอร์, 8 เวิร์คเกอร์ และ 16 เวิร์คเกอร์ใช้เวลา 40.73 นาที, 20.32 นาที, 10.30 นาที, 5.29 นาทีและ 2.83 นาทีตามลำดับซึ่งจากการประมวลผลด้วย 1 เวิร์คเกอร์ ที่ใช้เวลาถึง 40.73 นาที เหลือเพียง 2.83 นาที เมื่อใช้ 16 เวิร์คเกอร์ในการประมวลผล อย่างไรก็ตาม ประโยชน์ของการเพิ่มจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผล ซึ่งส่งผลให้เวลาที่ใช้ในการประมวลผลมีค่าลดลงนั้นมีประโยชน์ลดลงเมื่อจำนวนเวิร์คเกอร์มีค่ามากขึ้น (เวลาของการประมวลผลไม่ได้ลดลงเป็นจำนวนเท่าตามจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลที่เพิ่มขึ้น) เช่น จากผลการทดสอบพบว่า การเพิ่มจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลเป็น 2 เวิร์คเกอร์ ส่งผลให้เวลาที่ใช้ในการประมวลผลมีค่าลดลงเกือบ 2 เท่า เมื่อเทียบกับเวลาที่ใช้ในการประมวลผลจาก 1 เวิร์คเกอร์ แต่เมื่อเพิ่มจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลเป็น 16 เวิร์คเกอร์ กลับไม่ส่งผลให้เวลาที่ใช้ในการประมวลผลมีค่าลดลงใกล้เคียง 16 เท่า เมื่อเทียบกับเวลาที่ใช้ในการประมวลผลจาก 1 เวิร์คเกอร์ เป็นต้น โดยสาเหตุที่เวลาที่ใช้ในการประมวลผลไม่ได้มีค่าลดลงอย่างที่ควรจะเป็น เมื่อจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้นนั้น เนื่องจากงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง มีงานบางส่วนที่ไม่สามารถประมวลผลแบบขนานได้ ทำให้เวลาที่ใช้ในการประมวลผลของงานส่วนนี้ไม่สามารถลดลงได้ด้วยการประมวลผลแบบขนาน ซึ่งส่งผลให้เมื่อจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผล มีจำนวนเพิ่มขึ้นเวลาที่ใช้ในการประมวลผลจึงไม่มีค่าลดลง ส่วนสาเหตุที่เวลาที่ใช้ในการประมวลผลไม่ได้ลดลงเป็นจำนวนเท่าเมื่อจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มสูงขึ้น คือ เนื่องจากการประมวลผลแบบขนานมีเวลาที่ต้องใช้เพิ่มเข้ามาเฉพาะสำหรับการประมวลผลแบบขนาน คือ เวลาของการสื่อสาร (Communication

time) ดังนั้น การเพิ่มจำนวนของเวิร์คเกอร์ที่ใช้ในการประมวลผลส่งผลให้เวลาของการสื่อสารมีค่าเพิ่มขึ้นเช่นกัน

2. การประเมินประสิทธิภาพของระบบการคำนวณแบบขนานด้วย Speedup โดยจากผลการทดสอบเวลาในการประมวลผลงานจากตารางที่ 7, ตารางที่ 8, ตารางที่ 9, ตารางที่ 10 และตารางที่ 11 สามารถคำนวณหา Speedup ซึ่งผลการทดสอบ Speedup ของแต่ละการทดสอบแสดงดังภาพที่ 50, ภาพที่ 52, ภาพที่ 54, ภาพที่ 56 และภาพที่ 58 จากผลการทดสอบเห็นได้ว่า สำหรับแต่ละขนาดปัญหาของงานนั้น Speedup มีค่าเพิ่มขึ้น เมื่อจำนวนเทรดที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้นจนกระทั่งจำนวนเทรดมีค่าเท่ากับค่าหนึ่งๆ (ขึ้นกับแต่ละผลการทดสอบ Speedup นั้นๆ) จะให้ค่า Speedup สูงสุด หลังจากนั้นค่า Speedup จะมีค่าลดลงอย่างช้าๆเมื่อจำนวนเทรดมีค่าเกินค่าหนึ่งๆ นั้น ตัวอย่างเช่น จากภาพที่ 52 กรณีที่ปัญหามีขนาดเท่ากับ 530 ลูกค้า ที่ทำการเพิ่มจำนวนของเทรดจาก 8 เทรด เป็น 16 เทรด ทำให้ Speedup มีค่าลดลง เป็นต้น เนื่องจากในกรณีเช่นนี้ Overhead ที่เกิดจากการเพิ่มจำนวนเทรดที่ใช้ในการประมวลผลมีค่าเพิ่มขึ้นมากเกินไป ซึ่งเมื่อเปรียบเทียบกับเวลาที่ใช้ในการประมวลผลของงานนั้นมีความแตกต่างกันน้อยมากจึงส่งผลให้ Speedup มีค่าไม่เพิ่มขึ้นแต่กลับมีค่าลดลงเมื่อจำนวนของเทรดเพิ่มขึ้น สิ่งหนึ่งที่น่าสนใจจากผลการทดสอบ คือ Speedup ของการประมวลผลสำหรับงานที่มีขนาดปัญหาใหญ่กว่ามีค่าสูงกว่า Speedup ของการประมวลผลสำหรับงานที่มีขนาดปัญหาเล็กกว่าเมื่อจำนวนเทรดที่ใช้ในการประมวลผลคงที่ เนื่องจากงานที่มีขนาดปัญหาใหญ่กว่ามีอัตราส่วนระหว่างเวลาที่ใช้ในการประมวลผลจริง ต่อ Overhead สูงกว่างานที่มีขนาดปัญหาเล็กกว่าจึงส่งผลให้ Speedup มีค่ามากกว่า ตัวอย่างเช่น จากภาพที่ 56 ค่า Speedup ของการแก้ปัญหาขนาด 848 ลูกค้ามีค่ามากกว่าค่า Speedup ของการแก้ปัญหาขนาด 212 ลูกค้า เป็นต้น ทำให้กล่าวได้ว่าระบบสามารถทำงานได้ค่า Speedup ที่สูงขึ้น เมื่องานมีขนาดปัญหาที่ใหญ่ขึ้น

3. การประเมินประสิทธิภาพของระบบการคำนวณแบบขนานด้วย Efficiency ซึ่งแสดงให้เห็นถึงประสิทธิภาพของระบบการคำนวณแบบขนานของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จากผลการทดสอบ Efficiency ของแต่ละการทดสอบจากภาพที่ 51, ภาพที่ 53, ภาพที่ 55, ภาพที่ 57 และภาพที่ 59 แสดงให้เห็นว่า สำหรับแต่ละขนาดปัญหางานนั้น Efficiency มีค่าสูง เมื่อจำนวนเทรดและจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนน้อยเพราะ Overhead มีค่าต่ำ (ค่า Overhead ของการเพิ่มจำนวนเทรดและค่า Overhead ของการเพิ่มจำนวนเวิร์คเกอร์) และค่า Efficiency มีอัตราการลดลงที่เพิ่มขึ้น เมื่อเทรดและเวิร์คเกอร์ที่ใช้ในการประมวลผลมีจำนวนเพิ่มขึ้น เนื่องจาก Overhead ที่มีค่าเพิ่มขึ้นตามจำนวน

เทรดและจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลที่เพิ่มขึ้น โดยจากผลการทดสอบพบว่า Efficiency ของการประมวลผลสำหรับงานที่มีขนาดปัญหาใหญ่มีค่าสูงกว่า Efficiency ของการประมวลผลสำหรับงานที่มีขนาดปัญหาเล็ก เมื่อจำนวนเทรดและจำนวนเวิร์คเกอร์ที่ใช้ในการประมวลผลคงที่ ซึ่งสอดคล้องกับผลการทดสอบที่ 1 คือ “การทดสอบประสิทธิภาพบนเครื่องเวิร์คเกอร์ที่ประกอบด้วย 2 โปรเซสเซอร์ซึ่งแต่ละโปรเซสเซอร์ประกอบด้วย 4 คอร์ จำนวน 2 เวิร์คเกอร์”

ภาพที่ 60 และภาพที่ 61 เป็นผลการทดสอบประสิทธิภาพของ Speedup และ Efficiency ของระบบ DPDPTWI ซึ่งแสดงถึงประสิทธิภาพในการประมวลผล ซึ่งในกรณีที่ดีที่สุดที่ปัญหามีขนาด 848 ลูกค้า สามารถให้ค่า Speedup ถึง 56.08 เท่า เมื่อเทียบกับการประมวลผลบน 1 โปรเซสเซอร์ (1 เวิร์คเกอร์ประกอบด้วย 4 โปรเซสเซอร์) และให้ค่า Efficiency ถึง 0.88 (คิดเป็น 88 %) เมื่อใช้ 64 โปรเซสเซอร์ (16 เวิร์คเกอร์) ซึ่งแสดงให้เห็นว่าระบบ DPDPTWI สามารถแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่มีขนาดใหญ่ได้อย่างมีประสิทธิภาพ

การทดสอบการทำงานของระบบ DPDPTWI โดยใช้โมเดล TSA สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

การทดสอบการทำงานของระบบ DPDPTWI โดยใช้โมเดล TSA สำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เป็นการทดสอบระบบ DPDPTWI กับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เกณฑ์มาตรฐาน (PDPTW Benchmark) โดยระบบ DPDPTWI ที่ใช้โมเดล TSA นี้ได้พัฒนาต่อเนื่องมาจากระบบ DPDPTWI ที่ใช้โมเดล SIA ซึ่งคำนึงถึงการนำไปใช้งานจริงและความต้องการของผู้ใช้ คือ ได้ผลลัพธ์จากการแก้ปัญหาภายในเวลาที่กำหนด โดยระบบ DPDPTWI ที่ใช้โมเดล TSA นี้ใช้วิธีการประมวลผลแบบขนานที่ได้กล่าวไว้ในหัวข้อ “การแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ด้วยโมเดล SIA และ TSA บนแนวคิดของการคำนวณแบบขนานโดยใช้กรอบการทำงานแบบสมรรถนะสูง”

โดยวัตถุประสงค์ของการทดสอบ คือ เพื่อเปรียบเทียบผลลัพธ์จากการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ระหว่าง ผลลัพธ์จากการแก้ปัญหาที่ดีที่สุด (Best know solution), ผลลัพธ์จากการแก้ปัญหาด้วยวิธีของ Russell Bent

และ Pascal Van Hentenryck, ผลลัพธ์จากการแก้ปัญหาด้วยวิธีของ Stefan Ropke และ David Pisinger และผลลัพธ์จากการแก้ปัญหาบนระบบ DPDPTWI ซึ่งผลลัพธ์จากการแก้ปัญหา คือ จำนวนของยานพาหนะที่ใช้กับปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง โดยการทดสอบใช้เวิร์คเกอร์สำหรับการทำงานจำนวน 1 เครื่อง ซึ่งรายละเอียดแสดงดังตารางที่ 12

ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งที่นำมาทดสอบนั้นใช้ปัญหาที่นำเสนอโดย Haibing Li และ Andrew Lim (Li and Lim, 2004) และการทดสอบการแก้ปัญหาบนระบบ DPDPTWI ที่ใช้โมเดล TSA นี้ กำหนดเวลาของการประมวลผลสำหรับแต่ละปัญหา คือ เวลา 5 นาที

ตารางที่ 12 รายละเอียดของฮาร์ดแวร์และซอฟต์แวร์สำหรับการทดสอบประสิทธิภาพของระบบ DPDPTWI

Machine	Hardware	Operating System	Software
1 Manager	Intel(R) Core(TM)2 T5500 1.66 GHz, 2.5 GB of RAM	Windows XP Professional	Manager, MySQL Server for Manager
1 Worker	Intel(R) Core(TM)2 T5500 1.66 GHz, 2.5 GB of RAM	Windows XP Professional	Worker

โดยการเปรียบเทียบผลลัพธ์ของการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง แสดงดังตารางที่ 13 และตารางที่ 14

ตารางที่ 13 ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ขนาด 100 ลูกค้า

Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI	Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI
lc101	10	10	10	11	lc206	3	3	3	3
lc102	10	10	10	11	lc207	3	3	3	4
lc103	9	9	9	11	lc208	3	3	3	3
lc104	9	9	9	8	lr101	19	19	19	20
lc105	10	10	10	10	lr102	17	17	17	18
lc106	10	10	10	12	lr103	13	12	13	18
lc107	10	10	10	9	lr104	9	9	9	13
lc108	10	10	10	10	lr105	14	14	14	17
lc109	9	10	9	10	lr106	12	12	12	15
lc201	3	3	3	4	lr107	10	10	10	6
lc202	3	3	3	4	lr108	9	9	9	14
lc203	3	3	3	4	lr109	11	11	11	14
lc204	3	3	3	3	lr110	10	10	10	15
lc205	3	3	3	3	lr111	10	10	10	15

ตารางที่ 13 (ต่อ)

Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI	Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI
lr112	9	9	9	13	lrc103	11	11	11	15
lr201	4	4	4	5	lrc104	10	10	10	14
lr202	3	3	3	5	lrc105	13	13	13	18
lr203	3	3	3	4	lrc106	11	11	11	16
lr204	2	2	2	4	lrc107	11	11	11	15
lr205	3	3	3	4	lrc108	10	10	10	14
lr206	3	3	3	4	lrc201	4	4	4	5
lr207	2	2	2	2	lrc202	3	3	3	5
lr208	2	2	2	3	lrc203	3	3	3	5
lr209	3	3	3	4	lrc204	3	3	3	4
lr210	3	3	3	4	lrc205	4	4	4	5
lr211	2	3	2	3	lrc206	3	3	3	5
lrc101	14	14	14	16	lrc207	3	3	3	5
lrc102	12	12	12	16	lrc208	3	3	3	4

ตารางที่ 14 ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ขนาด 200 ลูกค้า

Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI	Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI
lc1_2_1	20	20	20	22	lc2_2_5	6	6	6	8
lc1_2_2	19	19	19	25	lc2_2_6	6	6	6	9
lc1_2_3	17	18	17	22	lc2_2_7	6	6	6	9
lc1_2_4	17	17	17	21	lc2_2_8	6	6	6	8
lc1_2_5	20	20	20	25	lc2_2_9	6	6	6	9
lc1_2_6	20	20	20	28	lc2_2_10	6	6	6	8
lc1_2_7	20	20	20	22	lr1_2_1	20	20	20	24
lc1_2_8	20	20	20	24	lr1_2_2	17	18	17	21
lc1_2_9	18	18	18	22	lr1_2_3	15	15	15	19
lc1_2_10	17	18	17	22	lr1_2_4	10	11	10	17
lc2_2_1	6	6	6	9	lr1_2_5	16	17	16	19
lc2_2_2	6	6	6	9	lr1_2_6	14	14	14	20
lc2_2_3	6	6	6	9	lr1_2_7	12	13	12	18
lc2_2_4	6	7	6	8	lr1_2_8	9	10	9	15

ตารางที่ 14 (ต่อ)

Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI	Problem	Best Know Solution	R. Bent & P.V. Hentenryck	S. Ropke & D. Pisinger	DPDPTWI
lr1_2_9	14	14	14	19	lrc1_2_3	13	13	13	17
lr1_2_10	11	12	11	17	lrc1_2_4	10	11	10	16
lr2_2_1	5	5	5	6	lrc1_2_5	16	16	16	22
lr2_2_2	4	4	4	6	lrc1_2_6	17	17	17	21
lr2_2_3	4	4	4	6	lrc1_2_7	14	16	14	20
lr2_2_4	3	4	3	4	lrc1_2_8	13	14	13	20
lr2_2_5	4	4	4	6	lrc1_2_9	13	15	13	20
lr2_2_6	4	4	4	6	lrc1_2_10	12	13	12	19
lr2_2_7	3	4	3	5	lrc2_2_1	6	7	6	9
lr2_2_8	2	3	2	4	lrc2_2_2	5	6	5	8
lr2_2_9	3	4	3	6	lrc2_2_3	4	5	4	6
lr2_2_10	3	4	3	6	lrc2_2_4	3	4	3	5
lrc1_2_1	19	19	19	21	lrc2_2_5	5	5	5	9
lrc1_2_2	15	16	15	19	lrc2_2_6	5	5	5	7

จากตารางที่ 13 และตารางที่ 14 แสดงการเปรียบเทียบผลลัพธ์ (จำนวนยานพาหนะ) ที่ได้รับจากการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งแสดงให้เห็นถึงผลลัพธ์ที่ได้จากระบบ DPDPTWI มีผลลัพธ์ที่ใกล้เคียงกับผลลัพธ์จากการแก้ปัญหาที่ดีที่สุด และใกล้เคียงกับผลลัพธ์จากการแก้ปัญหาด้วยวิธีของ Russell Bent และ Pascal Van Hentenryck และผลลัพธ์จากการแก้ปัญหาด้วยวิธีของ Stefan Ropke และ David Pisinger

จากการทดสอบการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งบนระบบ DPDPTWI แสดงให้เห็นว่า หากทำการทดสอบเพิ่มโดยการเพิ่มจำนวนของเวิร์คเกอร์ที่ใช้ประมวลผล, เพิ่มจำนวนเทรดให้กับแต่ละเครื่อง และเพิ่มเวลาที่ใช้สำหรับการประมวลผล ผู้วิจัยคาดว่าผลลัพธ์ที่ได้จากการแก้ปัญหาดังกล่าวมีแนวโน้มที่จะดีกว่าเดิม

โดยในการทดสอบการทำงานของระบบ DPDPTWI ที่ใช้โมเดล TSA นี้ เพื่อแสดงให้เห็นถึงคุณภาพของผลลัพธ์ที่ได้จากการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง และระบบ DPDPTWI สามารถรองรับกับจำนวนของเวิร์คเกอร์ที่เพิ่มขึ้น เพื่อใช้รองรับกับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งนี้ สามารถช่วยเพิ่มประสิทธิภาพในการประมวลผลและสามารถรองรับกับปริมาณงานจำนวนมากได้

หมายเหตุ ตารางที่ 13 และตารางที่ 14 แสดงผลลัพธ์จากการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งแสดงเฉพาะจำนวนยานพาหนะที่ใช้สำหรับการแก้ปัญหาเท่านั้น

สรุปและข้อเสนอแนะ

สรุป

วิทยานิพนธ์นี้ เสนอระบบสำหรับการเพิ่มความเร็วในการประมวลผลให้กับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งโดยใช้พลังการคำนวณแบบขนานแบบ Hybrid ซึ่งผสมผสานระหว่าง Message passing และ Multithreading ที่ได้จากระบบที่พัฒนาขึ้นซึ่งระบบรองรับกับการใช้งานของผู้ใช้ โดยระบบจัดเตรียมแบบฟอร์มเอ็กเซลสำหรับให้ผู้ใช้สร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง และสามารถเรียกดูผลลัพธ์จากงานดังกล่าวผ่านทางแบบฟอร์มนี้ และผู้ใช้สามารถส่งงานเข้าสู่ระบบผ่านทาง GUI ซึ่งสะดวกต่อผู้ใช้ ซึ่งซ่อนความซับซ้อนของการประมวลผลแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งจากผู้ใช้ ทำให้ผู้ใช้ยังคงสร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่งได้อย่างปกติ แต่พลังการประมวลผลสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ได้รับจากระบบมีค่ามากยิ่งขึ้น โดยจากผลการทดสอบแสดงให้เห็นว่า การแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง จากที่ใช้เวลา 158.72 นาที ลดลงเหลือเพียง 2.80 นาที เมื่อใช้เครื่องมัลติคอร์คอมพิวเตอร์จำนวน 16 เครื่อง 64 คอร์ ในการประมวลผลแบบขนาน ซึ่งจากความเร็วของการประมวลผลที่เพิ่มขึ้นนี้ ทำให้ผู้ใช้สามารถสร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่มีขนาดใหญ่ได้มากยิ่งขึ้น และในด้านของความทนต่อความผิดพลาดของระบบ เมื่อเกิดความล้มเหลวจากเครื่องเวิร์คเกอร์ที่ใช้ประมวลผล ระบบสามารถทนต่อความผิดพลาดของเวิร์คเกอร์ได้ในระดับหนึ่ง และสามารถดำเนินการระบบต่อไปได้

ข้อเสนอแนะ

งานวิจัยนี้ได้พัฒนาระบบ Distributed Pickup and Delivery Problem with Time Windows Infrastructure (DPDPTWI) ซึ่งเป็นระบบที่รองรับการคำนวณแบบขนานเพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งเป็นระบบต้นแบบ จึงต้องมีการพัฒนาระบบเพิ่มเติม เพื่อทำให้ระบบสามารถรองรับการใช้งานได้ดีและสมบูรณ์ยิ่งขึ้น เช่น การทนต่อความผิดพลาดของเวิร์คเกอร์ให้ดียิ่งกว่าเดิม, การจัดการกับความผิดพลาดที่เกิดขึ้นกับเครื่องคอมพิวเตอร์ในระบบ, การพัฒนาระบบความปลอดภัยสำหรับผู้ใช้งานระบบและสำหรับข้อมูลของระบบ และการพัฒนาระบบ DPDPTWI ให้สามารถรองรับการแก้ปัญหาการจัดเส้นทางยานพาหนะภายใต้เงื่อนไขอื่นได้

เอกสารและสิ่งอ้างอิง

Alba, E. 2005. **Parallel Metaheuristics: A New Class of Algorithms**. John Wiley & Sons, Inc., Hoboken, New Jersey.

Asanovic, K., R. Bodik., B.C. Catanzaro., J.J. Gebis., P. Husbands., K. Keutzer., D.A. Patterson.,W.L. Plishker., J. Shalf., S.W. Williams and K.A. Yelick. 2006. **The Landscape of Parallel Computing Research: A View from Berkeley.**

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.

Bent, R. and P. V. Hentenryck. 2006. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. **Elsevier Ltd** 33 (2006): 875-893.

Carver, R. H. and K. C. Tai. 2006. **Modern Multithreading: Implementing, Testing, and Debugging Multithreaded Java and C++/Pthreads/Win32 Programs.** John Wiley & Sons, Inc., Hoboken, New Jersey.

Chrysanthakopoulos, G. 2007. **Concurrency Runtime: An Asynchronous Messaging Library for C# 2.0**. Channel 9 Wiki Microsoft. Available Source: http://channel9.msdn.com/wiki/wiki/ConcurrencyRuntime/, May 15, 2008.

Chrysanthakopoulos, G. and S. Singh. 2005. **An Asynchronous Messaging Library for C#.** Synchronization and Concurrency in Object-Oriented Languages (SCOOL) at OOPSLA Workshop, San Diego, CA. Available Source: http://urresearch.rochester.edu/handle/1802/2105, July 1, 2008.

Dongarra, J. 2007. The Promise and Perils of the Coming Multicore Revolution and Its Impact. **CTWatch Quarterly.** 3.

_____, I. Foster., G. Fox., W. Gropp., K. Kennedy., L. Torczon and A. White. 2003. **The Sourcebook of Parallel Computing**. Morgan Kaufmann, San Francisco.

Erl, T. 2004. **Service-Oriented Architecture: A field guide to integrating XML and Web Services.** Prentice Hall Ptr.

Johns, K. and T. Taylor. 2008. **Professional Microsoft® Robotics Developer Studio.** Wiley Publishing, Inc., Indianapolis, Indiana.

Kumar, V., A. Grama, A. Gupta and G. Karypis. 1994. **Introduction to Parallel Computing: Design and Analysis of Algorithms.** 1st ed. The Benjamin/Cummings Publishing Company, Inc., California.

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.

Microsoft. 2009. **Microsoft Robotics Developer Studio CCR Introduction.** Available Source: http://msdn.microsoft.com/en-us/library/bb648752.aspx, May 5, 2009.

Microsoft. 2009. **Microsoft Robotics Developer Studio DSS Introduction.** Available Source: http://msdn.microsoft.com/en-us/library/bb483056.aspx, May 5, 2009.

Microsoft Robotics. 2008**. Microsoft Robotics Studio is a Windows-based environment that includes end-to-end Robotics Development Platform, lightweight service-oriented runtime, and a scalable and extensible platform**. Microsoft Robotics Developer Center. Available Source: http://msdn.microsoft.com/robotics/, March 20, 2008.

Nielsen, H. F. and G. Chrysanthakopoulos. 2007. **Decentralized Software Services Protocol.**

Paradiseo. 2009. **Paradiseo.** Available Source: http://paradiseo.gforge.inria.fr/index.php?n=Paradiseo.Home?from=Main.HomePage, December 7, 2009.

Qiu, X., G. Fox., G. Chrysanthakopoulos and H. F. Nielsen. 2007. **High Performance Multi-Paradigm Messaging Run Time on Multicore Systems.**

_____ and A. Ho. 2007. **Analysis of Concurrency and Coordination Runtime CCR and DSS**. Anabas Inc.

_____, H.Yuan., S. H. Bae., G. Chrysanthakopoulos and H. F. Nielsen. 2007. **Performance Measurements of CCR and MPI on Multicore Systems Summary.** Available Source: http://grids.ucs.indiana.edu/ptliupages/presentations/MCPerformanceSept21-07.ppt, June 21, 2008.

_____, G.C. Fox., H. Yuan., S. H. Bae., G. Chrysanthakopoulos and H. F. Nielsen. 2007. High Performance Multi-Paradigm Messaging Runtime Integrating Grids and Multicore Systems, pp. 407-414. *In* **Proceedings of the Third IEEE International Conference on e-Science and Grid Computing**. Washington, DC, USA.

Qiu, X., A. Ho, G.C. Fox., H. Yuan., S. H. Bae., G. Chrysanthakopoulos and H. F. Nielsen. 2008. Performance of Multicore Systems on Parallel Data Clustering with Deterministic Annealing, pp. 407-416. *In* **Proceedings of the 8th International Conference on Computational Science (ICCS).** Kraków, Poland.

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

Richter, J. 2006. Concurrent Affairs: Concurrency and Coordination Runtime. **MSDN Magazine.** (September 2006).

Ropke, S. 2005. **Heuristic and exact algorithms for vehicle routing problems.** Ph.D. Thesis, Computer science department at the University of Copenhagen.

_____ and D. Pisinger. 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows, pp. 455-472. **Transportation Science**. Institute for Operations Research and the Management Sciences (INFORMS), Maryland.

Silberschatz, A., P. B. Galvin and G. Gagne. 2006. **Operating System Principles.** 7th ed. John Wiley & Sons (Asia) Pte Ltd.

Tabu Search Algorithm. 1993. **Tabu search algorithm.** Available Source: http://74.125.153.132/search?q=cache:WPhpaO4B8YAJ:ocw.kfupm.edu.sa/user/EE55601/07-TS-modified.doc+tabu+search+algorithm+filetype:doc&cd=1&hl=th&ct=clnk&gl=th&client=firefox-a, January 8, 2010.

Tanenbaum, A. S. 2001. **Modern Operating Systems.** 2nd ed. Pearson Education, Inc., India.

Tanenbaum, A. S. and M.V. Steen. 2002. **Distributed Systems: Principles and Paradigms**. 2nd ed. Prentice Hall Inc., New Jersey.

Wilkinson, B. and M. Allen. 1999. **Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers.** Prentice Hall., New Jersey.

ภาคผนวก

ภาคผนวก ก

ขั้นตอนการติดตั้งระบบ

ขั้นตอนการติดตั้งระบบ

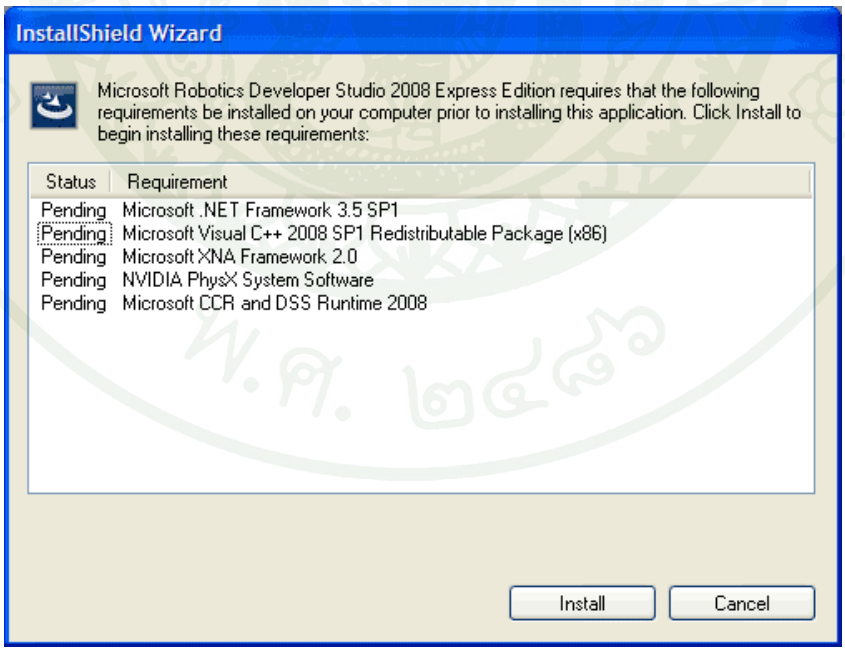
ระบบ DPDPTWI มีส่วนประกอบอยู่ 3 ส่วน คือ เมเนเจอร์ (Manager), เวิร์คเกอร์ (Worker) และผู้ใช้ (User) โดยแต่ละส่วนประกอบมีรายละเอียดของขั้นตอนการติดตั้ง ดังต่อไปนี้

เมเนเจอร์ (Manager)

1. ติดตั้งระบบปฏิบัติการไมโครซอฟท์วินโดวส์ (Microsoft Windows) โดยสำหรับการติดตั้งระบบ DPDPTWI ใช้ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ เอกซ์พี (Microsoft Windows XP)

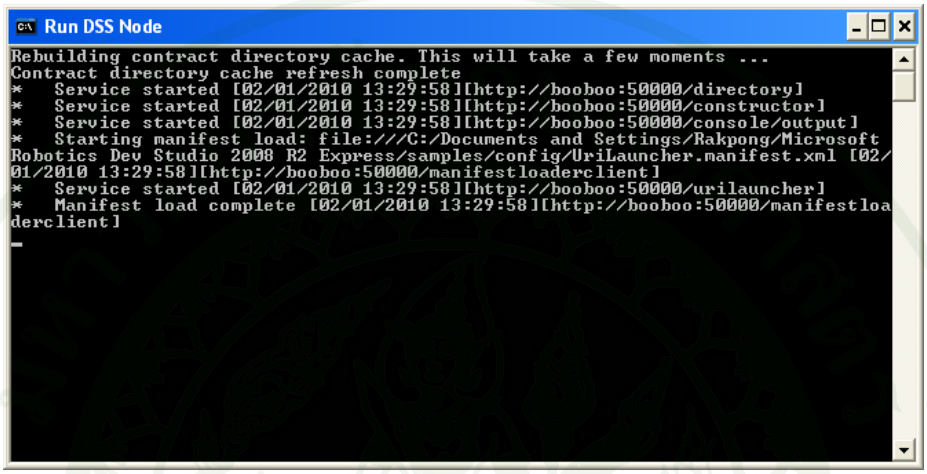
2. ติดตั้ง Microsoft Office 2007

3. ติดตั้ง Microsoft Robotics Developer Studio 2008 (MRDS) โดยให้ทำการติดตั้งตามขั้นตอนที่ปรากฏขึ้นในหน้าต่างของการติดตั้ง Microsoft Robotics Developer Studio 2008 โดยการติดตั้ง Microsoft Robotics Developer Studio 2008 แสดงดังภาพผนวกที่ ก1



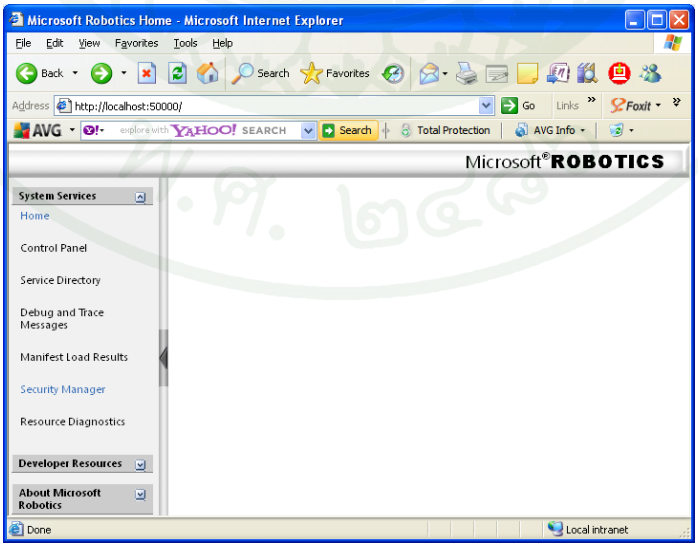
ภาพผนวกที่ ก1 การติดตั้ง MRDS

4. เมื่อติดตั้ง Microsoft Robotics Developer Studio 2008 เสร็จสิ้น ให้เปิดระบบ Microsoft Robotics Developer Studio 2008 โดยกดปุ่ม **Start > All Programs > Microsoft Robotics Developer Studio 2008 R2 Express > Run DSS Node** จากนั้นจะปรากฏหน้าต่าง Run DSS Node ดังภาพผนวกที่ ก2



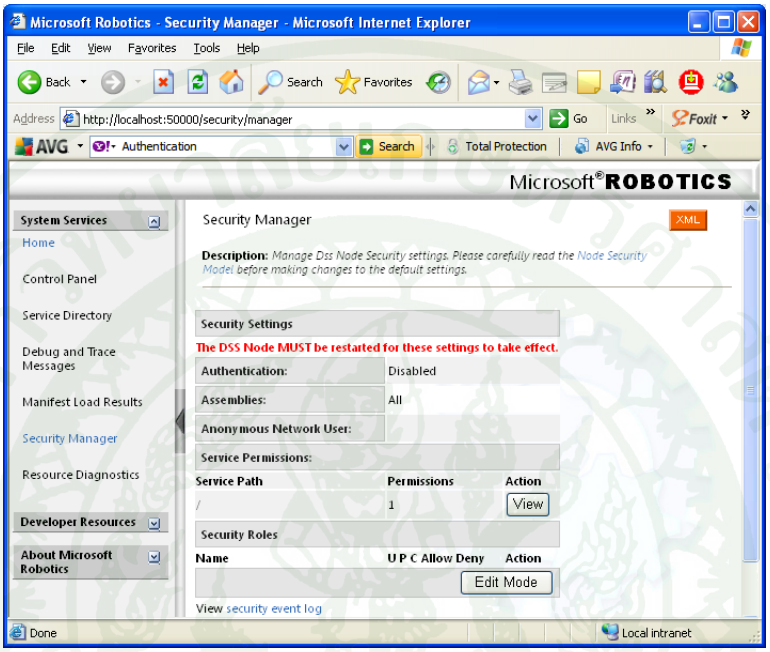
ภาพผนวกที่ ก2 การทำงานของ MRDS

5. จากขั้นตอนที่ 3 ให้เปิดเว็บบราวเซอร์(แนะนำให้ใช้ Internet Explorer) โดยพิมพ์ URL ชื่อ http://localhost:50000/ จากนั้นจะปรากฏหน้าเว็บไซต์ ดังภาพผนวกที่ ก3



ภาพผนวกที่ ก3 หน้าเว็บไซต์บน MRDS

6. จากขั้นตอนที่ 4 ปิดระบบรักษาความปลอดภัยโดยเลือก **Security Manager > กดปุ่ม Edit Mode > กดปุ่ม Disable ของหัวข้อ Authentication หลังจากนั้นกดปุ่ม Finish** โดยจะปรากฏผลลัพธ์ดังภาพผนวกที่ ก4



ภาพผนวกที่ ก4 การปิดระบบรักษาความปลอดภัยบน MRDS

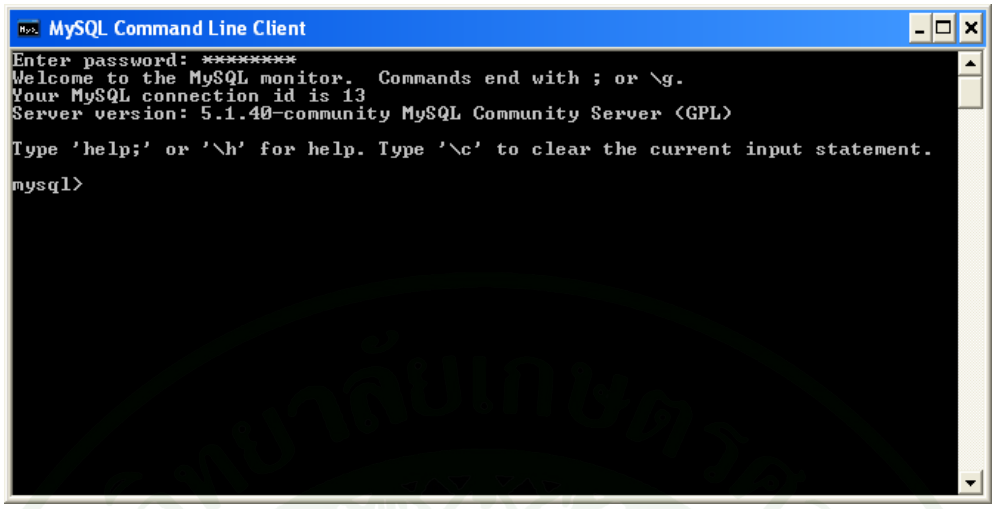
7. ติดตั้ง MySQL Server (การติดตั้งนี้ เลือกใช้ MySQL Server 5.1) โดยให้ทำการติดตั้งตามขั้นตอนที่ปรากฏขึ้นในหน้าต่างของการติดตั้ง MySQL Server ซึ่งการติดตั้งนี้ เลือกรูปแบบการติดตั้งเป็นแบบ **Typical** หลังจากการติดตั้งเสร็จสิ้น ในขั้นตอนสุดท้ายของการติดตั้ง มีการถามว่าต้องการทำการเซ็ตค่าของ MySQL Server เลยหรือไม่ ซึ่งถ้าต้องการ ก็ให้ทำการเลือก **Configure the MySQL Server now** แต่ถ้าไม่ต้องการ ก็ไม่ต้องทำการเลือก **Configure the MySQL Server now** โดยในการติดตั้งนี้ ทำการเลือก **Configure the MySQL Server now** เพื่อทำการเซ็ตค่าของ MySQL Server ในทันที ซึ่งเมื่อกดปุ่ม **Finish** ในหน้าต่างของการติดตั้ง MySQL Server ก็ปรากฏหน้าต่าง **MySQL Server Instance Configuration Wizard** ขึ้น เพื่อใช้ในการเซ็ตค่าของ MySQL Server ก็ให้ทำการขั้นตอนที่ปรากฏขึ้นต่อไป โดยการเซ็ตค่าของ MySQL Server ในที่นี้เลือก **Configuration Type** เป็นแบบ **Standard Configuration**

8. ติดตั้ง MySQL Connector/Net เพื่อทำให้ MySQL สามารถเชื่อมต่อกับ NET Framework ได้ (การติดตั้งนี้ เลือกใช้ MySQL Connector/Net 5.1.7) โดยให้ทำการติดตั้งตามขั้นตอนที่ปรากฏขึ้นในหน้าต่างของการติดตั้ง MySQL Connector/Net

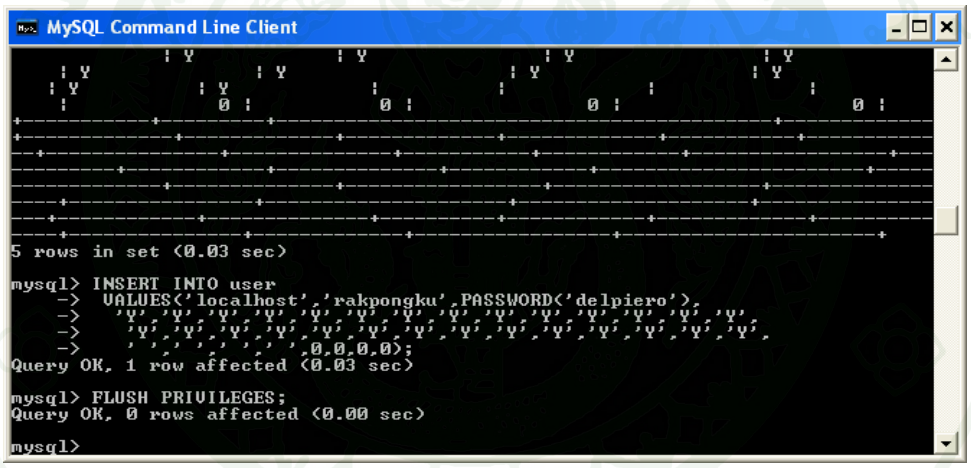
9. สร้างฐานข้อมูล (Database) สำหรับ DPDPTWI บน MySQL Server โดยแบ่งเป็น 2 ส่วน คือ

9.1 สร้างบัญชีรายชื่อ (Account) สำหรับ DPDPTWI โดยกำหนดให้มีชื่อว่า “rakpong” (rakpong เป็นชื่อเดิมของระบบ ซึ่งในการติดตั้งต่อไปทั้งหมดใช้ชื่อนี้) ซึ่งการสร้างบัญชีรายชื่อสำหรับ DPDPTWI ทำได้โดยกดปุ่ม **Start** เลือก **All Programs** > **MySQL** > **MySQL Server <*Version*> > MySQL Command Line Client** จากนั้น ปรากฏหน้าต่าง **MySQL Command Line Client** ขึ้น ให้ทำการใส่ Password ของ Root ก็ปรากฏผลลัพธ์ ดังภาพผนวกที่ ก5 หลังจากนั้นให้ทำการสร้างบัญชีรายชื่อสำหรับ DPDPTWI โดยใช้คำสั่งดังต่อไปนี้ (สามารถคัดลอก (Copy) คำสั่งทั้งหมด แล้วนำไปวาง (Paste) โดยคลิ้กขวา แล้วเลือก Paste บนหน้าต่าง **MySQL Command Line Client** ได้ ซึ่งผลลัพธ์ที่ได้ ปรากฏดังภาพผนวกที่ ก6 จากนั้นให้ใช้คำสั่ง **Exit** เพื่อจบการทำงาน)

```
show databases;
use mysql;
show tables;
select * from user;
INSERT INTO user
 VALUES('localhost','rakpong',PASSWORD('delpiero'),
  'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y',
   'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y',
   ' ',' ',' ',' ',0,0,0,0);
FLUSH PRIVILEGES;
```

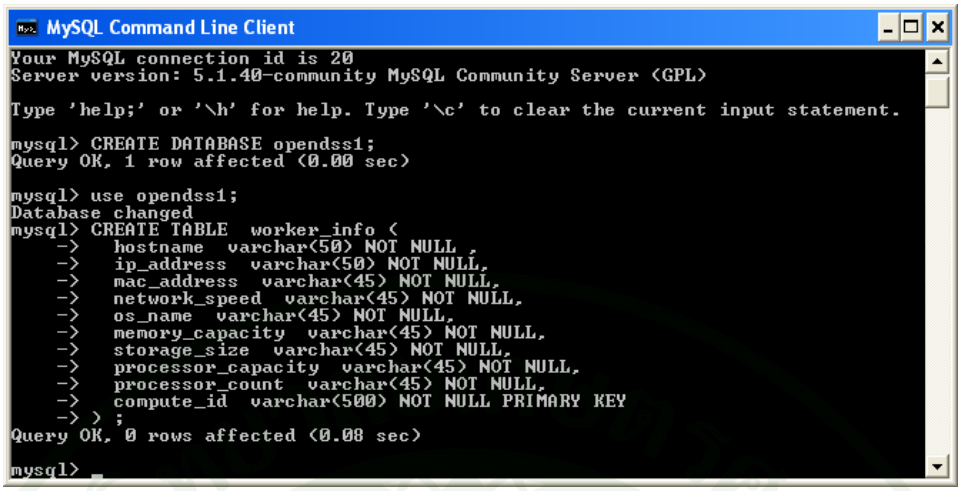


ภาพผนวกที่ ก5 การสร้างบัญชีรายชื่อสำหรับ DPDPTWI บน MySQL Server (1)



ภาพผนวกที่ ก6 การสร้างบัญชีรายชื่อสำหรับ DPDPTWI บน MySQL Server (2)

9.2 สร้างฐานข้อมูลสำหรับใช้ในการทำงานของ DPDPTWI ที่มีชื่อว่า “rakpong” โดยกดปุ่ม **Start** เลือก **All Programs** > **MySQL** > **MySQL Server <*Version*>** > **MySQL Command Line Client** จากนั้น ปรากฏหน้าต่าง **MySQL Command Line Client** ขึ้น ให้ทำการใส่ Password ของ Root ก็ปรากฏผลลัพธ์ ดังภาพผนวกที่ ก5 ที่กล่าวมาข้างต้น หลังจากนั้นให้ทำการสร้างฐานข้อมูล สำหรับใช้ในการทำงานของ DPDPTWI โดยใช้คำสั่งดังต่อไปนี้ (สามารถคัดลอก (Copy) คำสั่งทั้งหมด แล้วนำไปวาง (Paste) โดยคลิ้กขวา แล้วเลือก Paste บนหน้าต่าง **MySQL Command Line Client** ได้ ซึ่งผลลัพธ์ที่ได้ ปรากฏดังภาพผนวกที่ ก7 จากนั้นให้ใช้คำสั่ง **Exit** เพื่อจบการทำงาน)



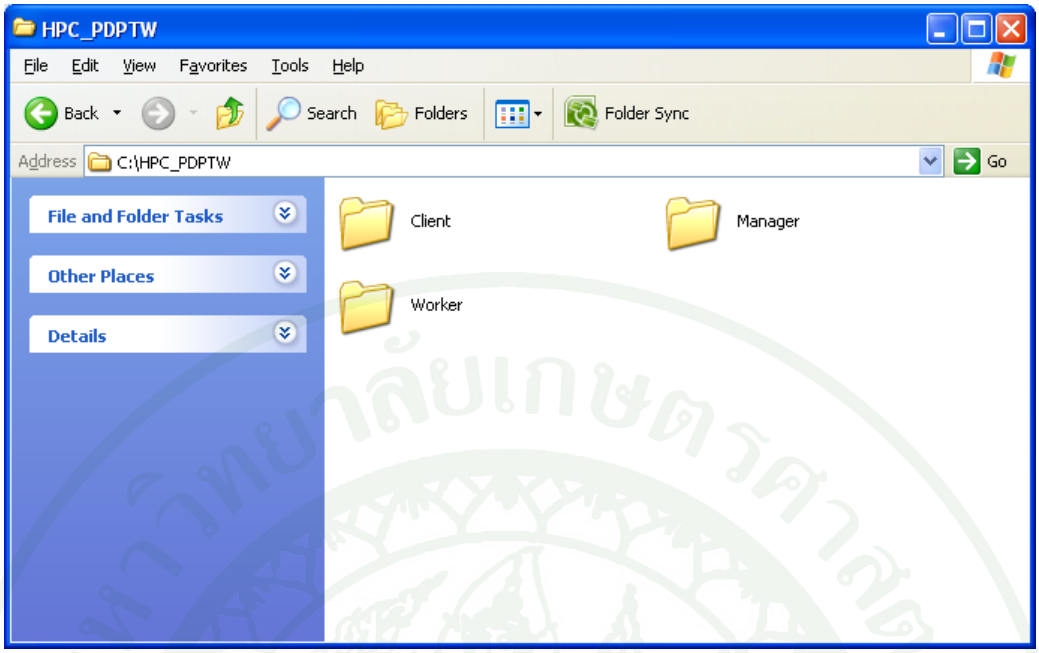
ภาพผนวกที่ ก7 การสร้างฐานข้อมูลสำหรับใช้ในการทำงานของ DPDPTWI บน MySQL Server

```
CREATE DATABASE opendss;
use opendss;
CREATE TABLE  worker_info (
  hostname  varchar(50) NOT NULL ,
  ip_address  varchar(50) NOT NULL,
  mac_address  varchar(45) NOT NULL,
  network_speed  varchar(45) NOT NULL,
  os_name  varchar(45) NOT NULL,
  memory_capacity  varchar(45) NOT NULL,
  storage_size  varchar(45) NOT NULL,
  processor_capacity  varchar(45) NOT NULL,
  processor_count  varchar(45) NOT NULL,
  compute_id  varchar(500) NOT NULL PRIMARY KEY
) ;
```

```
use opendss;
CREATE TABLE client_info (
  hostname  varchar(100)  NOT NULL ,
  ip_address  varchar(45)  NOT NULL,
  mac_address  varchar(45)  NOT NULL,
  network_speed varchar(45)  NOT NULL,
  os_name  varchar(45)  NOT NULL,
  memory_capacity  varchar(45)  NOT NULL,
  storage_size  varchar(45)  NOT NULL,
  processor_capacity  varchar(45)  NOT NULL,
  processor_count  varchar(45)  NOT NULL,
  client_id  varchar(200)  NOT NULL PRIMARY KEY
) ;
```

```
use opendss;
CREATE TABLE  job_info (
  job_id  int(10) unsigned NOT NULL AUTO_INCREMENT PRIMARY KEY,
  job_name  varchar(60) NOT NULL,
  client_request  varchar(200) NOT NULL,
  compute_responsible  varchar(200) NOT NULL,
  processor   int(10) unsigned NOT NULL,
  time_span   int(10) unsigned NOT NULL
) ;
```

10. การติดตั้ง DPDPTWI Manager โดยนำ Directory ชื่อ HPC_PDPTW วางไว้ที่ตำแหน่ง C:\ เท่านั้น ซึ่งภายใน Directory HPC_PDPTW ประกอบด้วย Directory ชื่อ Manager, Worker และ Client โดยแสดงดังภาพผนวกที่ ก8



ภาพผนวกที่ ก8 ตำแหน่งของการติดตั้ง DPDPTWI Manager

เวิร์คเกอร์ (Worker)

1. ติดตั้งระบบปฏิบัติการไมโครซอฟท์วินโดวส์ เอกซ์พี

2. ติดตั้ง Microsoft Robotics Developer Studio 2008 (โดยทำการติดตั้งตามขั้นตอนที่ 3 ถึง 6 ในหัวข้อการติดตั้งเมเนเจอร์)

3. การติดตั้ง DPDPTWI Worker โดยนำ Directory ชื่อ HPC_PDPTW วางไว้ที่ตำแหน่ง C:\ เท่านั้น ซึ่งภายใน Directory HPC_PDPTW ประกอบด้วย Directory ชื่อ Manager, Worker และ Client โดยแสดงดังภาพผนวกที่ ก8

ผู้ใช้ (User)

1. ติดตั้งระบบปฏิบัติการไมโครซอฟท์วินโดวส์ เอกซ์พี

2. ติดตั้ง Microsoft Office 2007

3. ติดตั้ง Microsoft Robotics Developer Studio 2008 (โดยทำการติดตั้งตามขั้นตอนที่ 3 ถึง 6 ในหัวข้อการติดตั้งเมเนเจอร์)

4. การติดตั้ง DPDPTWI Client โดยนำ Directory ชื่อ HPC_PDPTW วางไว้ที่ตำแหน่ง C:\ เท่านั้น ซึ่งภายใน Directory HPC_PDPTW ประกอบด้วย Directory ชื่อ Manager, Worker และ Client โดยแสดงดังภาพผนวกที่ ก8

ภาคผนวก ข

วิธีการใช้งานระบบ

วิธีการใช้งานระบบ

วิธีการใช้งานระบบ แบ่งเป็น 2 ส่วนคือ

1. วิธีการสร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

2. ขั้นตอนการใช้งานระบบ

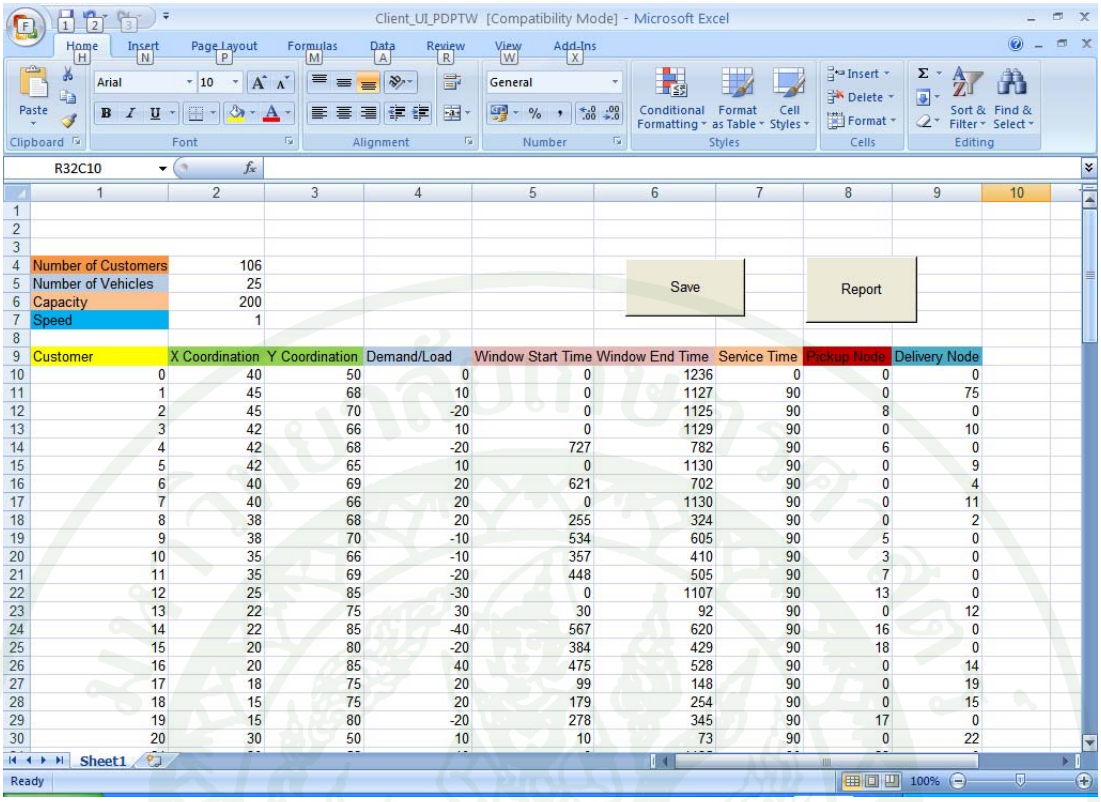
ซึ่งมีรายละเอียดดังต่อไปนี้

วิธีการสร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุดโดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

วิธีการสร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง สำหรับประมวลผลบนระบบ DPDPTWI ระบบได้จัดเตรียมแบบฟอร์มเอกสารเอ็กเซล (Excel Template) เพื่อใช้สำหรับการสร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เพื่อสะดวกแก่ผู้ใช้ แสดงตัวอย่างดังภาพผนวกที่ ข1 และมีขั้นตอนดังต่อไปนี้

1. ผู้ใช้สร้างงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง (กรณีสร้างงานบนโปรแกรม DPDPTWI Client) โดยระบุรายละเอียดของปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ให้ครบตามช่องที่ระบุไว้

2. เมื่อทำตามขั้นตอนที่ 1 เสร็จสิ้น ให้กดปุ่ม Save โปรแกรมจะบันทึกงานการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เป็นเอกสารชื่อ Input.txt ซึ่งจะเก็บไว้ที่ตำแหน่ง C:\HPC_PDPTW\Client\Input\ (กรณีสร้างงานบนโปรแกรม DPDPTWI Manager จะเก็บงานไว้ที่ตำแหน่ง C:\HPC_PDPTW\Manager\Input\) โดยเอกสาร Input.txt มีรูปแบบตรงตามระบบ DPDPTWI และใช้เป็นอินพุตของระบบ DPDPTWI ต่อไป

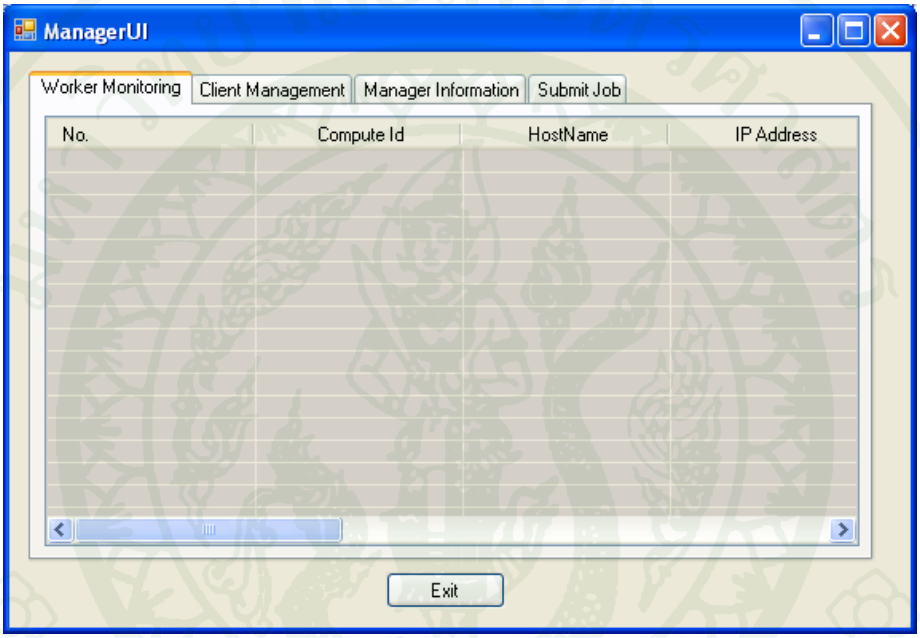


Customer	X Coordination	Y Coordination	Demand/Load	Window Start Time	Window End Time	Service Time	Pickup Node	Delivery Node
0	40	50	0	0	1236	0	0	0
1	45	68	10	0	1127	90	0	75
2	45	70	-20	0	1125	90	8	0
3	42	66	10	0	1129	90	0	10
4	42	68	-20	727	782	90	6	0
5	42	65	10	0	1130	90	0	9
6	40	69	20	621	702	90	0	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

ภาพผนวกที่ ข1 ตัวอย่างของงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

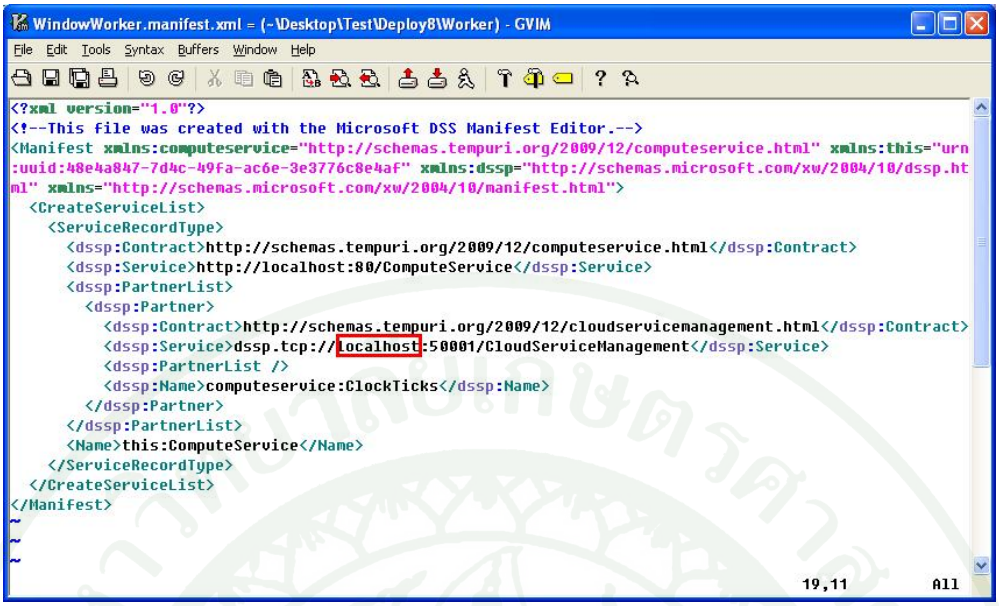
ขั้นตอนการใช้งานระบบ

1. เปิดโปรแกรม DPDPTWI Manager (เปิดโปรแกรมโดย Double Click ที่ตำแหน่ง C:\HPC_PDPTW\Manager\WindowManager) ในส่วนของ GUI บนเมเนเจอร์ซึ่งปรากฏโปรแกรม DPDPTWI Manager ดังภาพผนวกที่ ข2 จากนั้นให้รอการเชื่อมต่อจาก DPDPTWI Worker เพื่อทำการลงทะเบียนกับ DPDPTWI Manager



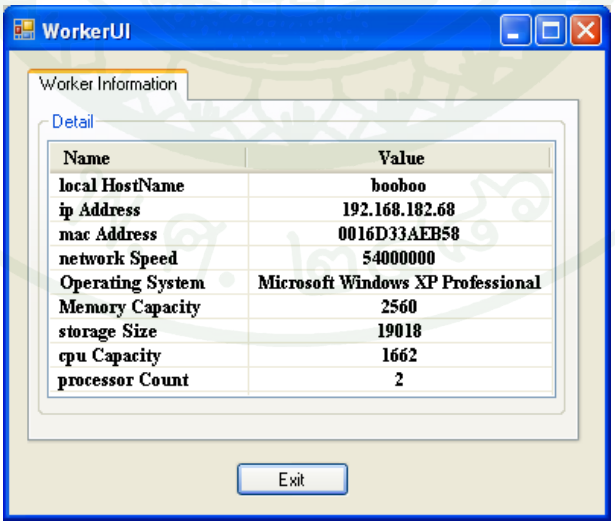
ภาพผนวกที่ ข2 โปรแกรม DPDPTWI Manager

2. โปรแกรม DPDPTWI Worker (เปิดโปรแกรมโดย Double Click ที่ตำแหน่ง C:\HPC_PDPTW\Worker\WindowWorker) ซึ่งก่อนการเปิดโปรแกรม DPDPTWI Worker นั้น ผู้ใช้จะต้องทำการแก้ไขไฟล์ที่ชื่อ WindowWorker.manifest โดยต้องระบุ IP Address หรือชื่อ Host name ของ Manager ลงไปยังตำแหน่งของ Tag ซึ่งแสดงดังภาพผนวกที่ ข3 โดยจากภาพผนวกที่ ข3 จะทำการแก้ไขชื่อจาก localhost เป็น IP Address หรือ ชื่อ Host name ของ Manager

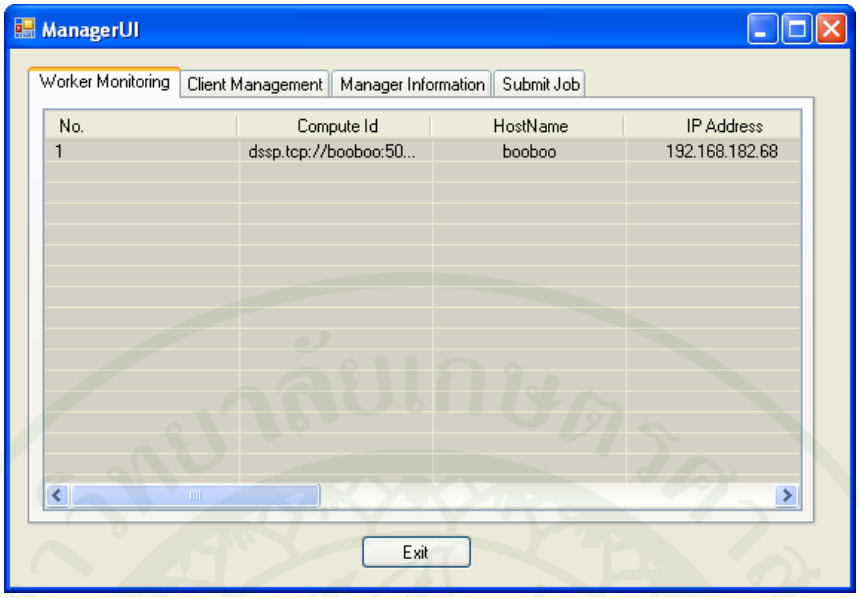


ภาพผนวกที่ ข3 การระบุ IP Address หรือ Host name ของ Manager บน Worker

เมื่อระบุ IP Address หรือ Host name ของ Manager เสร็จเรียบร้อยแล้ว เปิดโปรแกรม DPDPTWI Worker ซึ่งแสดงดังภาพผนวกที่ ข4 โดย Worker จะร้องขอเพื่อทำการลงทะเบียนกับทางฝั่ง Manager อัตโนมัติเมื่อเปิดโปรแกรม DPDPTWI Worker หากทำการลงทะเบียนสำเร็จ ข้อมูลของ Worker จะปรากฏบนหัวข้อ Worker Monitoring ของ Manager ดังภาพผนวกที่ ข5



ภาพผนวกที่ ข4 โปรแกรม DPDPTWI Worker



ภาพผนวกที่ ข5 การตรวจสอบข้อมูลและสถานะของ Worker ในระบบ DPDPTWI

3. หลังจากเสร็จสิ้นจากขั้นตอนที่ 1 และ 2 แล้ว ระบบมีความพร้อมในการใช้งาน ซึ่งผู้ใช้สามารถส่งงานเข้ามาประมวลผลบนระบบได้ โดยสามารถทำได้ 2 ทาง คือ ส่งงานผ่านทางโปรแกรม DPDPTWI Manager ในหัวข้อ Submit Job ดังภาพผนวกที่ ข6 และส่งงานผ่านทางโปรแกรม DPDPTWI Client ในหัวข้อ Submit Job ดังภาพผนวกที่ ข7 โดยการส่งงานเข้าสู่ระบบ มีข้อมูลที่ต้องระบุดังต่อไปนี้

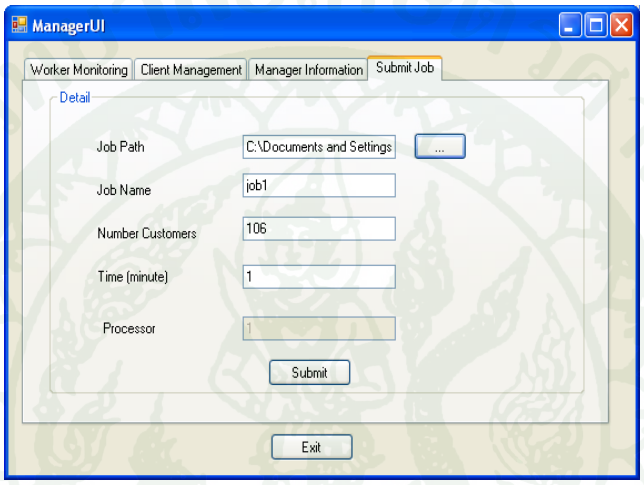
Job Path เป็นตำแหน่งของไฟล์งานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ต้องการส่งเข้าสู่ระบบ (กรณีสร้างงานบนโปรแกรม DPDPTWI Client เก็บงานไว้ที่ตำแหน่ง C:\HPC_PDPTW\Client\Input\), (กรณีสร้างงานบนโปรแกรม DPDPTWI Manager เก็บงานไว้ที่ตำแหน่ง C:\HPC_PDPTW\Manager\Input\)

Job Name เป็นชื่อของงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ที่ต้องการส่งเข้าสู่ระบบ

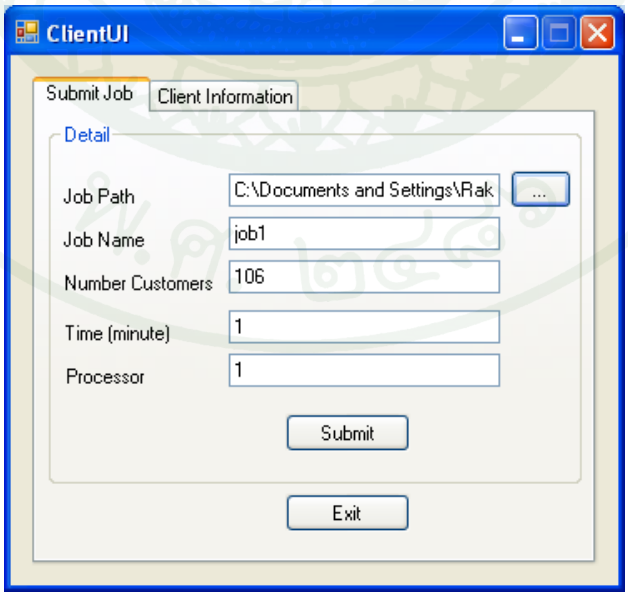
Number Customers เป็นจำนวนของลูกค้า (ร้านค้า) ทั้งหมดที่ต้องให้บริการ

Time (minute) เป็นเวลา (นาที) ที่ต้องการใช้ในการประมวลผลกับงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

Processor เป็นจำนวนของหน่วยประมวลผลที่ต้องการใช้ในการประมวลผลแบบขนานกับงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

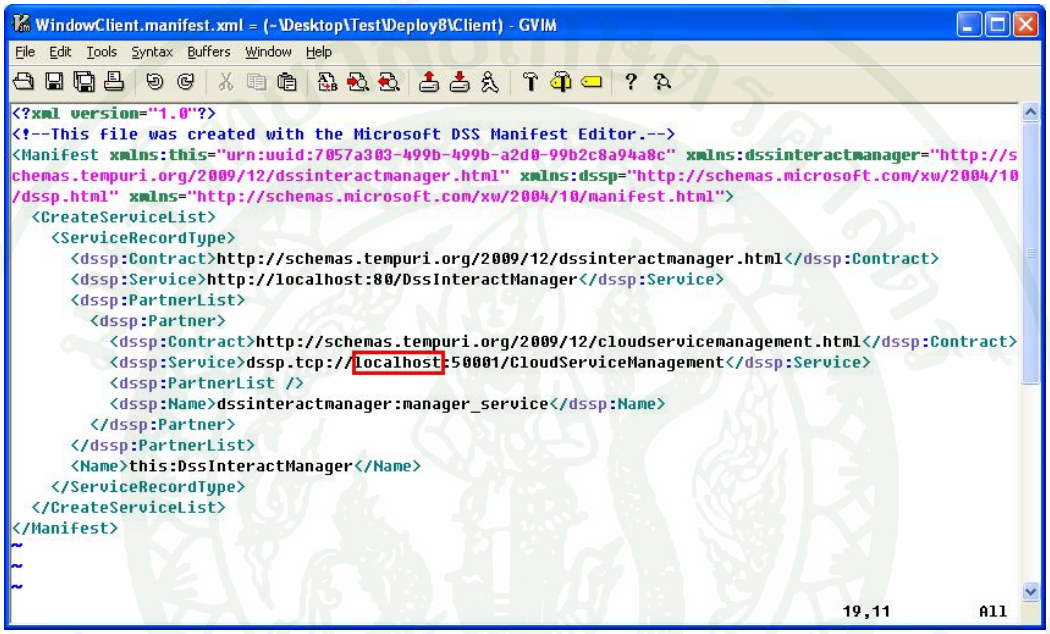


ภาพผนวกที่ ข6 การส่งงานเข้าสู่ระบบผ่านทางโปรแกรม DPDPTWI Manager



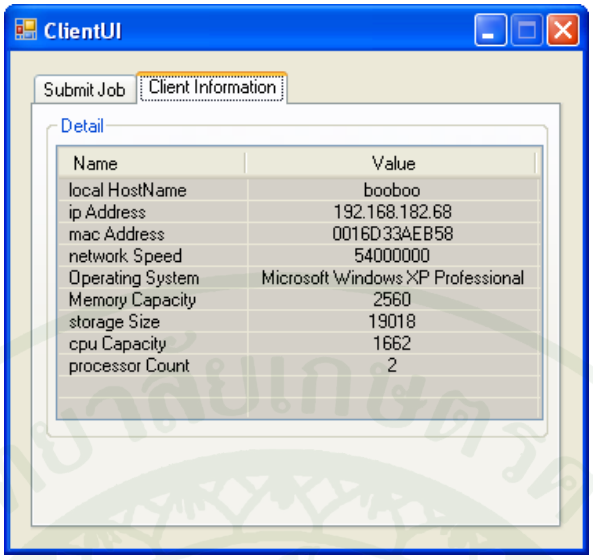
ภาพผนวกที่ ข7 การส่งงานเข้าสู่ระบบผ่านทางโปรแกรม DPDPTWI Client

4. โปรแกรม DPDPTWI Client (เปิดโปรแกรมโดย Double Click ที่ตำแหน่ง C:\HPC_PDPTW\Client\WindowClient) ซึ่งก่อนการเปิดโปรแกรม DPDPTWI Client นั้น ผู้ใช้จะต้องทำการแก้ไขไฟล์ที่ชื่อ WindowClient.manifest โดยต้องระบุ IP Address หรือชื่อ Host name ของ Manager ลงไปยังตำแหน่งของ Tag ซึ่งแสดงดังภาพผนวกที่ ข8 โดยจากภาพผนวกที่ ข8 จะทำการแก้ไขชื่อจาก localhost เป็น IP Address หรือ ชื่อ Host name ของ Manager

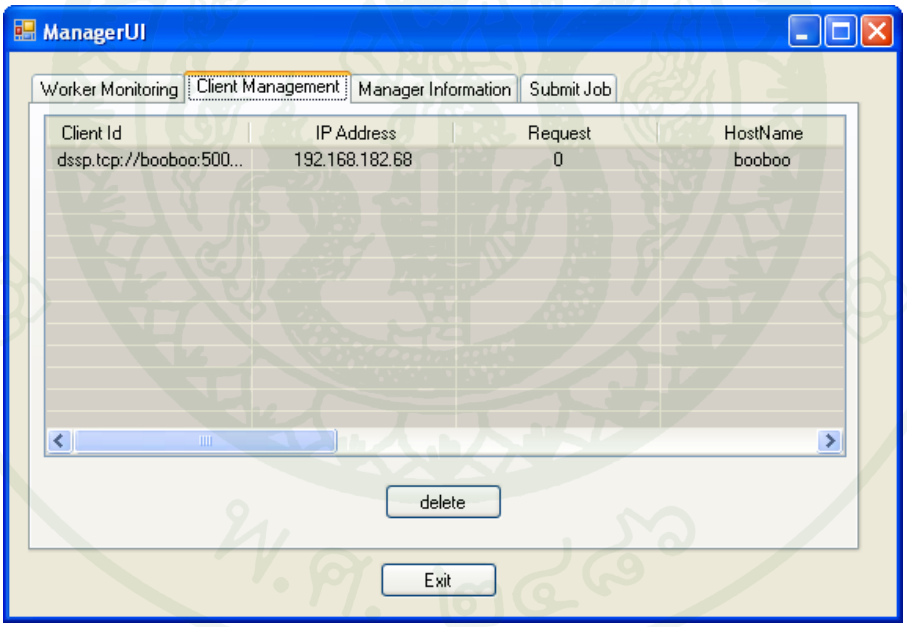


ภาพผนวกที่ ข8 การระบุ IP Address หรือ Host name ของ Manager บน Client

เมื่อทำการระบุ IP Address หรือ Host name ของ Manager เสร็จเรียบร้อยแล้ว ทำการเปิดโปรแกรม DPDPTWI Client ซึ่งแสดงดังภาพผนวกที่ ข9 โดย Client จะร้องขอเพื่อทำการลงทะเบียนกับทางฝั่ง Manager อัตโนมัติเมื่อเปิดโปรแกรม DPDPTWI Client หากทำการลงทะเบียนสำเร็จ ข้อมูลของ Client จะปรากฏบน หัวข้อ Client Management ของ Manager ดังภาพผนวกที่ ข10



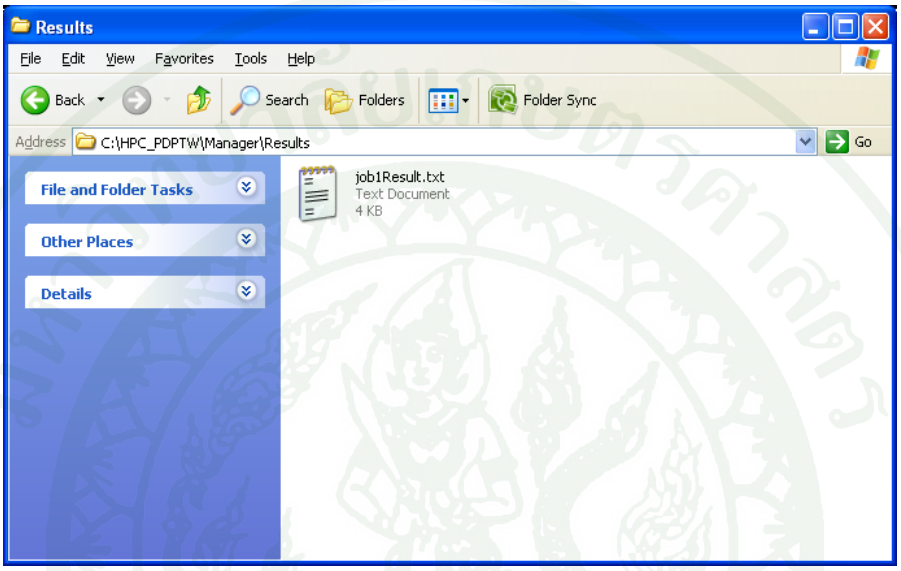
ภาพผนวกที่ ข9 โปรแกรม DPDPTWI Client



ภาพผนวกที่ ข10 การตรวจสอบข้อมูลและสถานะของ Client ในระบบ DPDPTWI

หัวข้อ Client Management ของโปรแกรม DPDPTWI Manager ผู้ดูแลระบบสามารถที่จะทำการลบ Client ได้ตามต้องการตามที่ผู้ดูแลระบบเห็นสมควร

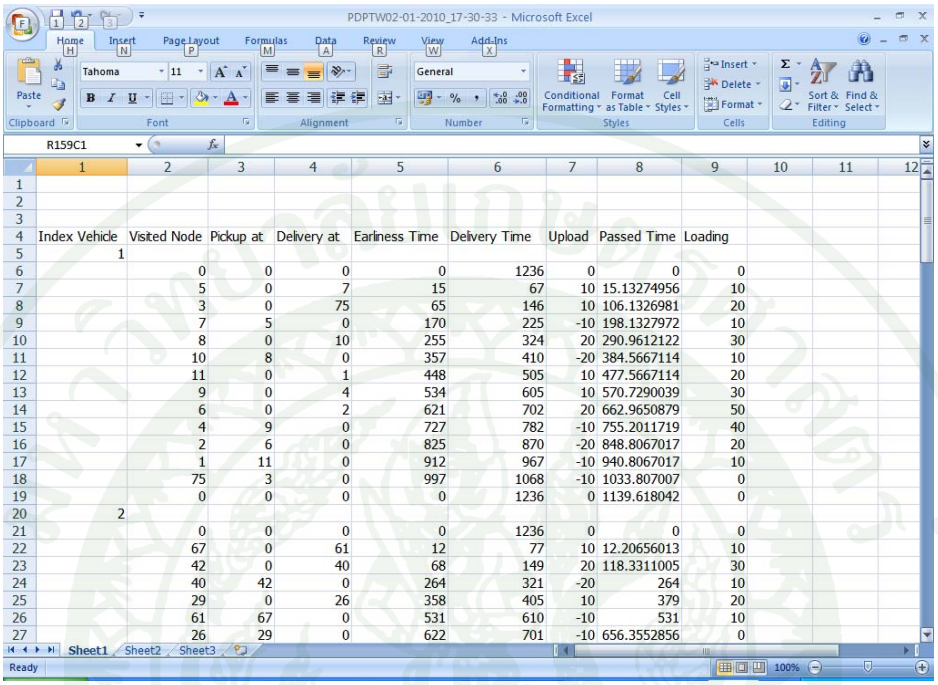
5. เมื่อส่งงานเข้าสู่ระบบแล้ว ระบบทำการประมวลผลงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่ง หลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง เสร็จสิ้นตามเวลาที่ผู้ใช้ระบุ ระบบจะส่งผลลัพธ์กลับไปยังผู้ร้องขอ (Client หรือ Manager) โดยผลลัพธ์จะถูกเก็บไว้ที่ Directory ชื่อ Results ดังภาพผนวกที่ ข11



ภาพผนวกที่ ข11 ผลลัพธ์ของการประมวลผลงานการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง

6. การแสดงผลลัพธ์ที่ได้จากการประมวลผลแบบขนานบนระบบ DPDPTWI สามารถทำได้โดยผ่านทางแบบฟอร์มเอกสารเอ็กเซลที่ระบบได้จัดเตรียมไว้ ซึ่งแบบฟอร์มเอ็กเซล มีปุ่ม Report เป็นปุ่มที่ใช้สำหรับแสดงผลลัพธ์ของการประมวลผลแบบขนานสำหรับการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีจุดรับ/ส่งหลายจุด โดยมีเงื่อนไขช่วงเวลาในการรับ/ส่ง ซึ่งผู้ใช้สามารถเปิดดูผลลัพธ์โดยการกดปุ่ม Report จากนั้นเลือกตำแหน่ง (ที่ตำแหน่ง C:\HPC_PDPTW\Client\Results) ของเอกสารผลลัพธ์ (กรณีส่งงานผ่านทางโปรแกรม DPDPTWI Client) โดยชื่อของเอกสารผลลัพธ์จะมีรูปแบบคือ “**ชื่อที่ผู้ใช้ระบุ (ชื่อตรงตามช่อง Job Name ที่ผู้ใช้ระบุ)**” ตามด้วย **Result.txt** ซึ่งได้แสดงตัวอย่างดังภาพผนวกที่ ข11 กรณีผู้ใช้ระบุจำนวนของโปรเซสเซอร์ มากกว่า 1 โปรเซสเซอร์ รูปแบบของเอกสารผลลัพธ์จะมีรูปแบบ คือ “**ชื่อที่ผู้ใช้ระบุ (ชื่อตรงตามช่อง Job Name ที่ผู้ใช้ระบุ)**” ตามด้วย “**_ตัวเลข(ค่าเริ่มต้นที่ 0)**” ตามด้วย **Result.txt** ตัวอย่างเช่น กรณีที่ผู้ใช้ระบุจำนวนโปรเซสเซอร์ เท่ากับ 2 เอกสารผลลัพธ์ที่ได้จะเป็นดังนี้ job1_0Results.txt, job1_1Result.txt เป็นต้น

โดยผลลัพธ์ที่ได้รับการประมวลผลแบบขนานบนระบบ DPDPTWI ที่แสดงผลลัพธ์ผ่านทางแบบฟอร์มเอ็กเซลแสดงดังภาพผนวกที่ ข12



PDPTW02-01-2010_17-30-33 - Microsoft Excel

Index Vehicle	Visited Node	Pickup at	Delivery at	Earliness Time	Delivery Time	Upload	Passed Time	Loading
1								
	0	0	0	0	1236	0	0	0
	5	0	7	15	67	10	15.13274956	10
	3	0	75	65	146	10	106.1326981	20
	7	5	0	170	225	-10	198.1327972	10
	8	0	10	255	324	20	290.9612122	30
	10	8	0	357	410	-20	384.5667114	10
	11	0	1	448	505	10	477.5667114	20
	9	0	4	534	605	10	570.7290039	30
	6	0	2	621	702	20	662.9650879	50
	4	9	0	727	782	-10	755.2011719	40
	2	6	0	825	870	-20	848.8067017	20
	1	11	0	912	967	-10	940.8067017	10
	75	3	0	997	1068	-10	1033.807007	0
	0	0	0	0	1236	0	1139.618042	0
2								
	0	0	0	0	1236	0	0	0
	67	0	61	12	77	10	12.20656013	10
	42	0	40	68	149	20	118.3311005	30
	40	42	0	264	321	-20	264	10
	29	0	26	358	405	10	379	20
	61	67	0	531	610	-10	531	10
	26	29	0	622	701	-10	656.3552856	0

ภาพผนวกที่ ข12 การแสดงผลลัพธ์จากการประมวลแบบขนานบนระบบ DPDPTWI

ประวัติการศึกษา และการทำงาน

ชื่อ–นามสกุล	นายรักพงค์ แก้วพวง
วัน เดือน ปี ที่เกิด	วันที่ 21 เดือนธันวาคม พ.ศ. 2526
สถานที่เกิด	จังหวัดศรีสะเกษ
ประวัติการศึกษา	วท.บ. (วิทยาการคอมพิวเตอร์) มหาวิทยาลัยอุบลราชธานี