ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)

ปริญญา

วิศวกรรมอุตสาหการ | วิศวกรรมอุตสาหการ

สาขา | **ภาควิชา**

เรื่อง การพัฒนาไลบารี่การควบคุมโปรแกรมจำลองแบบคู่ขนานสำหรับโปรแกรมภาษาคอมพิวเตอร์ทั่วไป

A Development of Parallel Simulation Control Library for General Programming Language

นามผู้วิจัย นายอภิศักดิ์ วิทยาประภากร

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(อาจารย์พรเทพ อนุสสรนิติสาร, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

(ผู้ช่วยศาสตราจารย์จุฑา พิชิตลำเค็ญ, Ph.D.)

หัวหน้าภาควิชา

(รองศาสตราจารย์อนันต์ มุ่งวัฒนา, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

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

วิทยานิพนธ์

เรื่อง

การพัฒนาไลบารี่การควบคุมโปรแกรมจำลองแบบคู่ขนานสำหรับโปรแกรมภาษาคอมพิวเตอร์ทั่วไป

A Development of Parallel Simulation Control Library for General Programming Language

โดย

นายอภิศักดิ์ วิทยาประภากร

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)

พ.ศ. 2555

อภิศักดิ์ วิทยาประภากร 2555: การพัฒนาไลบารี่การควบคุมโปรแกรมจำลองแบบคู่ขนานสำหรับโปรแกรมภาษาคอมพิวเตอร์ทั่วไป ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ) สาขาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: อาจารย์พรเทพ อนุสสรนิติสาร, Ph.D. 116 หน้า

ในปัจจุบันการจำลองสถานการณ์เป็นทางเลือกหนึ่งที่ใช้สำหรับการวิเคราะห์ระบบที่ซับซ้อน โดยเฉพาะในกรณีของการค้นหาคำตอบร่วมกับการจำลองสถานการณ์ (Simulation Optimization) จำเป็นต้องใช้เวลาที่ยาวนานในการหาคำตอบที่ต้องการ เนื่องจากระบบที่ซับซ้อนจะมีพารามิเตอร์เป็นจำนวนมากที่ต้องเปลี่ยนแปลงไปสู่คำตอบที่ดีที่สุด จากปัญหาที่ได้กล่าวข้างต้น เป็นปัญหาที่ส่งผลในเรื่องของการใช้เวลาที่ยาวนานในการประมวลผลหรือค้นหาคำตอบที่ต้องการ ดังนั้นทางผู้วิจัยจึงได้เสนอทางเลือกหนึ่งสำหรับการแก้ปัญหาที่เรียกว่าการประมวลผลแบบขนานหรือการประมวลผลแบบกระจายมาเป็นเครื่องมือในการลดเวลาสำหรับการแก้ปัญหา งานวิจัยชิ้นนี้จึงมีจุดมุ่งหมายในการสร้างเครื่องมือควบคุมการประมวลผลแบบขนานเพื่อเป็นทางเลือกหนึ่งสำหรับนักวิจัยที่ตัดสินใจใช้การประมวลผลแบบขนานมาช่วยในการค้นหาคำตอบ เพื่อลดเวลาในการวิเคราะห์ระบบที่ซับซ้อนและค้นหาคำตอบที่ต้องการ โดยเครื่องมือที่พัฒนามาสำหรับงานวิจัยชิ้นนี้จะมีทั้งหมด 2 ส่วนคือ

1. แอพพลิเคชั่นควบคุมการประมวลผลแบบขนานสำหรับโปรแกรม Arena ควบคุมการกระจายรอบทำซ้ำกรณีจำลองสถานการณ์ทำซ้ำหลายรอบ และควบคุมการเปลี่ยนแปลงค่าพารามิเตอร์ในตัวแบบจำลองกรณีที่มีค่าพารามิเตอร์ต่างกัน

2. ไลบรารีควบคุมการประมวลผลแบบขนานภาษา Visual Basic สำหรับนักวิจัยที่ต้องการพัฒนาอัลกอริทึมแบบขนาน

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Apisak Vittayaprapakorn 2012: A Development of Parallel Simulation Control Library for General Programming Language. Master of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering. Thesis Advisor: Mr. Pornthep Anussornnitisarn, Ph.D. 116 pages.

Nowadays, Simulation is one of the effective methods to analyze performance of the complex system. However ,using simulation to find an optimal setting sequence of simulation. It needs long operation time due to number of parameters of the system. As the mentioned problem, researcher proposed an alternative method which is called parallel computing to reduce time. In this research, we proposed a parallel simulation on a way to reduce time by create the tools for researchers who need parallel and distributed computing method. The developed tools of this research are separated into two types.

1. Parallel simulation application for Arena software. It is used to distribute number of replication and control parameters value change as per the simulated model in case of different parameter value

2.Parallel simulation control library for Visual Basic Language. It is used as tool for the researchers who are interested parallel algorithm development.

_____ _____ __ / __ / __

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ ดร.พรเทพ อนุสสรนิติสาร อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก และ ผศ.ดร.จุฑา พิชิตลำเค็ญ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ให้คำปรึกษาในการเรียน การค้นคว้าวิจัย ตลอดจนการตรวจแก้ไขวิทยานิพนธ์จนกระทั่งเสร็จสมบูรณ์ และกราบขอบ พระคุณ ผศ.ดร.พีรวัฒน์ วัฒนพงศ์ ประธานการสอบวิทยานิพนธ์ และ รศ.ดร.ยุทธชัย บรรเทิงจิตรผู้ทรงคุณวุฒิภายนอกที่ได้ให้ความกรุณาตรวจแก้ไขวิทยานิพนธ์ให้สมบูรณ์ยิ่งขึ้น

ขอกราบขอบพระคุณอาจารย์ภาควิชาวิศวกรรมอุตสาหการ มหาวิทยาลัยเกษตรศาสตร์ทุกท่านที่ได้อบรมสั่งสอนและมอบความรู้อันเป็นประโยชน์ยิ่งในการนำไปใช้ประโยชน์ต่อไป

ขอขอบคุณ นายธนพันธ์ คงทอง ที่เอื้อเฟื้อโจทย์สำหรับทำการทดลอง

ขอขอบคุณ นางสาววาสนา เอมวัฒน์ ผู้ให้คำปรึกษาและช่วยเหลือในการให้ความรู้ในการเขียนโปรแกรมและไลบรารีสำหรับวิทยานิพนธ์ชิ้นนี้

สุดท้ายนี้ความดีหรือประโยชน์อันใดที่เกิดจากการศึกษาค้นคว้าครั้งนี้ ผู้วิจัยขอมอบแด่บิดา มารดา ครู อาจารย์ และผู้มีพระคุณทุกท่าน ที่ได้ให้ความกรุณาและให้กำลังใจมาโดยตลอด

อภิศักดิ์ วิทยาประภากร

มีนาคม 2555

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

ภาพผนวกที่

สารบัญภาพ (ต่อ)

การพัฒนาไลบารี่การควบคุมโปรแกรมจำลองแบบคู่ขนานสำหรับโปรแกรมภาษาคอมพิวเตอร์ทั่วไป

A Development of Parallel Simulation Control Library for General Programming Language

คำนำ

ในปัจจุบันการจำลองสถานการณ์เป็นทางเลือกหนึ่งที่ใช้สำหรับการวิเคราะห์ระบบที่ซับซ้อน ซึ่งจำนวนรอบของการจำลองระบบมีจำนวนรอบที่มาก นอกจากนั้น ในกรณีของการค้นหาคำตอบที่เราต้องการหรือการใช้อัลกอริทึมร่วมกับการจำลองสถานการณ์ เช่น การค่าที่ดีที่สุดของระบบ อาจต้องใช้เวลาที่ยาวนานเนื่องจากมีค่าคำตอบที่เป็นไปได้นั้นมีจำนวนมาก จากปัญหาที่ได้กล่าวข้างต้น เห็นได้ว่าเป็นปัญหาที่ส่งผลในเรื่องของการใช้เวลาที่ยาวนานในการประมวลผลหรือหาคำตอบ ดังนั้นผู้วิจัยจึงได้เสนอทางเลือกหนึ่งสำหรับการแก้ปัญหาที่เรียกว่าการประมวลผลแบบขนานหรือการประมวลผลแบบกระจายมาเป็นเครื่องมือในการลดเวลาสำหรับการแก้ปัญหา

คำนิยามของการประมวลผลแบบขนาน คือ “การใช้เครื่องประมวลผลกลุ่มหนึ่งในการคำนวณหรือแก้ปัญหาจากโจทย์เดียวกัน เพื่อให้ได้คำตอบที่เร็วขึ้นกว่าการคำนวณบนเครื่องประมวลผลเครื่องเดียว” แนวความคิดดังกล่าวได้จากการศึกษาธรรมชาติของการเขียนโปรแกรม ซึ่งพบว่าโจทย์หรือปัญหาส่วนมากสามารถแตกออกได้เป็นหลายโมดูลย่อย และแต่ละโมดูลสามารถประมวลผลพร้อมกันได้ โดยอาศัยการแลกเปลี่ยนข้อมูลระหว่างเครื่องประมวลผล การประมวลผลแบบขนานจึงเปรียบได้กับการทำงานกลุ่ม ซึ่งต้องอาศัยทั้งประสิทธิภาพของสมาชิกในกลุ่มและการสื่อสารที่ดีระหว่างสมาชิก

Visual Basic เป็นภาษาคอมพิวเตอร์ทั่วไปตัวหนึ่งที่ง่ายต่อการศึกษาและนำมาพัฒนางานที่มีอยู่ให้เป็นระบบงานที่มีประสิทธิภาพ พร้อมทั้งมีความเรียบง่ายของภาษาที่สอดคล้องกับแนวทางการพัฒนาเทคโนโลยีของไมโครซอฟท์ จึงทำให้ผู้วิจัยเลือกใช้ Visual Basic มาพัฒนาแอพพลิเคชั่นและไลบรารีสำหรับการเชื่อมต่อเพื่อประมวลผลแบบขนาน

จากปัญหาและทางออกของปัญหาที่กล่าวไว้ข้างต้น ทำให้งานวิจัยชิ้นนี้มีจุดมุ่งหมายที่จะนำแนวคิดในเรื่องของการประมวลผลแบบขนานมาลดเวลาการค้นหาคำตอบที่เราต้องการหรือการใช้อัลกอริทึมร่วมกับการจำลองสถานการณ์ โดยนำแนวคิดนี้มาพัฒนาสร้างแอพพลิเคชั่นและไลบารี่เป็นเครื่องมือโดยใช้ภาษา Visual Basic สำหรับทำการเชื่อมต่อระหว่างคอมพิวเตอร์ให้สามารถทำการประมวลผลหาคำตอบแบบขนานได้นั่นเอง

วัตถุประสงค์

งานวิจัยฉบับนี้มีวัตถุประสงค์ที่จะทำการศึกษาดังนี้ คือ

1. พัฒนาไลบารี่โดยภาษา Visual Basic สำหรับเชื่อมโยงระหว่างคอมพิวเตอร์ เพื่อสามารถทำการประมวลผลอัลกอริธึมแบบขนานได้

2. พัฒนาแอพพลิเคชั่นโดยภาษา Visual Basic สำหรับเชื่อมโยงระหว่างคอมพิวเตอร์ เพื่อสามารถทำการประมวลผลโปรแกรม Arena แบบขนานได้

ขอบเขตของงานวิจัย

งานวิจัยนี้เพื่อพัฒนาเครื่องมือสำหรับการเชื่อมต่อระหว่างคอมพิวเตอร์ให้สามารถทำการประยุกต์สำหรับการประมวลผลแบบขนานได้ โดยการพัฒนาจะแบ่งออกเป็น 2 ส่วนคือ ส่วนของแอพพลิเคชั่นที่ใช้สำหรับการประมวลผลแบบขนานสำหรับโปรแกรม Arena และส่วนของไลบรารีที่ใช้สำหรับการเขียนโปรแกรมเพื่อควบคุมการประมวลผลแบบขนานโดยมีรายละเอียดของขอบเขตดังต่อไปนี้

1. มุ่งเน้นในการพัฒนาการสื่อสารและส่งผ่านข้อมูลระหว่างคอมพิวเตอร์ภายใต้ระบบ Client / Server โดยใช้ Visual Basic Studio Version 2008 Enterprise

2. มุ่งเน้นในเรื่องของการสร้างแอพพลิเคชั่นและไลบรารีสำหรับ Visual Basic Studio Version 2008 Enterprise เพื่อเป็นเครื่องมือสำหรับการประมวลผลแบบขนาน มิได้ประเมินคุณภาพของ Search Algorithms หรือมุ่งเน้นความถูกต้องในเชิงสถิติของคำตอบที่ได้

3. สมมติว่าระบบมีความปลอดภัย ไม่ได้ครอบคลุมการทำงานของระบบรักษาความปลอดภัย เช่น Firewall, Antivirus

การตรวจเอกสาร

ในการพัฒนาเครื่องมือในการจำลองสถานการณ์ ได้ตรวจสอบเอกสารที่เกี่ยวข้อง โดยแบ่งเป็น การประมวลผลแบบขนาน ความสัมพันธ์ระหว่างความสามารถในการประมวลผลและจำนวนหน่วยประมวลผลแบบขนาน การใช้ไลบารี่ เครือข่ายเชื่อมต่อ การสื่อสารผ่านเครือข่ายอินเทอร์เน็ตแบบ TCP/ IP และ งานวิจัยที่เกี่ยวข้อง

การประมวลผลแบบขนาน

หนังสือเทคโนโลยีการประมวลผลแบบขนานและแบบกระจาย สำนักพิมพ์ท้อป จำกัด โดย รศ.ดร.ธีรณี อจลากุลหน้า 1-2 ได้กล่าวถึงการประมวลผลแบบขนานไว้ดังนี้

ในช่วงไม่กี่ทศวรรษที่ผ่านมา ความก้าวหน้าทางด้านเทคโนโลยี การประมวลผลที่เกิดขึ้นในอัตราที่สูงมาก ความเร็วของไมโครโพรเซสเซอร์เพิ่มจาก 40 MHz ในปี พ.ศ. 2531 เป็น 3.0 GHz ใน ปี พ.ศ. 2550 ความก้าวหน้าของเทคโนโลยีทางด้านการประมวลผลดังกล่าวส่งผลให้เทคโนโลยีการพัฒนาโปรแกรมในสาขาต่างๆเกิดขึ้นอย่างรวดเร็วไปด้วย โปรแกรมในปัจจุบันใช้ทรัพยากรมากกว่าในอดีตอย่างมาก จนกระทั่งเทคโนโลยีไมโครเซสเซอร์ตามไม่ทัน การประมวลผลแบบขนานจึงเป็นศาสตร์การเขียนโปรแกรมที่ได้รับความนิยมสูงในหมู่นักวิทยาศาสตร์และวิศวกรที่ต้องการความเร็วในการประมวลผลสูงในระดับที่เครื่องคอมพิวเตอร์เครื่องใดเครื่องหนึ่งไม่สามารถรองรับได้ นอกจากนี้เทคนิคการประมวลผลแบบขนานยังสามารถลดเวลาในการแก้ปัญหาขนาดใหญ่ (ซึ่งปกติต้องใช้เวลานานในการคำนวณ) ลง ทำให้ได้ผลข้อมูลซึ่งสามารถนำไปใช้ได้ทันท่วงที

คำนิยามของการประมวลผลแบบขนาน คือ “การใช้เครื่องประมวลผลกลุ่มหนึ่งในการคำนวณหรือแก้ปัญหาจากโจทย์เดียวกัน เพื่อให้ได้คำตอบที่เร็วขึ้นกว่าการคำนวณบนเครื่องประมวลผลเครื่องเดียว” แนวความคิดดังกล่าวได้จากการศึกษาธรรมชาติของการเขียนโปรแกรม ซึ่งพบว่าโจทย์หรือปัญหาส่วนมากสามารถแตกออกได้เป็นหลายโมดูลย่อย และแต่ละโมดูลสามารถประมวลผลพร้อมกันได้โดยอาศัยการแลกเปลี่ยนข้อมูลระหว่างเครื่องประมวลผล การประมวลผลแบบขนานจึงเปรียบได้กับการทำงานกลุ่ม ซึ่งต้องอาศัยทั้งประสิทธิภาพของสมาชิกในกลุ่มและการสื่อสารที่ดีระหว่างสมาชิก

การพัฒนาโปรแกรมแบบขนานมักถูกมองว่าเป็นเรื่องที่ยากและต้องใช้ความพยายามสูงแต่อย่างไรก็ดี การประมวลผลแบบขนานในปัจจุบันได้รับการยอมรับว่าเป็นเทคนิคมาตรฐานที่สามารถเพิ่มความเร็วและประสิทธิภาพในการประมวลผลให้กับเครื่องประมวลผลที่มีอยู่ อีกทั้งยังเป็นการใช้เครื่องประมวลให้เต็มประสิทธิภาพอีกด้วย

ความสัมพันธ์ระหว่างความสามารถในการประมวลผลและจำนวนหน่วยประมวลผลแบบขนาน

รศ.ดร.ธีรณี อจลากุล ได้กล่าวถึงกฎของแอมดาห์ลและกฎของกุสตาฟสัน-บาร์ซิสในหนังสือเทคโนโลยีการประมวลผลแบบขนานเกี่ยวกับ ความสัมพันธ์ระหว่างความสามารถในการประมวลผลและจำนวนหน่วยประมวลผลแบบขนานไว้ดังนี้

กฎของแอมดาห์ล (Amdahl's Law)

กฎของแอมดาห์ลอธิบายว่า หากมีการปรับปรุงบางส่วนของระบบคอมพิวเตอร์ให้มีประสิทธิภาพ ดีขึ้นแล้ว การปรับปรุงนั้นจะส่งผลต่อประสิทธิภาพโดยรวมของระบบมากน้อยเพียงใด เช่น ถ้าหากเพิ่มจำนวนหน่วยประมวลผล ความเร็วในการประมวลผลจะเพิ่มขึ้นอย่างไร โดยทั่วไป กฎของแอมดาห์ลจะถูกใช้เพื่อทำนายค่าสปีดอัพสูงสุดที่เป็นไปได้ทางทฤษฎี ค่าสปีดอัพดังกล่าวจะถูกจำกัดด้วยสัดส่วนของโปรแกรมที่ต้องประมวลผลแบบตามลำดับ ดังนั้นให้สัญลักษณ์ f แทนค่าอัตราส่วนของโปรแกรมที่ต้องถูกประมวลผลแบบตามลำดับเทียบกับโปรแกรมทั้งหมด กล่าวคือ $f = \frac{\sigma(n)}{\sigma(n)+\mu(n)}$

จากสมการสปีดอัพ $s \leq \frac{\sigma(n)+\mu(n)}{\sigma(n)+\frac{\mu(n)}{p+T_0}}$ เราทราบอยู่แล้วว่า T_0 นั้นจะมีค่ามากกว่า 0 เสมอ ดังนั้นสมการจึงสามารถเขียนใหม่ได้ว่า $s \leq \frac{\sigma(n)+\mu(n)}{\sigma(n)+\frac{\mu(n)}{p+T_0}} \leq \frac{\sigma(n)+\mu(n)}{\sigma(n)+\mu(n)/p}$

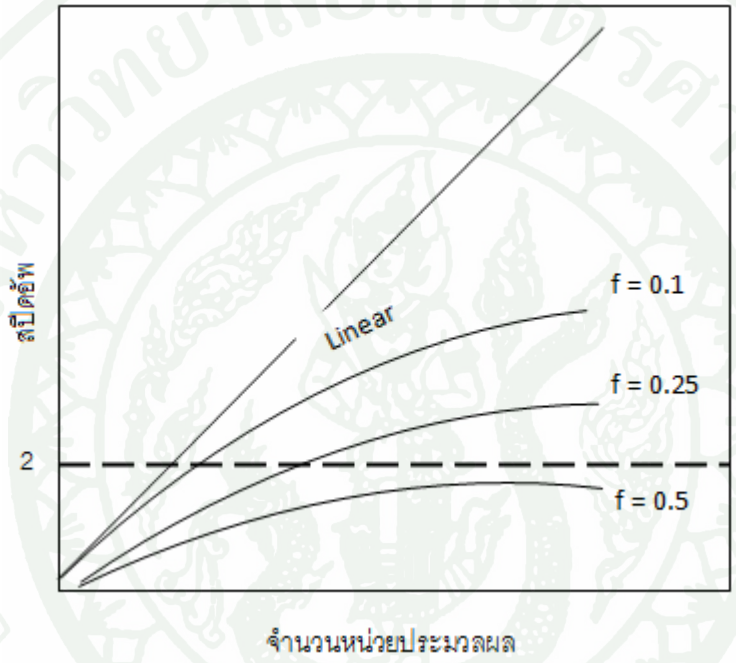
หากแทนค่า f เข้าไปในสมการดังกล่าวจะได้

$$s \leq \frac{\sigma(n)/f}{\sigma(n) + \sigma(n)\left(\frac{1}{f}-1\right)/p}$$

$$s \leq \frac{1/f}{1+\left(\frac{1}{f}-1\right)p} = \frac{1}{f+(1-f)/p}$$

กฎของแอมดาห์ลกล่าวว่า หาก f แสดงค่าของโปรแกรมที่ต้องประมวลผลแบบตามลำดับ ซึ่งมีค่าอยู่ระหว่าง 0 และ 1$(0 \leq f \leq 1)$ ค่าสปีดอัพที่สูงที่สุดเท่าที่จะเป็นไปได้ทางทฤษฎีเมื่อใช้หน่วยประมวลผลจำนวน p หน่วยในการประมวลผล คือ

$$s \leq \frac{1}{f + (1-f)/p}$$



ภาพที่ 1 แสดงตัวอย่างของสปีดอัพที่ได้สำหรับอัตราส่วนต่างๆ

ตัวอย่างเช่น ถ้า 50% ของโปรแกรมต้องประมวลผลแบบตามลำดับของโปรแกรมต้องประมวลผลแบบตามลำดับ $(f = 0.5)$ ค่าสปีดอัพที่ได้จากกฎของแอมดาห์ลจะมีค่าได้มากที่สุดเท่ากับ 2 ไม่ว่าจะใช้หน่วยประมวลผลช่วยคำนวณจำนวนมากเท่าใดก็ตาม $(p \to \infty)$ ทั้งนี้เนื่องจากโปรแกรมสามารถใช้ประโยชน์จากการประมวลผลขนานได้เพียงครึ่งหนึ่งเท่านั้น

$$s \leq \frac{1}{0.5 + \frac{(1-0.5)}{p}}$$

$$\lim_{p \to \infty} \frac{1}{0.5 + \frac{(1-0.5)}{p}} = 2$$

กฎของแอมดาห์ลมองข้ามโอเวอร์เฮดของการประมวลผลขนาน ดังนั้นผลการทำนายสปีดอัพจึงมักจะได้ค่ามากกว่าความเป็นจริงเสมอ ซึ่งถือเป็นข้อจำกัดของสมการเลยทีเดียว อย่างไรก็ตาม โดยทั่วไปแล้วค่าโอเวอร์เฮดมักจะมีความซับซ้อนต่ำกว่าส่วนของโปรแกรมที่ประมวลผลขนานกัน $(T_0 < \mu(n))$ ดังนั้นการเพิ่มขนาดของโจทย์ปัญหาจึงเพิ่มเวลาในการประมวลผลมากกว่าเพิ่มโอเวอร์เฮดของการส่งข้อมูล ด้วยจำนวนหน่วยประมวลผลคงที่สปีดอัพจึงจะเพิ่มขึ้นตามขนาดของโจทย์ปัญหา

กฎของกุสตาฟสัน-บาร์ซิส (Gustafson-Barsis's Law)

กฎของแอมดาห์ลตั้งสมมติฐานว่า ประเด็นหลักของการประมวลผลแบบขนานคือ การลดเวลาในการประมวลผล สมการจึงแสดงผลของเวลาในการประมวลผลที่ลดลงเมื่อมีการเพิ่มจำนวนหน่วยประมวลผลเข้ามาในระบบ อย่างไรก็ตามในหลายกรณีเป้าหมายหลักอาจมิได้อยู่ที่เวลาการประมวลผลเพียงอย่างเดียว แต่อยู่ที่การหาค่าผลลัพธ์ที่ตอบโจทย์ปัญหาให้ได้ดีที่สุดในเวลาที่จำกัด เราจะเห็นตัวอย่างได้จากโปรแกรมการสร้างแบบจำลองประเภทต่างๆ (Simulation Model) การใช้เครื่องประมวลผลจำนวนมากจะสามารถสร้างคำตอบที่ใกล้เคียงกับความเป็นจริงได้มากที่สุดในระยะเวลาจำกัด

กฎของกุสตาฟสัน-บาร์ซิสจะทำนายความสามารถในการแก้โจทย์ปัญหาที่มีขนาดเพิ่มขึ้นเมื่อเพิ่มจำนวนหน่วยประมวลผลในขณะที่เวลาคงที่

> กฎของกุสตาฟสัน-บาร์ซิสกล่าวว่า ในการแก้โจทย์ปัญหาที่มีขนาดเท่ากับ n โดยใช้หน่วยประมวลผลจำนวน p หน่วย หากให้ f แทนค่า สัดส่วนของโปรแกรมที่ต้องประมวลผลแบบตามลำดับ ค่าสปีดอัพที่สูงที่สุดที่สามารถทำได้ คือ
>
> $$s \leq p + (1 - p)f$$

กฎของกุสตาฟสัน-บาร์ซิสจะประมาณการว่า หากใช้เทคนิคการประมวลผลแบบขนานแล้ว เวลาในการคำนวณจะลดลงได้ในอัตราเท่าใด เมื่อเทียบกับการประมวลผลตามลำดับในกรณีทั่วไปผู้พัฒนาโปรแกรมไม่สามารถตั้งสมมติฐานว่าการใช้หน่วยประมวลผล p หน่วยจะสามารถเพิ่มความเร็วในการคำนวณได้ p เท่า เนื่องจากโอเวอร์เฮดของการประมวลผลแบบขนานที่เกิดขึ้น

กฎนี้จึงกำหนดให้เวลาในการคำนวณคงที่และทำการศึกษาขนาดของโจทย์ปัญหาเพิ่มขึ้น การเพิ่มจำนวนหน่วยประมวลผลจึงทำให้ระบบคอมพิวเตอร์สามารถแก้โจทย์ปัญหาที่มีขนาดใหญ่ขึ้นได้ และยังลดสัดส่วนของโปรแกรมที่ต้องประมวลผลตามลำดับได้อีกด้วย ค่าสปีดอัพที่ได้จึงต่างจากกฎของแอมดาห์ล

การใช้ไลบารี่

การใช้ไลบารี่เป็นวิธีที่ได้รับความนิยมมากสุดในปัจจุบัน เนื่องจากผู้พัฒนาโปรแกรมไม่จำเป็นต้องเรียนรู้ภาษาใหม่ แต่สามารถเขียนโปรแกรมแบบขนานได้โดยการเรียก ใช้ฟังก์ชั่นต่างๆ ที่สร้างไว้ในไลบารี่ความสัมพันธ์ของภาษาเก่า เช่น ภาษา C หรือภาษา JAVA การเขียนโปรแกรมแบบขนานนั้นต่างจากแบบตามลำดับ คือ โปรแกรมต้องมีการสร้างโพรเซสสำหรับประมวลผลมากกว่า 1 โพรเซส ซึ่งทำงานไปพร้อมกัน ในไลบารี่จึงต้องมีชุดฟังก์ชั่นสำหรับการบริหารจัดการโพรเซสต่างๆ เช่น สร้างโพรเซสใหม่ หรือยุติการทำงานของโพรเซสเก่า นอกจากนี้ยังต้องมีชุดฟังก์ชั่นสำหรับการติดต่อสื่อสารระหว่างโพรเซสอีกด้วย ตัวอย่างของไลบารี่ที่ถือว่าเป็นมาตรฐานในปัจจุบันคือ MPI (Message Passing Interface) ซึ่งใช้ในการพัฒนาโปรแกรมสำหรับสถาปัตยกรรมแบบหน่วยความจำกระจาย (Distributed Memory) นอกจากนี้ไลบารี่ OpenMP ยังได้รับความนิยมเป็นอย่างมากสำหรับใช้คู่กับสถาปัตยกรรมแบบหน่วยความจำร่วม (Shared Memory)

เครือข่ายต่อเชื่อม (Interconnection Network)

รศ.ดร.ธีรณี อจลากุล ได้อธิบายเรื่องหนังสือเทคโนโลยีการประมวลผลแบบขนานเกี่ยวได้กล่าวเกี่ยวกับเครือข่ายเชื่อมต่อดังต่อไปนี้

เครือข่ายต่อเชื่อมเป็นสื่อกลางที่ใช้ในการส่งผ่านข้อมูลระหว่างเครื่องคอมพิวเตอร์หรืออุปกรณ์ ต่างๆ อาทิ หน่วยประมวลผล หน่วยความจำ และอุปกรณ์อินพุตเอาท์พุต เครือข่ายแบ่งออกได้เป็น 4 ประเภทตามลักษณะการใช้งานและคุณสมบัติ ดังนี้

1. On-chip networks (OCNs) ใช้สำหรับเชื่อมต่ออุปกรณ์ เช่น รีจิสเตอร์ แคช และวงจรคำนวณ (ALU) ในสถาปัตยกรรมจิ๋ว (Micro-architecture) จำนวนอุปกรณ์ที่สามารถใช้เครือข่ายร่วมกันอยู่ในช่วงประมาณ 10 ชิ้น ไปจนถึง 100 ชิ้น ด้วยระยะทางระดับเซนติเมตร และความเร็วในการส่งประมาณ 2,400 Gbps OCNs จัดเป็นเครือข่ายที่มีไว้ใช้สำหรับสร้างเส้นทางข้อมูลบนไมโครชิบนั่นเอง ตัวอย่างเทคโนโลยีที่ใช้ในปัจจุบันคือ Core Connect ของบริษัท IBM

2. System/storage area networks (SANs) : ใช้สำหรับเชื่อมต่อระหว่างหน่วยประมวลผลจำนวนมาก และหน่วยประมวลผลกับหน่วยความจำ รวมทั้งอุปกรณ์อินพุตเอาท์พุตอื่นๆในเครื่องคอมพิวเตอร์ เครือข่ายลักษณะนี้มักใช้ในการสร้างเส้นทางข้อมูลภายในเครื่องซูเปอร์คอมพิวเตอร์ จำนวนอุปกรณ์ที่สามารถใช้เส้นทางร่วมกันมีตั้งแต่หลายร้อยไปจนถึงหลายพันชิ้น แล้วแต่ชนิดของเครือข่ายที่เลือกใช้ โดยทั่วไประยะทางในการเชื่อมต่อจะอยู่ในช่วง 10-100 เมตร ด้วยความเร็วในระดับประมาณ 120 Gbps ตัวอย่างเทคโนโลยีที่ใช้ในปัจจุบันคือ InfiniBand, Myrinet และเครือข่ายที่ออกแบบเฉพาะ (Proprietary) แบบต่างๆ

3. Local area networks (LANs) เป็นเครือข่ายที่รู้จักกันทั่วไปใช้สำหรับเชื่อมต่อเครื่องคอมพิวเตอร์จำนวนมาก เพื่อให้สามารถแลกเปลี่ยนข้อมูลกันได้ ตัวอย่างที่เห็นได้ชัดเจน คือ การเชื่อมต่อเครื่องในห้องปฏิบัติการคอมพิวเตอร์ผ่านระบบเครือข่าย LANs ซึ่งสามารถเชื่อมต่อคอมพิวเตอร์ได้กว่า 100 เครื่องในระยะทางประมาณ 1-10 กิโลเมตร เทคโนโลยี Ethernet เป็นตัวอย่างที่ใช้กันอย่างแพร่หลายในปัจจุบัน โดยที่ความเร็วอยู่ในช่วงประมาณ 10 Gbps

4. Wide area networks (WANs) ใช้สำหรับเชื่อมต่อเครื่องคอมพิวเตอร์ที่ตั้งอยู่ไกลกันทางกายภาพ และสามารถเชื่อมต่อเครื่องได้หลายล้านเครื่องในระยะทางหลายพันกิโลเมตร

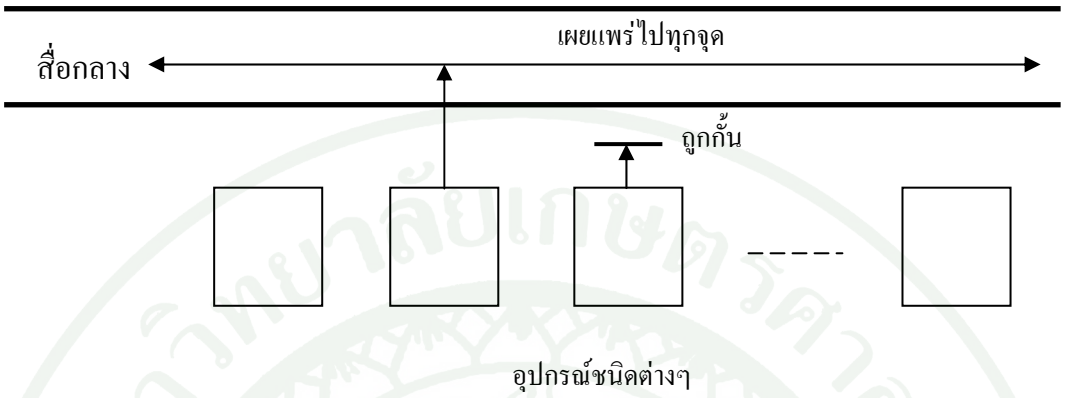
เกณฑ์ในการเลือกใช้เครือข่ายที่เหมาะสมสำหรับสถาปัตยกรรมแต่ละชนิด คือ ราคาและแบนด์วิธ (Bandwidth) ได้แก่ความสามารถสูงสุดในการส่งข้อมูลของสายส่ง ซึ่งมีหน่วยเป็นบิตหรือไบต์ต่อวินาที ปริมาณข้อมูลที่ส่งได้จริงในช่วงเวลาหนึ่ง (Effective Throughput) และความยากง่ายในการพัฒนา

ข้อมูลที่ถูกส่งผ่านเครือข่ายจะอยู่ในรูปของแพ็กเกต (Packet) ซึ่งมีขนาดคงที่ หากความยาวของข้อมูลที่ต้องการส่งมากกว่าขนาดของแพ็กเกต ข้อมูลจะถูกแบ่งย่อยและบรรจุลงในหลายๆ แพ็กเกต นอกจากนี้ในบางกรณีที่สายส่งมีแบนด์วิธต่ำ แพ็กเกตอาจถูกแบ่งออกเป็นชิ้นย่อยได้อีกเพื่อให้อุปกรณ์สวิตช์สามารถทำงานได้สะดวกขึ้น

สื่อกลางที่ใช้ในการสร้างเครือข่ายเชื่อมต่อแบ่งออกได้เป็น 2 ประเภทหลัก คือ

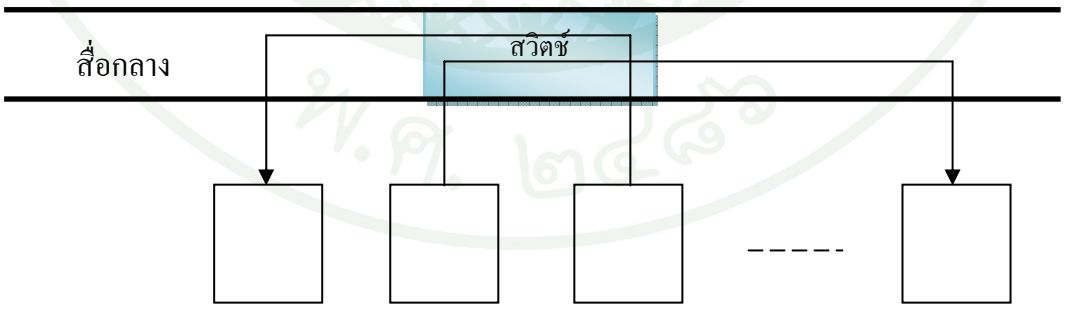
1. สื่อกลางแบบใช้ร่วมกัน (Shared Medium) การส่งผ่านข้อมูลระหว่างอุปกรณ์คู่ ใด ๆ จะทำได้โดยการเผยแพร่หรือกระจายข้อมูล (Broadcast) ไปบนสื่อกลาง อุปกรณ์ที่ต่อเชื่อมกับสื่อกลางทุกตัวจึงมองเห็นข้อมูล อย่างไรก็ดี อุปกรณ์ที่มีตำแหน่งตรงกับที่ระบุในแพ็กเกตเท่านั้นที่จะทำการรับข้อมูลการส่งข้อมูลอย่างถูกต้องจึงทำได้ทีละ 1 ข้อความเท่านั้น ไม่สามารถส่งพร้อมกัน

ได้ หากเกิดการชนกันของข้อความจะต้องมีการส่งใหม่ ตัวอย่างของสื่อประเภทนี้คือ บัส (Bus) และอีเทอร์เน็ท (Ethernet) รูปที่ 3 แสดงการส่งผ่านข้อมูลในสื่อกลางแบบดังกล่าว



ภาพที่ 2 สื่อกลางแบบใช้ร่วมกัน

2. สื่อกลางแบบสวิตช์ (Switched Medium) : สามารถรองรับการส่งผ่านข้อมูลแบบจุดต่อจุด (Point-to-point) ระหว่างอุปกรณ์หลายๆคู่พร้อมกันได้ โดยอุปกรณ์แต่ละชิ้นจะมีเส้นทางเฉพาะไปยังสวิตช์ ซึ่งจะทำหน้าที่เลือกเส้นทางบนสื่อกลางให้กับข้อความแต่ละข้อความที่ถูกส่ง การใช้สื่อกลางลักษณะนี้จะอำนวยความสะดวกในการขยายเครือข่ายเชื่อมต่อในกรณีที่มีการเพิ่มจำนวนอุปกรณ์ในระบบ รูปที่ 4 แสดงการส่งผ่านข้อมูลพร้อมกันระหว่างอุปกรณ์ 2 คู่ จากรูปสวิตช์จะทำหน้าที่แยกเส้นทางเดินของข้อความทั้งสอง ทำให้ไม่เกิดการชนกันบนสื่อกลาง



ภาพที่ 3 สื่อกลางแบบสวิตช์

การสื่อสารผ่านเครือข่ายอินเทอร์เน็ตแบบ TCP / IP

เอกสิทธิ์ วิริยจารีได้อธิบายเรื่อง TCP / IP ในหนังสือเรียนรู้ระบบเน็ตเวิร์กจากอุปกรณ์ของ Cisco ภาคปฏิบัติ ดังต่อไปนี้

TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต ได้รับการพัฒนามาตั้งแต่ปี 1960 โดยกระทรวงกลาโหมของสหรัฐอเมริกา หรือ DOD (Department of Defense) ซึ่งถูกใช้เป็นครั้งแรกในเครือข่ายARPANET ต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

TCP/IP เป็นโปรแกรม 2 เลเยอร์ TCP (Transmission Control Protocol) เป็นเลเยอร์ที่สูงกว่าทำหน้าที่จัดการแยกข้อความหรือไฟล์และประกอบให้เหมือนเดิม IP (Internet Protocol) เป็นเลเยอร์ที่ต่ำกว่า ทำหน้าที่จัดการส่วนของที่อยู่ของแต่ละชุดข้อมูล เพื่อทำให้มีปลายทางที่ถูกต้อง

TCP/IP มีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ ในการส่งข้อมูลจะมีการแบ่งข้อมูลที่ต้องการส่งออกเป็นส่วนย่อยๆที่เรียกว่า แพ็คเก็จ (package) จากนั้นแต่ละแพ็คเก็ตจะถูกส่งไปปลายทางด้วยเส้นทางต่างๆที่เชื่อมโยงกันในระบบตามเส้นทางที่สามารถส่งถึงปลายทางได้ โดยแต่ละแพ็คเก็ตไม่จำเป็นต้องเรียงลำดับหรือไปตามเส้นทางเดียวกัน ซึ่งในระบบจะมีอุปกรณ์ที่เรียกว่า เร้าเตอร์ (Router) จะเป็นตัวคอยจัดหาเส้นทางที่ดีที่สุดให้กับทุกแพ็คเก็ต เมื่อถึงจุดหมายระบบปลายทางจะรวบรวมแพ็กเก็ตกลับให้เป็นข้อมูลเดียว แต่ถ้าแพ็กเก็ตใดขาดหายหรือตกหล่น คอมพิวเตอร์ก็จะตรวจสอบ และส่งแพ็กเกตมาใหม่ จนข้อมูลครบเหมือนเดิม

งานวิจัยที่เกี่ยวข้อง

Pawlikowski *et al*. (1994) ศึกษาการกระจายการจำลองสถานการณ์และการหาคำตอบร่วมกันแบบขนาน (MRIP) ซึ่งสามารถช่วยแก้ปัญหาในด้านประสิทธิภาพการทำงาน โดยมีการใช้โปรแกรม AKAROA ช่วยในการจำลองสถานการณ์ โดยทำการเปรียบเทียบผลการทำงานของ SRIP และ MRIP ผลที่ได้คือความเร็วจะเพิ่มมากขึ้นเมื่อมีการทำงานที่ตัวประมวลผลหลายตัวหรือกระจายการทำงานไปที่เครื่องคอมพิวเตอร์หลายตัว นอกจากนี้ความเร็วในการประมวลผลยังขึ้นอยู่กับ การเชื่อมต่อและจำนวนตัวประมวลผล ในบางสถานการณ์การมีตัวประมวลผลมากไม่ได้ช่วยให้ความเร็วเพิ่มขึ้น

Bruschi *et al*. (2004) ได้ทำการพัฒนากระจายสภาวะแวดล้อมของการจำลองสถานการณ์แบบอัตโนมัติ (Ambiente de Simulação Distribuída Automático :ASDA) โดยมีเป้าหมายให้สามารถทำการกระจายการจำลองสถานการณ์ได้หลายรูปแบบอย่างอัตโนมัติ ทั้งแบบ SRIP และ MRIP ASDA ช่วยผู้ใช้ในการพัฒนาการจำลองสถานการณ์แบบกระจายได้ง่ายและรวดเร็ว สามรถใช้งานได้ง่ายโดยไม่จำกัดเฉพาะผู้ใช้ระดับสูงเท่านั้น จุดเด่นของ ASDA คือ ใช้การกระจายที่หลากหลาย ช่วยให้ผู้ใช้สามารถตัดสินใจได้ดีขึ้นจากผลลัพธ์ที่ได้ ใช้ภาษาที่แตกต่างในการจำลองสถานการณ์ และจัดลำดับโปรแกรมจำลองสถานการณ์โดยการใช้ MRIP

Mota *et al*. (2000) ได้ทำการศึกษาการลดเวลาในการจำลองสถานการณ์ที่มีการทำซ้ำแบบขนานบนตัวประมวลผลหลายตัวที่เชื่อมต่อกันผ่านเครือข่ายและการหาผลเฉลี่ยที่เหมาะสม โดยใช้วิธีการที่เรียกว่า Batch Means ซึ่งเป็นวิธีในการหาความเหมาะสมของการเปลี่ยนแปลงของลำดับผลลัพธ์ตั้งต้น และมีการใช้โปรแกรม AKAROA-2 ช่วยในการจำลองสถานการณ์ และใช้เครือข่ายคอมพิวเตอร์ในการคำนวณจำนวนรอบในการรันแบบขนาน จากการศึกษาได้ทำการทดลองกับแบบจำลองของปัญหาการติดต่อสื่อสาร ผลจาการทดลองได้เป็นที่น่าพอใจทั้งด้านความเร็วที่มากขึ้นและคุณภาพของผลลัพธ์ที่ได้

Yau (1999) ศึกษาการลดเวลาในการจำลองสถานการณ์ที่ต้องใช้ระยะเวลาในการประมวลผลนาน มีการใช้โปรแกรม AKAROA ในการจำลองสถานการณ์ มีการใช้วิธีการใหม่ในการวิเคราะห์ลำดับสำหรับ MIRP รวมถึงการเปลี่ยนการประมวลผลแบบไม่ขนานเป็นการประมวลผลแบบขนานที่เหมาะสม และการหยุดการทำงานอัตโนมัติเมื่อความต้องการบรรลุผล มีการใช้ตัวประมวลผลหลายตัว จากการศึกษาพบว่า SA-PTS (Spectral Analysis in Parallel TimeStreams) เป็นตัวเริ่มต้นที่ดีสำหรับ AKAROA โดยเป็นการการกำหนดค่าเริ่มต้นที่เหมาะสมให้กับAKAROA ในการทดลองและมีการใช้จำนวนเครื่องในการประมวลผลน้อยกว่า 10 เครื่อง

Alleon *et al*. (2006) ศึกษาเกี่ยวกับ การกระจายจำนวนการจำลองสถานการณ์แบบขนานในด้านการบิน สองสิ่งที่สำคัญมากที่เกี่ยวข้องกับการใช้ทรัพยากรของคอมพิวเตอร์คือการเคลื่อนที่ของอากาศและปรากฏการณ์ของเสียง ซอฟแวร์ถูกใช้ในด้านการวิเคราะห์ปัญหาที่ใหญ่และซับซ้อนในเวลาที่จำกัด จึงใช้การประมวลผลแบบขนานเข้ามาช่วยให้การทำงานมีประสิทธิภาพมากยิ่งขึ้น จากการทดลองจะเห็นได้ว่า เมื่อมีการเพิ่มตัวประมวลผลจะทำให้เวลาที่ใช้ในการประมวลผลลดลง โดยในการทดลองใช้จำนวนตัวประมวลผลสูงสุด 32 เครื่อง

Thole and Stüben (1999) ศึกษาเกี่ยวกับสถาปัตยกรรมของการประมวลผลแบบขนาน โดยเริ่มมีการสนใจตั้งแต่ปี 1984 และมีการพัฒนาการประมวลผลแบบขนานเรื่อยมา ในด้านของโรงงานอุตสาหกรรมเริ่มมีการใช้การประมวลผลแบบขนานในปี 1993 และมีการใช้กันอย่างแพร่หลายในปี 1999 การศึกษานี้ให้ความสนใจในการนำการประมวลผลแบบขนานไปใช้กับงาน 10 ประเภทคือ การเคลื่อนไหวที่เกี่ยวกับการออกแบบรถยนต์ การจำลองเกี่ยวกับความปลอดภัยของรถยนต์ การผลิตภาพการ์ตูนเคลื่อนไหว การออกแบบยา เครื่องหลอมโลหะกระบวนการผลิตโพลีเมอร์ กระบวนการของภาพดาวเทียม การวิเคราะห์ความปลอดภัย การออกแบบเครื่องจักรเทอร์โบ และการทดสอบเกี่ยวกับแม่เหล็กไฟฟ้าของพาหนะ การประมวลผลแบบขนานมีประโยชน์หลายทาง เช่น ช่วยลดเวลาในการออกแบบ ผลลัพธ์มีความน่าเชื่อถือ เพิ่มความสามารถในการวิเคราะห์ระบบที่ซับซ้อน

Bononi *et al*. (2005) ศึกษาทางเลือกใหม่ในการลดเวลาในการประมวลผลการจำลองสถานการณ์และการใช้ทรัพยากรในการสื่อสาร โดยได้เสนอการจำลองสถานการณ์แบบขนานและการกระจาย ซึ่งสนับสนุนการทำงานร่วมกันทำซ้ำแบบขนานและการกระจายการจำลองสถานการณ์ (Concurrent Replication of Parallel and Distributed Simulations :CR-PADS) ซึ่งเป็นทางเลือกใหม่ในการประมวลผลแบบเชิงเส้นของจำลองสถานการณ์แบบขนานและการกระจาย การทำงานของ CR-PADS จะเป็นการเริ่มที่เวลาเดียวกัน ซึ่งสอดคล้องกับการจำลองสถานการณ์แบบขนานและการกระจาย โดยแต่ละรอบของการทำงานต้องใช้แบบจำลองเดียวกัน มีค่าตัวแปรเริ่มต้นเหมือนกันและมีตัวเลขสุ่มที่ต่างกัน ผลที่ได้จากการทดสอบแสดงให้เห็นว่าสามารถลดเวลาและมีการใช้ทรัพยากรได้อย่างคุ้มค่า และมีประสิทธิภาพใกล้เคียงกับการจำลองสถานการณ์แบบขนานและการกระจาย

Fujimoto (2001) การจำลองสถานการณ์แบบขนานและการกระจายเริ่มมีการศึกษาในทศวรรษที่ 70 ถึง 80 และได้มีความสนใจเพิ่มมากขึ้นในระยะเวลาเพียง 10 ปี และในทุกวันนี้ได้นำไปประยุกต์ใช้ในหลายสาขา เช่น การฝึกทหาร การวิเคราะห์เครือข่ายการติดต่อสื่อสาร การควบคุมระบบคมนาคมทางอากาศ ในส่วนของการสอนแนะนำเทคนิคในการกระจายการดำเนินการของการจำลองสถานการณ์บนคอมพิวเตอร์หลายเครื่อง ส่วนสำคัญคือการทำงานพร้อมกัน ที่เรียกว่าการจัดการเวลา และการเทคนิคการกระจายข้อมูล

Kiesling (2006) งานวิจัยนี้ได้เสนอการผสมผสานระหว่างเทคนิคใหม่ในการจำลองสถานการณ์แบบขนานโดยใช้เวลาต่อเนื่อง ทำให้ผลลัพธ์ที่ไม่แน่นอนถูกคำนวณอย่างรวดเร็ว และสามารถปรับปรุงได้ภายหลัง ผู้ใช้สามารถยกเลิกการทำงาน ณ เวลาใดๆ เมื่อได้ผลเป็นที่พอใจเทคนิคในการจำลองสถานการณ์แบบขนานโดยใช้เวลาต่อเนื่องนั้น ไม่สามารถนำไปใช้ได้กับทุกแบบจำลอง แต่

มีความเป็นไปได้ในการนำไปใช้กับแบบจำลองที่เกี่ยวข้องกับการสนับสนุนการจัดตารางการทำงาน และควบคุมระบบในโรงงานอุตสาหกรรม ตัวอย่างในการประยุกต์ใช้เทคนิคนี้คือ การจำลองระบบแถวคอย

ปิยะวดี (2551) ได้ทำการทดลองพัฒนาระบบการประมวลผลแบบกระจายโดยใช้ Visual Basic 6 และ Visual Basic for Application เป็นภาษาในการพัฒนาระบบ พร้อมทั้งพิสูจน์ในเชิงสถิติว่าการกระจายจำนวนรอบประมวลผลสำหรับโปรแกรม Arena สามารถใช้การเปลี่ยน Seed Number เป็นตัวควบคุมการจำลองสถานการณ์ให้ผลไม่ต่างจากการประมวลผลแบบเครื่องเดี่ยวในเชิงสถิติ และพิสูจน์ว่าการประมวลผลแบบขนานสามารถที่จะลดเวลาในการประมวลผลได้จริง

อุปกรณ์และวิธีการ

อุปกรณ์

1. เครื่องคอมพิวเตอร์ Processor: Intel® Core™ i3 Ram: 1 GB มากกว่า 3 เครื่องขึ้นไป

2. โปรแกรมภาษา Visual Basic Studio Version 2008 Enterprise Edition สำหรับใช้ในการพัฒนาแอพพลิเคชั่นและไลบารี่ในการติดต่อสื่อสาร

3. Switch สำหรับการเชื่อมต่อเครื่องคอมพิวเตอร์สำหรับสร้างเครือข่าย

4. โปรแกรมสำเร็จรูป Arena Version 11 ขึ้นไปสำหรับใช้ในการทดลอง

5. โปรแกรม Microsoft Access 2003 ขึ้นไปเพื่อใช้สำหรับเปิดข้อมูลที่ประมวลผลแล้วของโปรแกรมสำเร็จรูป Arena

วิธีการ

1. ศึกษาสถาปัตยกรรมเครือข่าย

2. ศึกษาการเชื่อมต่อคอมพิวเตอร์แบบขนาน

3. ศึกษาการใช้งาน Visual Basic Studio Version 2008 Enterprise

4. สร้างแอพพลิเคชั่นสำหรับการเชื่อมต่อคอมพิวเตอร์ให้ทำการประมวลผลแบบขนาน

5. สร้างไลบารี่สำหรับการเชื่อมต่อคอมพิวเตอร์ให้ทำการประมวลผลแบบขนาน

6. ออกแบบการทดลอง

ศึกษาสถาปัตยกรรมเครือข่าย

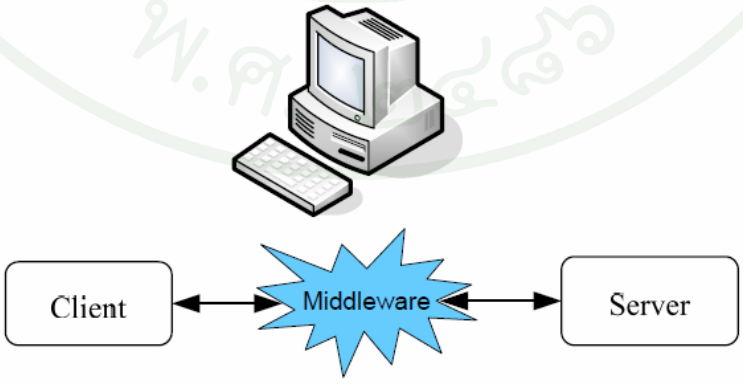
สถาปัตยกรรมซอฟต์แวร์ Client/Server

ระบบ Client/Server ได้ถูกพัฒนาขึ้นมาเพื่อตอบสนองแนวคิดการ Downsizing เป็นการลดค่าใช้จ่ายระบบ Time Sharing ของเครื่อง Mainframe ซึ่งระบบ Client/Server เป็นระบบประมวลแบบกระจาย (Distributed Processing) โดยจะแบ่งการประมวลผลระหว่าง Client และ Server โปรแกรมประยุกต์ (Application Program) จะประมวลผลบางส่วนที่ Client และบางส่วนก็ประมวลที่ Server โดยเป็นระบบที่เครื่องคอมพิวเตอร์เครื่องหนึ่ง ต่อเข้ากับคอมพิวเตอร์อีกเครื่องหนึ่งเป็นอย่างน้อย ซึ่งเครื่องที่เชื่อมต่อด้วยนี้จะมีขนาดใหญ่ มีโปรเซสเซอร์ตั้งแต่หนึ่งตัวขึ้น

ระบบ Client/Server จะมีความยืดหยุ่นสูง เพราะนอกเหนือจากการเชื่อมต่อเข้าด้วยกันตามปกติแล้ว ยังสามารถเลือกที่จะเชื่อมต่อทั้งระบบเข้ากับเครื่องในระดับ minicomputer หรือ mainframe ได้อีกด้วย โดยเครื่องที่แต่ละเครื่องที่ใช้อยู่ยังคงสามารถใช้งานในสภาวะแวดล้อมและโปรแกรมที่เราคุ้นเคยได้ดี ในขณะที่ผู้ใช้งานสามารถเลือกทำงานได้ทั้งงานในรูปแบบเครื่องเดี่ยว(stand alone) หรือแบบที่ประสานงานกับผู้ใช้รายอื่น รวมไปถึงการทำงานโดยอาศัยข้อมูลจำนวนเก็บอยู่ในเครื่อง mainframe อีกด้วย

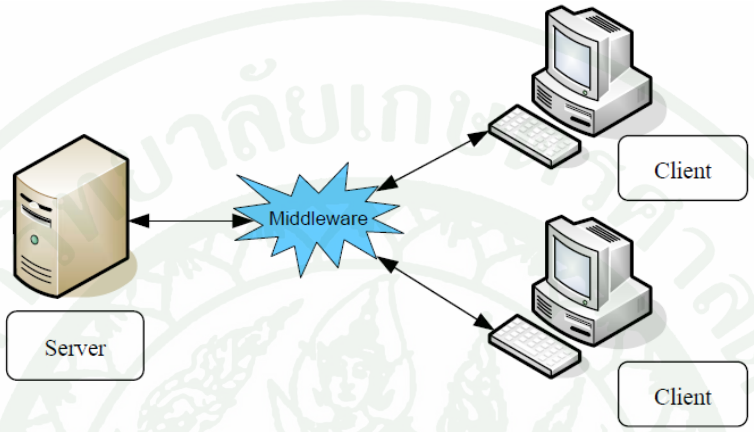
รูปแบบของ Client/Server แบ่งออกเป็น 4 ชนิด ดังนี้

1. Stand alone Client/Server ผู้ให้บริการจะอยู่บนเครื่องเดียวกับผู้ใช้บริการหรือ Client ความเร็วในการติดต่อสื่อสารสูง เรียกอีกชื่อหนึ่งว่า Tiny Client/Server



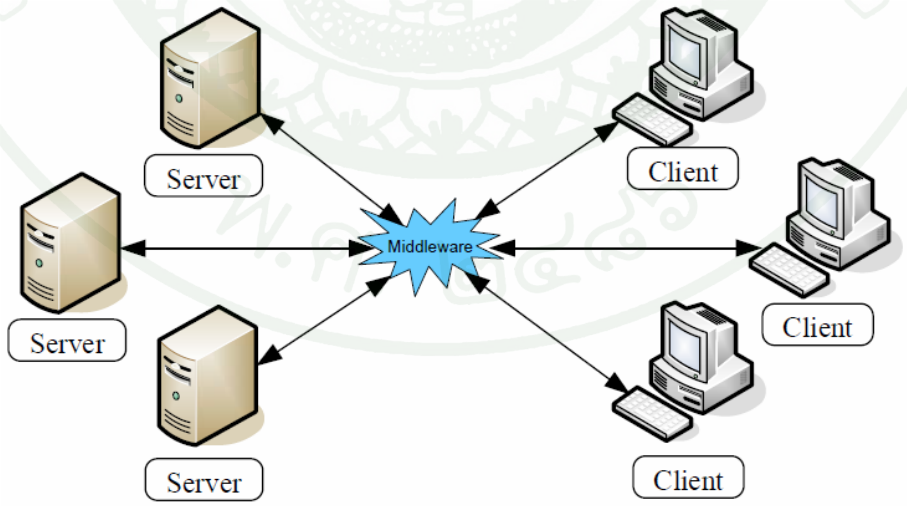
ภาพที่ 4 Stand alone Client/Server

2. Department Client/Server หรือ LAN based single Server เป็นการทำงานที่ผู้ให้บริการจะบริการฐานข้อมูลและโปรแกรมประยุกต์ต่างๆ ที่อยู่บน Server โดยจะเชื่อมต่อกันด้วยระบบเครือข่ายท้องถิ่น (LAN) และ Middleware ประสิทธิภาพการประมวลผลจะช้ากว่าStand alone เพราะต้องผ่านเครือข่าย



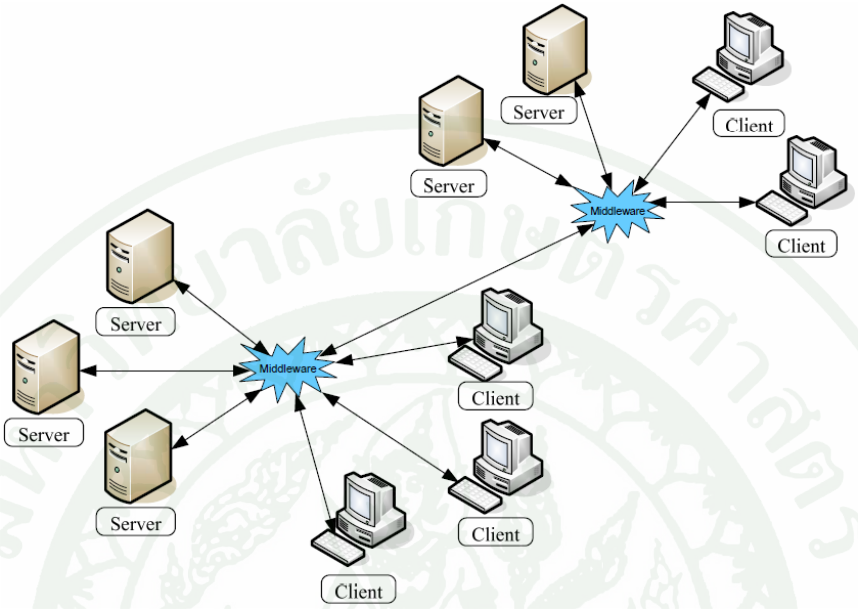
ภาพที่ 5 Department Client/Server หรือ LAN based single Server

3. Workgroups Client/Server กลุ่มของเซิร์ฟเวอร์หลายแพลตฟอร์มหลายผู้ผลิต เชื่อมต่อกันทางระบบเครือข่าย LAN หรือ WAN สามารถใช้ทรัพยากรร่วมกันได้ โดยผ่านMiddleware



ภาพที่ 6 Workgroups Client/Server

4. Enterprise Client/Server หลาย Workgroup Client/Server เชื่อมต่อกัน ทำให้มีการใช้ทรัพยากรได้อย่างมีประสิทธิภาพ



ภาพ 7 Enterprise Client/Server

ศึกษาการเชื่อมต่อคอมพิวเตอร์แบบขนาน

การประมวลผลแบบขนานหรือแบบกระจาย เป็นลักษณะของหน่วยประมวลผลที่มีมากกว่า 1 เครื่อง ซึ่งเป็นการแบ่งงานใหญ่ๆออกเป็นงานย่อยๆจำนวนมากแล้วให้เครื่องหลายเครื่องช่วยกันทำงานและสามารถทำงานย่อยๆนั้นได้พร้อมกัน ดังนั้นผู้ใช้จึงรู้สึกได้ทันทีว่าระบบเร็วขึ้นเนื่องจากแต่ละเครื่องรับงานน้อยลง

การเชื่อมต่อคอมพิวเตอร์แบบขนานตามความเข้าใจทั่วไปจะมีด้วยกัน 2 แบบคือ

1. เครื่องคอมพิวเตอร์หลายเครื่องเชื่อมต่อกันให้เสมือนเครื่องคอมพิวเตอร์เมนเฟรมเครื่องเดียว โดยการประมวลผลจะเหมือนกับมีเครื่องคอมพิวเตอร์เครื่องเดียว

2. เครื่องคอมพิวเตอร์หลายเครื่องช่วยกันประมวลผล โดยแต่ละเครื่องเป็นอิสระต่อกัน โดยงานวิจัยนี้ได้ใช้การเชื่อมต่อแบบขนานประมวลผลแบบแต่ละเครื่องเป็นอิสระต่อกัน

ศึกษาการใช้งาน Visual Basic Studio Version 2008 Enterprise

Visual Basic 2008 หากนับตามรุ่นก็ถือว่าเป็น Visual Basic เวอร์ชัน 9 แล้ว มันเป็นทั้งภาษาโปรแกรม เป็นทั้งเครื่องมือที่นักพัฒนาซอฟต์แวร์ได้นำมาพัฒนาซอฟต์แวร์ ที่ทำงานได้หลากหลาย ไม่ว่าจะทำงานบนคอมพิวเตอร์, อินเตอร์เน็ต, โทรศัพท์มือถือ ทำให้ Visual Basic เป็นภาษาที่ได้รับความสนใจจากนักเขียนโปรแกรมทั่วโลกสูงสุด

การเขียนโปรแกรมด้วย Visual Basic 2008 เป็นการเขียนโปรแกรมแบบ Event Driven Programming ซึ่งก็คือ การเขียนคำสั่งกำหนดให้โปรแกรมทำงานในสิ่งที่เราต้องการตามเหตุการณ์ (Event) ที่เกิดขึ้น

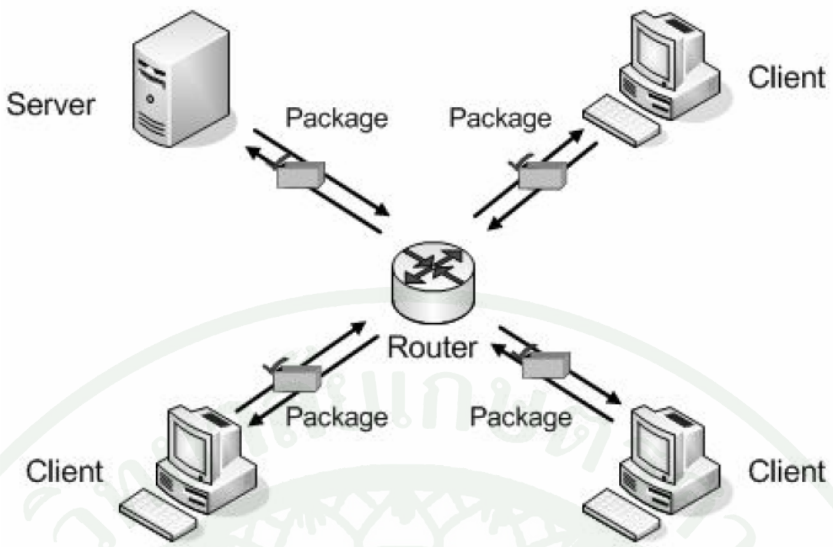
การเขียนโปรแกรมแบบ Event Driven สอดคล้องกับการทำงานของ Window ซึ่งเราจะเห็นว่ามีส่วนที่ติดต่อผู้ใช้เป็นปุ่ม, เมนู, ตัวเลือก (หรือออบเจ็กต์ขชนิดต่างๆ) โดย Event จะเกิดขึ้นได้หลายทาง เช่น ผู้ใช้งานคลิก, กดปุ่มบนคีย์บอร์ด รวมทั้งตัวเครื่องเองได้รับสัญญาณจากภายนอกมาก็ทำให้ Event เกิดขึ้นมาได้ ดังนั้น เราจะมาเขียนโปรแกรมเมื่อ Event ดังกล่าวเหล่านั้นเกิดขึ้น

ยกตัวอย่างเช่น เราเป็นโปรแกรมเมอร์ที่สร้างโปรแกรม Microsoft Word เราคงต้องเขียนโปรแกรมสักอย่างเมื่อผู้ใช้งานคลิกปุ่ม Save บนหน้าต่างโปรแกรม

แม้แอพพลิเคชันหนึ่งๆมี Event ที่เกิดขึ้นมากมายทั้งจากผู้ใช้งาน จากฮาร์ดแวร์ จากระบบปฏิบัติการ แต่เราก็ไม่จำเป็นต้องเขียนโปรแกรมเพื่อรองรับกับทุก Event เราสามารถเลือกเฉพาะ Event ที่สนใจ หรือมีผลโดยตรงกับการทำงานได้

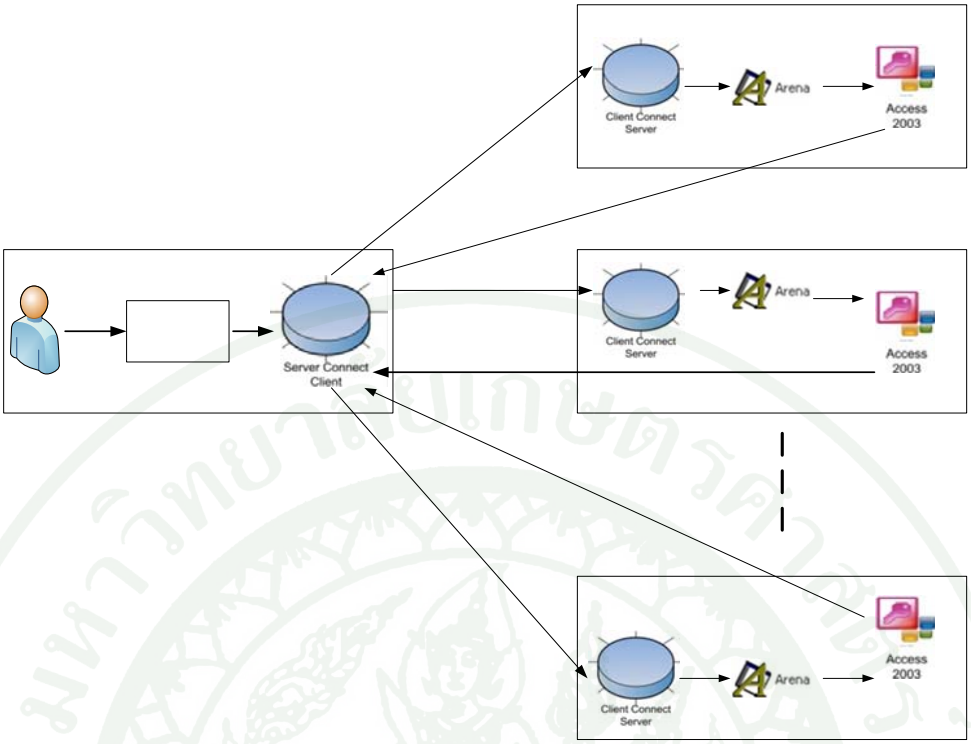
สร้างแอพพลิเคชั่นสำหรับการเชื่อมต่อคอมพิวเตอร์แบบขนาน

งานวิจัยนี้ได้เลือกใช้การติดต่อสื่อสารแบบ Department Client/Server โดยทำการติดต่อแบบ TCP/IP ให้ผู้ใช้ทำงานเปรียบเสมือน Server ที่ทำการรับส่งข้อมูลจาก Client



ภาพที่ 8 การสื่อสารแบบ Client/Server

โดยการพัฒนาแอพพลิเคชั่นที่ทำหน้าที่ประสานงานแบบขนานกับแบบจำลองสถานการณ์ ซึ่งทำหน้าที่สร้าง Script สำหรับสั่งให้เครื่อง Client ทำงานพร้อมทั้งเชื่อมโยงส่งและรับข้อมูลที่จำเป็นกับคอมพิวเตอร์ที่มีโปรแกรมแบบจำลองสถานการณ์อยู่ผ่านเครือข่ายอินเตอร์เน็ต โดยในแอพพลิเคชั่นที่พัฒนาขึ้นในเบื้องต้นนี้จะพัฒนาสำหรับใช้กับโปรแกรม Arena Version 10 เป็นต้น ซึ่งแอพพลิเคชั่นนี้จะถูกติดตั้งทั้งเครื่อง Server และเครื่อง Client



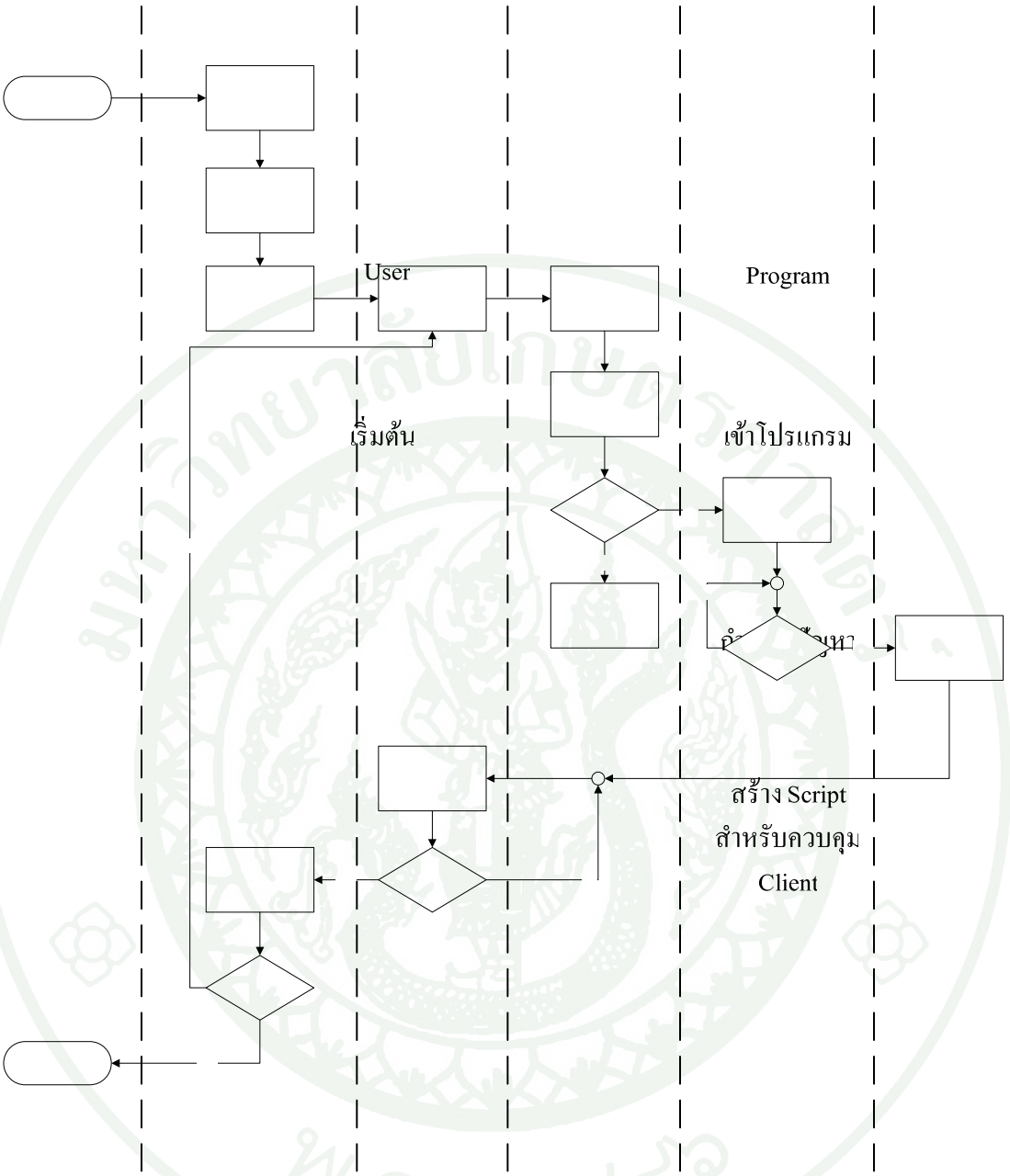
ภาพที่ 9 โครงสร้างของแอพพลิเคชั่นที่ทำหน้าที่ประสานงานแบบขนานกับแบบจำลองสถานการณ์

Program

โครงสร้างของแอพพลิเคชั่นที่ทำหน้าที่ประสานงานแบบขนานกับแบบจำลองสถานการณ์ดังแสดงในภาพ อธิบายได้ดังนี้

1. Server/User สร้าง Script ผ่านแอพพลิเคชั่นสำเร็จรูปเพื่อควบคุมการประมวลผลแบบจำลองสถานการณ์โปรแกรม Arena พร้อมทั้งทำการเชื่อมต่อไปยังเครื่อง Client จำนวน n เครื่อง

2. ส่งแบบจำลองสถานการณ์และสั่งให้แต่ละเครื่องประมวลผลแบบจำลองสถานการณ์ที่ส่งไปให้เครื่อง Client แต่ละเครื่อง ประมวลผลด้วยโปรแกรม Arena เสร็จเรียบร้อยจะทำการส่งผลการประมวลผลด้วยกลับมาในรูปแบบของไฟล์ Microsoft Office Access

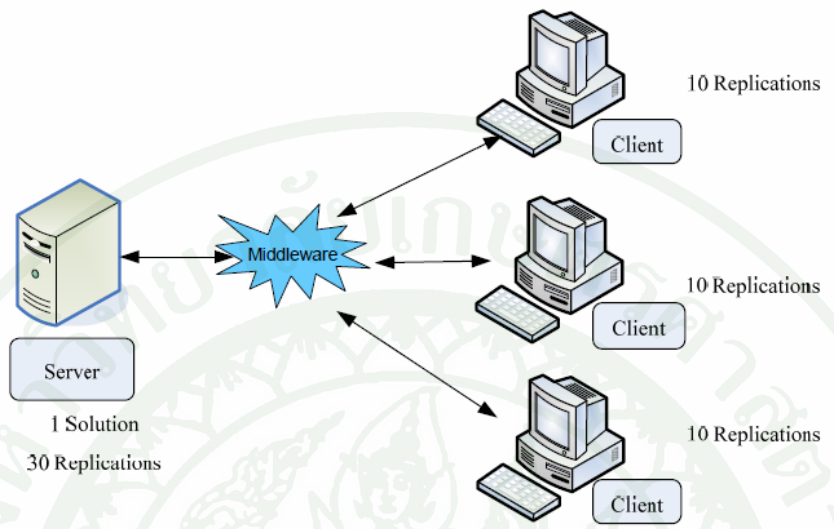


ภาพที่ 10 Swim lane diagram การทำงานของแอพพลิเคชั่น

No

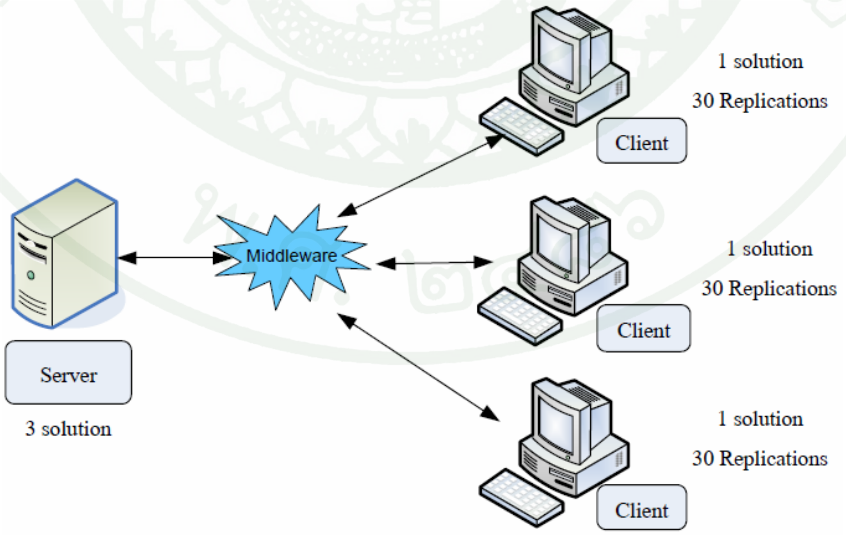
แอพพลิเคชั่นสำหรับประมวลผลแบบจำลองสถานการณ์แบบขนานเป็นแอพพลิเคชั่นที่กำหนดรูปแบบการส่งไว้ล่วงหน้าโดย Server จะส่ง Script ที่กำหนดไว้ล่วงหน้าไปยัง Client ที่ได้กำหนดเอาไว้ จึงสามารถที่จะประยุกต์ใช้ได้ 2 รูปแบบคือ

1. ประมวลผลทีละชุดพารามิเตอร์โดยการแบ่งรอบรันโปรแกรม Arena พร้อมทั้งกำหนด Seed Number ให้ต่างกันแล้วนำผลมารวมกัน



ภาพที่ 11 การทำงานแบบประมวลผลทีละชุดพารามิเตอร์

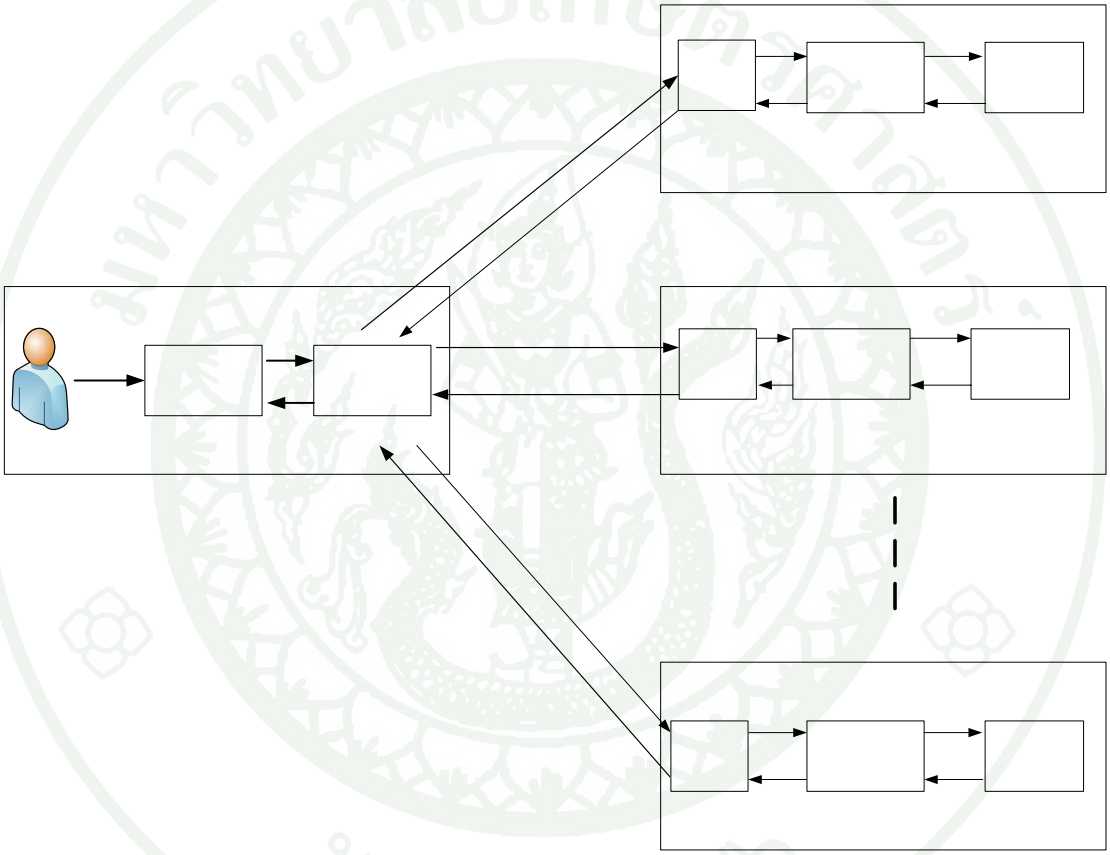
2. ประมวลผลทีละหลายชุดพารามิเตอร์โดยให้ Client แต่ละเครื่องรันชุดพารามิเตอร์ที่เราได้กำหนดไว้ล่วงหน้า



ภาพที่ 12 การทำงานแบบทีละหลายชุดพารามิเตอร์

สร้างไลบารี่สำหรับการเชื่อมต่อคอมพิวเตอร์ให้ทำการประมวลผลแบบขนาน

ในการสร้างไลบารี่สำหรับการเชื่อมต่อคอมพิวเตอร์แบบขนานเป็นการสร้างฟังชั่นสำหรับการเขียนโปรแกรมด้วย Visual Basic Studio Version 2008 Enterprise Edition เพื่อให้สามารถนำไปใช้งานในการเขียนโปรแกรมภาษา VB.Net เพื่อประยุกต์ใช้เชิง Logic ซึ่งต้องอาศัยตอบสนองไปกลับระหว่าง Client และ Server โดยทำการติดต่อแบบ TCP/IP ได้



ภาพที่ 13 โครงสร้างของการประยุกต์ใช้ไลบารี่ที่ทำหน้าที่ประสานงานแบบขนาน

โครงสร้างของการประยุกต์ใช้ไลบารี่ที่ทำหน้าที่ประสานงานแบบขนาน ดังแสดงในภาพที่ อธิบายได้ดังนี้

1. Server/User สร้างโปรแกรมสำหรับการควบคุมการสร้าง Script สำหรับการเปลี่ยนพารามิเตอร์เพื่อรันโปรแกรมที่เครื่อง Client และเช็ค Logic พร้อมทั้งดึงไลบารี่มาใช้งานเพื่อทำการเชื่อมต่อไปยังเครื่อง Client จำนวน n เครื่อง

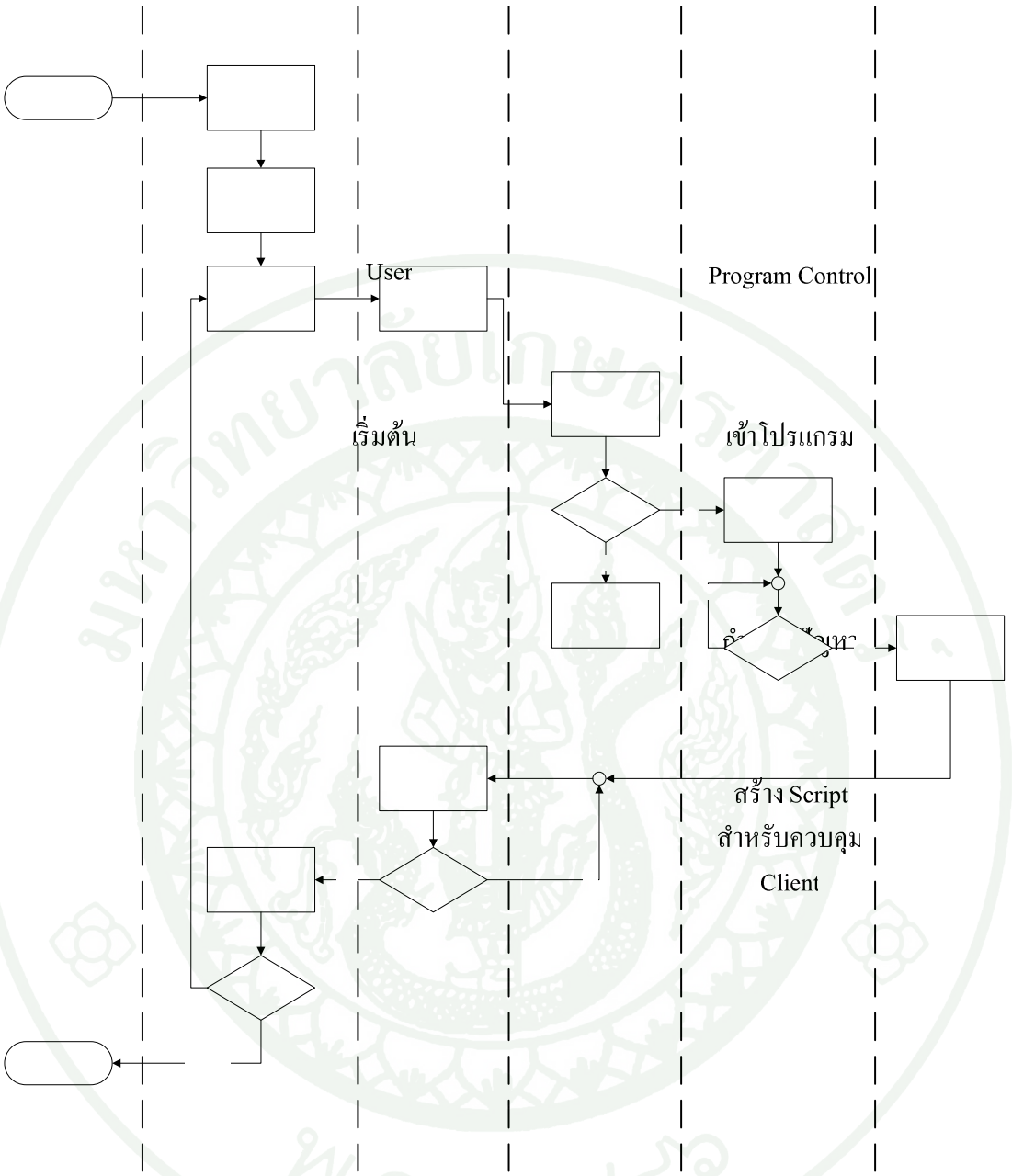
Program Control Library

2. ส่ง Script ไปยังเครื่อง Client โดยแต่ละเครื่อง Client ต้องมีการเขียนโปรแกรมเพื่อรองรับการทำงานของ Script ไว้ล่วงหน้า

3. เครื่อง Client จะทำงานตามคำสั่ง Script โดยการเปลี่ยนแปลงค่าพารามิเตอร์ของโปรแกรมตามที่ Script ได้เขียนเอาไว้พร้อมทั้งรันโปรแกรม

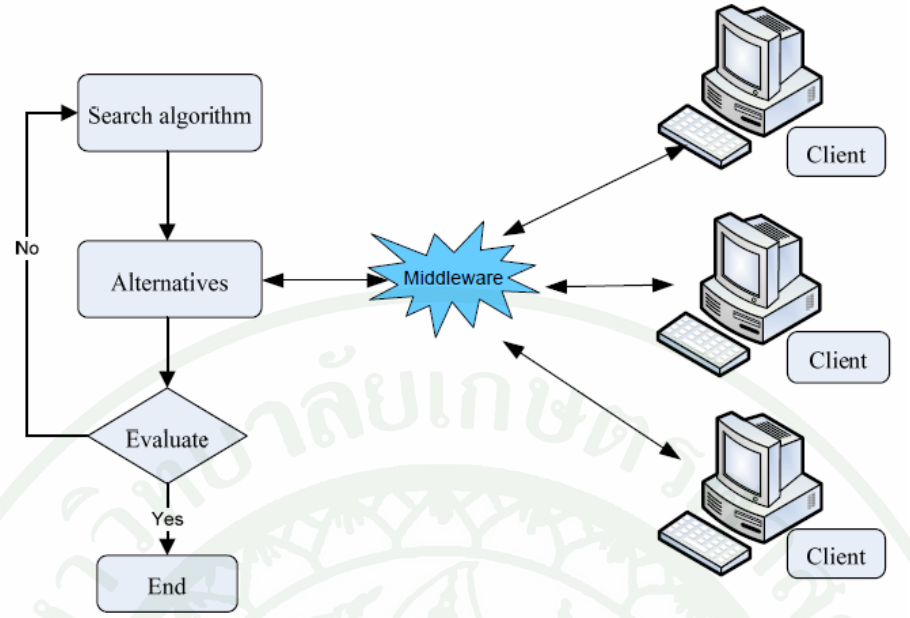
4. เครื่อง Client จะนำ Output ที่ได้ส่งกลับไปเครื่อง Server โดยอาศัยการดึงไลบารี่มาใช้งาน

5. Server จะนำค่า Output มาตรวจสอบพร้อมทั้งเขียน Script เพื่อเปลี่ยนพารามิเตอร์ตาม Logic สำหรับเครื่อง Client แล้วส่งกลับไปยังเครื่อง Client จนกว่าจะเข้าเงื่อนไขของการหยุดทำงาน



ภาพที่ 14 Swim lane diagram การประยุกต์ใช้ไลบรารี

ไลบารี่สำหรับการเชื่อมต่อเป็นเครื่องมือที่สร้างขึ้นมาสำหรับประยุกต์ใช้ในการประมวลผลแบบขนานสำหรับการจำลองสถานการณ์ หรือสำหรับโปรแกรม Search Algorithms อื่นๆ ซึ่งนำมาใช้สร้างโปรแกรมสำหรับควบคุมการประมวลผลแบบขนาน โดยสามารถทำให้มีการตอบสนองไป-กลับระหว่าง Server กับ Client โดยโปรแกรมควบคุมจะทำการเปลี่ยนแปลงค่าพารามิเตอร์ต่าง ๆ ได้โดยอัตโนมัติตาม Logic กำหนดไว้ โดยไม่ต้องกำหนดรูปแบบการส่งไว้ล่วงหน้า



ภาพที่ 15 การทำงานแบบ Logic

การออกแบบการทดลอง

การออกแบบการทดลองในส่วนแอพพลิเคชั่นประมวลผลแบบขนานของ Arena

การออกแบบการทดลองในส่วนแอพพลิเคชั่นประมวลผลแบบขนานของ Arena ผู้วิจัยได้ใช้แบบจำลอง Testing and Repair Shop จากหนังสือ Simulation Modeling And Arena สำนักพิมพ์ Wiley มาเป็นแบบจำลองเพื่อใช้ในการทดสอบการทำงาน ว่าสามารถประยุกต์ใช้งานได้ครบทั้ง 2 รูปแบบ อันประกอบด้วย ประมวลผลทีละชุดพารามิเตอร์ และ ประมวลผลทีละหลายชุดพารามิเตอร์ หรือไม่ โดยต้องการทดสอบสมมุติฐาน 2 ประการคือ

1. แอพพลิเคชั่นสามารถใช้งานได้ตามที่ผู้วิจัยได้ทำการออกแบบ

2. การรันแบบขนานของโปรแกรมใช้เวลาการทำงานน้อยกว่าการรันแบบเครื่องเดี่ยว

2.1 Client แต่ละเครื่องช่วยกันประมวลผลแบบจำลองชุดพารามิเตอร์เดียวกันโดยการแบ่งรอบทำซ้ำ ปัจจัยที่นำมาวิเคราะห์ คือ เวลาเฉลี่ยที่ได้จากการทำงานของแอพพลิเคชั่นในแบบ

เครื่องเดี่ยวและแบบขนาน จำนวนเครื่อง Client คือ 1, 2, 4 และ 5 เครื่อง กระจายชุดพารามิเตอร์ไปยังเครื่อง Client 4 แบบด้วยกันคือ

2.1.1 จำนวนเครื่อง Client 1 เครื่อง รอบทำซ้ำเท่ากับ 100 รอบ

2.1.2 จำนวนเครื่อง Client 2 เครื่อง รอบทำซ้ำของแต่ละเครื่องเท่ากับ 50 รอบ

2.1.3 จำนวนเครื่อง Client 4 เครื่อง รอบทำซ้ำของแต่ละเครื่องเท่ากับ 25 รอบ

2.1.4 จำนวนเครื่อง Client 5 เครื่อง รอบทำซ้ำของแต่ละเครื่องเท่ากับ 20 รอบ โดยเป็นการกระจายจำนวนรอบจาก 100 รอบและกำหนดให้แต่ละเครื่องมี Seed Number แตกต่างกัน

2.2 Client 1 เครื่อง 1 ชุดพารามิเตอร์ โดยการกระจายชุดพารามิเตอร์ให้แต่ละเครื่อง Client ปัจจัยที่นำมาวิเคราะห์ คือ เวลาเฉลี่ยที่ได้จากการทำงานของแอพพลิเคชั่นในแบบเครื่องเดี่ยวและแบบขนาน จำนวนเครื่อง Client คือ 1, 2 และ 4 เครื่อง กระจายชุดพารามิเตอร์ไปยังเครื่อง Client 4 แบบด้วยกันคือ

2.2.1 จำนวนเครื่อง Client 1 เครื่อง กระจายครั้งละ 1 ชุดพารามิเตอร์

2.2.2 จำนวนเครื่อง Client 2 เครื่อง กระจายครั้งละ 2 ชุดพารามิเตอร์

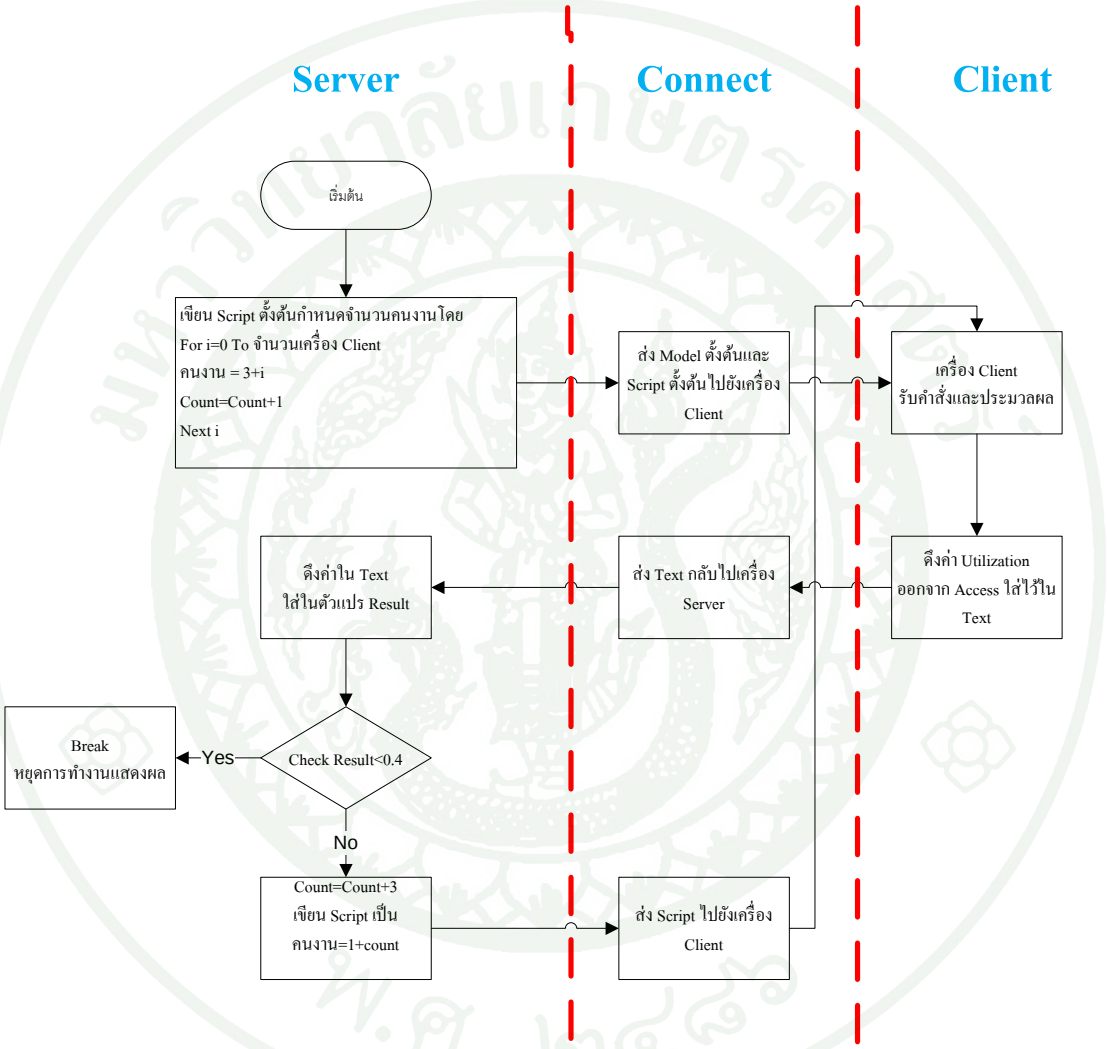
2.2.3 จำนวนเครื่อง Client 4 เครื่อง กระจายครั้งละ 4 ชุดพารามิเตอร์ โดยเป็นการกระจายจาก 4 ชุดพารามิเตอร์โดยทำการประมวลผลพารามิเตอร์ละ 100 รอบ

การออกแบบการทดลองไลบารี่สำหรับการเชื่อมต่อคอมพิวเตอร์

การออกแบบการทดลองในส่วนนี้ผู้วิจัยต้องการทดสอบในเรื่องประยุกต์ใช้ในเชิง Logic เป็นหลัก โดยผู้วิจัยได้แบ่งการทดลองเป็น 3 ส่วนดังต่อไปนี้

1. ทดสอบการใช้งานไลบรารีเบื้องต้นโดยนำแบบจำลอง Testing and Repair Shop จากหนังสือ Simulation Modeling And Arena สำนักพิมพ์ Wiley มาเป็นแบบจำลองเพื่อใช้ในการทดสอบ

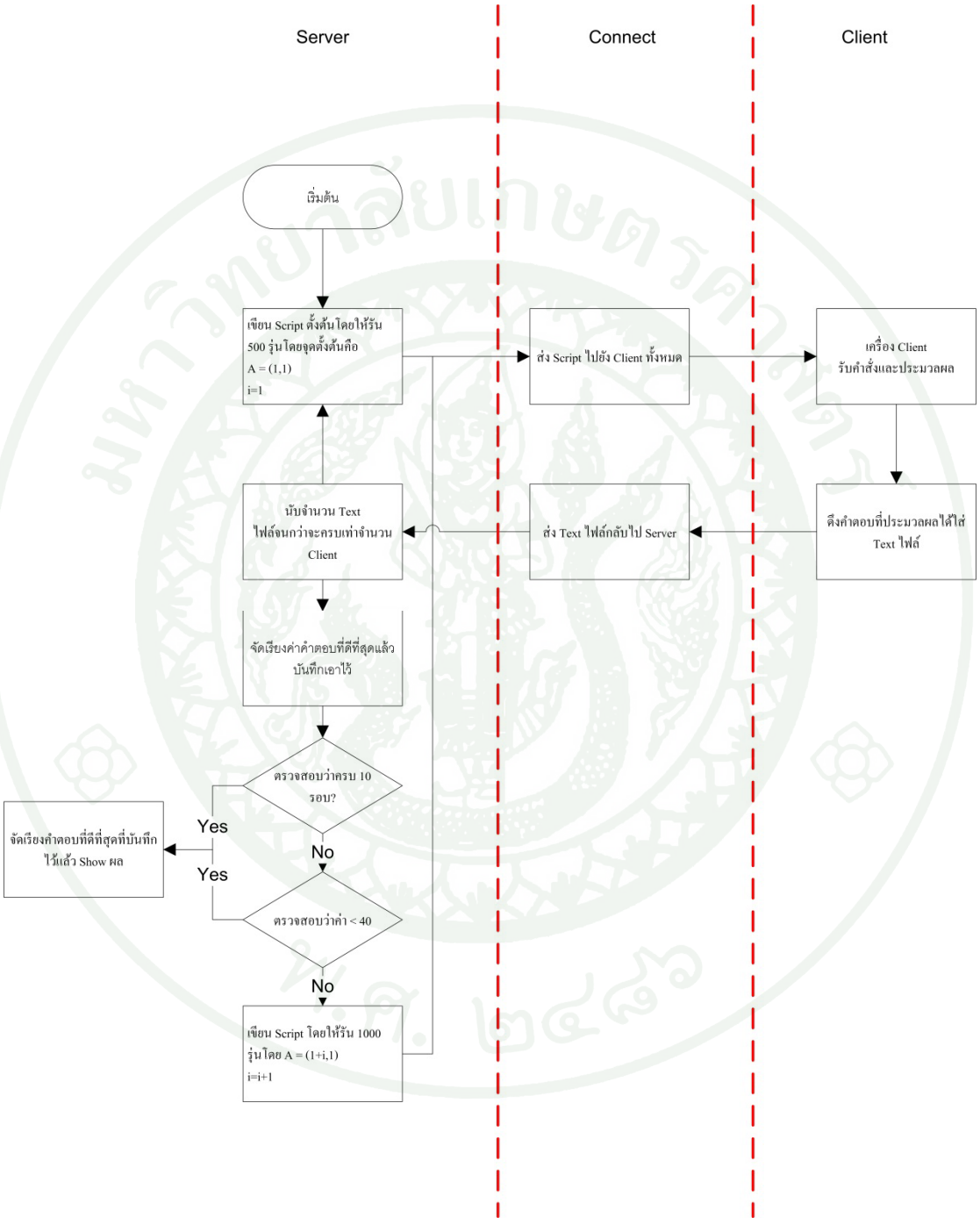
การทำงานเชิง Logic อย่างง่ายโดยกำหนดให้มีการค้นหาหาจำนวน Repairing Worker ที่น้อยที่สุด (Minimize) ในแบบจำลองที่มีเงื่อนไขว่า Utilization ต้องอยู่ต่ำกว่า 0.4 โดยปัญหานี้มีปัจจัยนำเข้าของระบบที่โปรแกรมต้องทำการเปลี่ยนแปลงคือจำนวน Repairing Worker และมีผลที่โปรแกรมต้องคอยตรวจสอบเมื่อมีการเปลี่ยนแปลงค่า Repairing Worker คือค่า Utilization



ภาพที่ 16 Logic ในการหาหาจำนวน Repairing Worker ให้ Utilization อยู่ระหว่าง 0 ถึง 0.4

2. สร้างโปรแกรมค้นหาเส้นทางโดยใช้ Genetic Algorithms ที่อ้างอิงจากหนังสือ Discrete Optimization in Transport and Logistics โดย ดร.ณกร อินทร์พยุงหน้า 147-171 เป็น Search Algorithms ในการหาเส้นทาง เพื่อทำการทดสอบว่าไลบารี่ที่สร้างขึ้นนั้นสามารถที่จะประยุกต์ใช้งานกับโปรแกรม

อื่นนอกจาก Arena ได้หรือไม่ โดยผู้วิจัยได้ออกแบบ Logic สำหรับการควบคุมการค้นหาเส้นทางแบบขนานไว้ดังภาพที่ 17



ภาพที่ 17 Logic ในการหาเส้นทางและที่ตั้งในการขนส่ง

จากภาพที่ 17 เป็น Search Logic ที่ผู้วิจัยได้กำหนดเองสำหรับค้นหาเส้นทางขนส่งโดยมีจุดที่ต้องส่งดังตารางที่ 1 ซึ่งจุด A เป็นจุดที่ใช้สำหรับการกระจายสินค้า สามารถเลือกได้ทั้งหมด 10 จุด โดยสิ่งที่ต้องการคือ จุดที่ตั้งของจุด A และเส้นทางขนส่งที่สั้นที่สุดหรือมีระยะทางไม่เกิน 40 กิโลเมตร

ตารางที่ 1 ตำแหน่งที่ตั้งของจุดขนส่งและจุด A

ตำแหน่ง	พิกัด	
	X	Y
A	1	1
	2	1
	3	1
	4	1
	5	1
	6	1
	7	1
	8	1
	9	1
	10	1
B	5	5
C	9	9
D	3	3
E	12	12
F	13	13
G	5	9
H	7	7
I	14	14
J	12	12
K	1	9
L	7	1
M	11	11
N	15	15

3. สร้างโปรแกรมจัดเรียงลำดับงานเข้าเครื่องจักรด้วยภาษา Visual Basic เพื่อควบคุมให้โปรแกรม Arena ทำการค้นหารูปแบบการจัดเรียงลำดับงานที่ให้ค่าเวลาที่งานทุกงานทำงานเสร็จมีค่าน้อยที่สุด (Minimize C_{max}) ซึ่งอัลกอริทึมที่นำมาสร้างเป็นโปรแกรมค้นหาคำตอบคือ Shifting Bottleneck Heuristic จากหนังสือ Planning and Scheduling in Manufacturing and Services สำนักพิมพ์ Springer เขียนโดย Michael C.Pinedo โดยมีการจัดสร้างโปรแกรมออกเป็น 2 รูปแบบ คือ โปรแกรมที่ทำงานสำหรับการค้นหาคำตอบแบบเครื่องเดี่ยว และโปรแกรมที่ทำงานค้นหาคำตอบแบบขนานโดยใช้คอมพิวเตอร์ 4 เครื่องประมวลผล เพื่อทำการทดสอบการใช้งานไลบรารีในปัญหาที่มีความซับซ้อน และเปรียบเทียบเวลาระหว่างการทำงานแบบเครื่องเดี่ยว และเครื่องขนาน โดยได้กำหนดโจทย์ปัญหาดังนี้

3.1 กำหนดมีงานทั้งหมด 5 งาน มีเครื่องจักร 4 ชนิด ชนิดละ 1 เครื่อง โดยงานมีลำดับของการเข้าเครื่องจักรแต่ละชนิดดังนี้

งานที่ 1-2 มีลำดับเข้าเครื่องจักรคือ 1-2-3-4

งานที่ 3-4 มีลำดับเข้าเครื่องจักรคือ 2-4-1-3

งานที่ 5 มีลำดับเข้าเครื่องจักรคือ 3-1-4-2

มีเวลาการทำงานเป็นการกระจายตัวแบบ Normal Distibution ค่าเบี่ยงเบนมาตรฐาน 0.001โดยมีเวลาเฉลี่ยดังตารางต่อไปนี้

ตารางที่ 2 เวลาทำงานที่งาน 5 งานต้องใช้กับเครื่องจักรแต่ละชนิดมีหน่วยเวลาเป็น ชั่วโมง

Machine	Job				
	1	2	3	4	5
1	0.2790	0.2460	0.1400	0.0630	0.5740
2	0.3100	0.12800	0.5240	0.1640	0.8620
3	0.6950	0.4580	0.4080	0.4080	0.4270
4	0.3460	0.0090	0.1040	0.2310	0.2390

โดยต้องการให้มีการจัดลำดับงานเข้าเครื่องจักรที่ทำให้เวลาที่งานทุกงานทำงานเสร็จมีเวลาน้อยที่สุด (Minimize C_{max}) ซึ่งระบบนี้ปัจจัยนำเข้าคือ รูปแบบการจัดลำดับงานเข้าเครื่องจักร และผลที่ได้คือ เวลาที่งานทุกงานทำงานเสร็จ

3.2 กำหนดมีงานทั้งหมด 10 งาน มีเครื่องจักร 4 ชนิด ชนิดละ 1 เครื่อง โดยงานมีลำดับของการเข้าเครื่องจักรแต่ละชนิดดังนี้

งานที่ 1-3 มีลำดับเข้าเครื่องจักรคือ 1-2-3-4

งานที่ 4-6 มีลำดับเข้าเครื่องจักรคือ 2-1-4-3

งานที่ 7-10 มีลำดับเข้าเครื่องจักรคือ 3-2-4-1

มีเวลาการทำงานเป็นการกระจายตัวแบบ Normal Distibution ค่าเบี่ยงเบนมาตรฐาน 0.001โดยมีเวลาเฉลี่ยดังตารางต่อไปนี้

ตารางที่ 3 เวลาทำงานที่งาน 10 งานต้องใช้กับเครื่องจักรแต่ละชนิดมีหน่วยเวลาเป็น ชั่วโมง

Machine	**Job**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.4904	0.1847	0.8894	0.5963	0.5609	0.3564	0.5425	0.7690	0.6523	0.4851
2	0.4260	0.2750	0.3419	0.7936	0.6502	0.4310	0.3603	0.7160	0.7160	0.9212
3	0.0472	0.6244	0.7104	0.3168	0.7252	0.1550	0.2191	0.7552	0.7552	0.4821
4	0.5264	0.8266	0.8516	0.0463	0.5342	0.0911	0.1921	0.5373	0.5373	0.5382

โดยต้องการให้มีการจัดลำดับงานเข้าเครื่องจักรที่ทำให้เวลาที่งานทุกงานทำงานเสร็จมีเวลาน้อยที่สุด (Minimize C_{max}) ซึ่งระบบนี้ปัจจัยนำเข้าคือ รูปแบบการจัดลำดับงานเข้าเครื่องจักร และผลที่ได้คือ เวลาที่งานทุกงานทำงานเสร็จ

3.3 กำหนดมีงานทั้งหมด 20 งาน มีเครื่องจักร 4 ชนิด ชนิดละ 1 เครื่อง โดยงานมีลำดับของการเข้าเครื่องจักรแต่ละชนิดดังนี้

งานที่ 1-7 มีลำดับเข้าเครื่องจักรคือ 1-2-3-4

งานที่ 8-14 มีลำดับเข้าเครื่องจักรคือ 2-1-4-3

งานที่ 15-20 มีลำดับเข้าเครื่องจักรคือ 3-2-4-1

มีเวลาการทำงานเป็นการกระจายตัวแบบ Normal Distibution ค่าเบี่ยงเบนมาตรฐาน 0.001โดยมีเวลาเฉลี่ยดังตารางต่อไปนี้

ตารางที่ 4 เวลาทำงานที่งาน 20 งานต้องใช้กับเครื่องจักรแต่ละชนิดมีหน่วยเวลาเป็น ชั่วโมง

Machine	**Job**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.9882	0.3156	0.4720	0.8344	0.7253	0.9556	0.8425	0.1493	0.7731	0.1906
2	0.4022	0.4982	0.1578	0.7558	0.1800	0.0561	0.3782	0.5498	0.2788	0.4919
3	0.0769	0.8810	0.3628	0.1419	0.8897	0.1817	0.1167	0.8530	0.5363	0.2609
4	0.2215	0.4683	0.7219	0.4658	0.9220	0.2606	0.7606	0.7645	0.7022	0.6458
Machine	**Job**									
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	0.3144	0.7263	0.8191	0.6347	0.5352	0.9313	0.4826	0.0124	0.8755	0.2068
2	0.6850	0.0873	0.4504	0.2650	0.2111	0.7229	0.2193	0.3445	0.6603	0.9705
3	0.7270	0.6514	0.4444	0.1018	0.5690	0.8478	0.8543	0.6227	0.6336	0.4331
4	0.3983	0.4107	0.5805	0.1162	0.3432	0.0301	0.5795	0.5053	0.2642	0.2842

โดยต้องการให้มีการจัดลำดับงานเข้าเครื่องจักรที่ทำให้เวลาที่งานทุกงานทำงานเสร็จมีเวลาน้อยที่สุด (Minimize C_{max}) ซึ่งระบบนี้ปัจจัยนำเข้าคือ รูปแบบการจัดลำดับงานเข้าเครื่องจักร และผลที่ได้คือ เวลาที่งานทุกงานทำงานเสร็จ

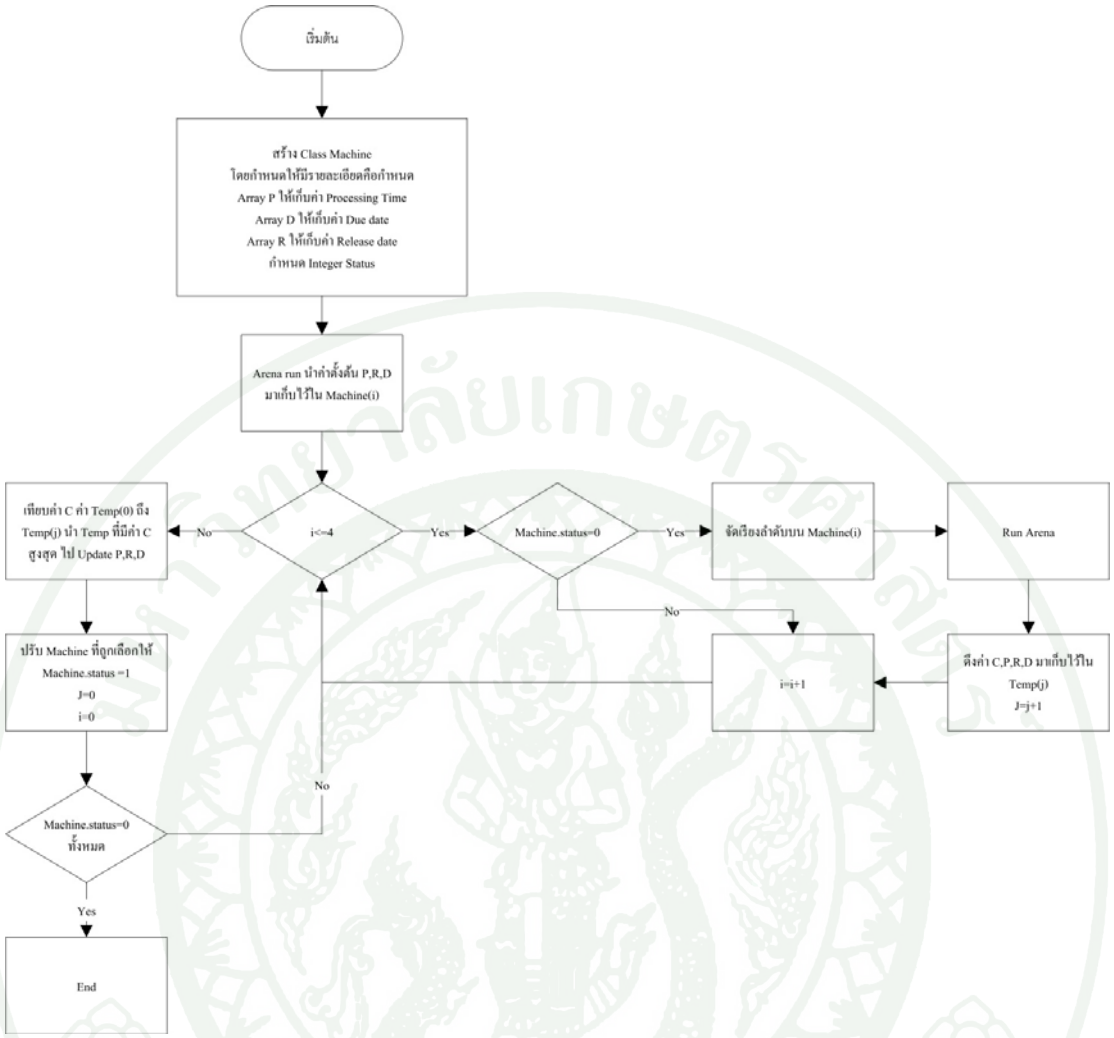
จากเนื้อหาที่กล่าวมาข้างต้นผู้วิจัยต้องการที่จะทดสอบสมมุติฐาน 3 ประการคือ

1. ไลบารี่สามารถนำมาประยุกต์ใช้สำหรับประมวลผลแบบขนานกับโปรแกรม Arena ในเชิง Logic ได้

2. การรันแบบขนานโดยใช้ไลบารี่สามารถทำงานได้เร็วกว่าการรันแบบเครื่องเดี่ยว Client แต่ละเครื่องช่วยกันประมวลผลด้วยโปรแกรม Arena ในเชิง Logic ปัจจัยที่นำมาวิเคราะห์ คือ เวลาเฉลี่ยที่ได้จากการทำงานของโปรแกรมในแบบเครื่อง เดี่ยวและแบบขนาน จำนวนเครื่อง Client คือ 1, 2, 3 และ 4 เครื่อง รันทั้งหมด 50 รอบทำซ้ำ โดยเป็นการทำตาม Logic ที่ได้ออกแบบไว้ดังภาพแล้วนำเวลามาเปรียบเทียบกัน

3. ไลบารี่สามารถนำมาประยุกต์ใช้สำหรับประมวลผลแบบขนานกับโปรแกรมอื่นๆได้ โดยทำการทดสอบให้ประมวลผลแบบขนานด้วย Client 4 เครื่องพร้อมทั้งให้ Client แต่ละเครื่องประมวลผล 1000 รุ่นต่อ 1 ตำแหน่งของจุด A ที่เปลี่ยนไป

4. ไลบารี่สามารถนำมาประยุกต์ใช้สำหรับการหาคำตอบแบบขนานผ่านโปรแกรม Arena ในปัญหาที่มีความซับซ้อนได้ พร้อมทั้งเวลาการหาคำตอบใช้เวลาน้อยกว่าการหาคำตอบแบบเครื่องเดียว ปัจจัยที่นำมาวิเคราะห์คือการทำงานของโปรแกรม และเวลาในการประมวลผลของโปรแกรมแบบเดี่ยว และโปรแกรมแบบขนานเปรียบเทียบกัน



ภาพที่ 18 Logic ในการหาจัดลำดับงานเข้าเครื่องจักร

ภาพที่ 18 เป็น Logic ในการจัดลำดับงานเข้าเครื่องจักร โดยมีหลักการคือตั้งต้นตอนแรกเครื่องจักรจะไม่มีการจัดลำดับงานจึงให้ Arena รันเพื่อนำค่า Processing Time, Release Date,Due Date มาใช้ในการจัดรูปแบบโดยรอบแรกจะจัดลำดับงานของเครื่องจักรทุกเครื่องก่อนแล้วเลือกรูปแบบการจัดของเครื่องจักรที่ให้ค่า C_{max} มากที่สุดมาใช้ แล้วทำการปรับค่า Processing Time, Release Date, Due Date โดยเอามาจากเครื่องจักรที่เราเลือก ทำการจัดไปเรื่อยๆ จนกว่าจะครบ

ผลการวิจัยและวิจารณ์

ผลการทดลองในส่วนแอพพลิเคชั่นประมวลผลแบบขนานของ Arena

ผลการทดสอบสมมุติฐานที่ 1 แอพพลิเคชั่นสามารถใช้งานได้ตามที่ผู้วิจัยได้ทำการออกแบบ

จากการทดลองรันแอพพลิเคชั่นประยุกต์ใช้งานจากการทดลองนี้ผู้วิจัยได้ทดลองในส่วนของโปรแกรมว่าสามารถทำงานได้ถูกต้องตามที่ออกแบบการใช้งานได้หรือไม่ โดยใช้ไฟล์โมเดล Testing and Repair Shop จากหนังสือ Simulation Modeling And Arena สำนักพิมพ์ Wiley มาเป็นแบบจำลองเพื่อใช้ในการทดสอบการทำงาน ซึ่งกำหนดให้มีการรันที่เครื่อง Client 30 รอบ และส่ง Database ให้มาเป็นชื่อ Test ผลจากการทดลองสามารถสรุปได้ว่าโปรแกรมสามารถใช้งานได้จริงตามที่ได้ออกแบบไว้ โดยเครื่อง Client ทำการรันจำนวน 30 รอบและส่ง Database กลับมาในชื่อ Test สามารถตรวจสอบได้ว่าแอพพลิเคชั่นที่สร้างขึ้นนั้นสามารถทำงานได้ตามกระบวนการที่ผู้วิจัยได้ออกแบบอย่างถูกต้อง ดังภาพที่ 10

ผลการทดสอบสมมุติฐานที่ 2 การรันแบบขนานของแอพพลิเคชั่นใช้เวลาทำงานน้อยกว่าการรันแบบเครื่องเดี่ยว

การเปรียบผลด้านเวลาโดยการประมวลผลด้วยโปรแกรม Arena ระหว่างการประมวลผลแบบเครื่องเดี่ยว และแบบขนานนั้น สามารถเปรียบเทียบระหว่าง 2 แบบ คือ

แบบที่ 1 Client แต่ละเครื่องช่วยกันประมวลผลแบบจำลองชุดพารามิเตอร์เดียวกัน โดยการแบ่งรอบทำซ้ำ

แบบที่ 2 Client 1 เครื่อง 1 ชุดพารามิเตอร์ โดยการกระจายชุดพารามิเตอร์ให้แต่ละเครื่อง Client

ผลการทดสอบสมมุติฐานที่ 2.1 Client แต่ละเครื่องช่วยกันประมวลผลแบบจำลองชุดพารามิเตอร์เดียวกันโดยการแบ่งรอบทำซ้ำ

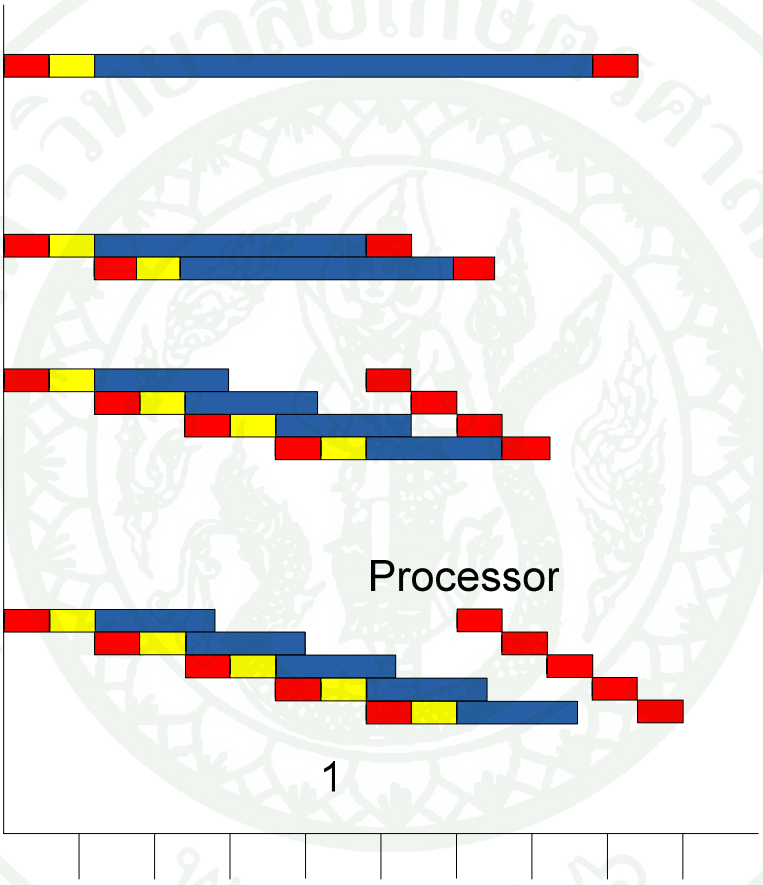
ในการทดลองได้กระจายชุดพารามิเตอร์ไปยังเครื่อง Client 4 แบบด้วยกันคือ

1. 1 เครื่อง รอบทำซ้ำเท่ากับ 100 รอบ

2. 2 เครื่อง รอบทำซ้ำของแต่ละเครื่องเท่ากับ 50 รอบ

3. 4 เครื่อง รอบทำซ้ำของแต่ละเครื่องเท่ากับ 25 รอบ

5. 5 เครื่อง รอบทำซ้ำของแต่ละเครื่องเท่ากับ 20 รอบ



ภาพที่ 19 ชาร์ตแสดงการทำงานของระบบกรณีแบ่งรอบทำซ้ำ

จากภาพที่ 19 แสดงถึงการทำงานในการกระจายชุดพารามิเตอร์ให้กับเครื่อง Client จำนวน 1, 2, 4 และ 5 เครื่อง ซึ่งในการทำงานของแอพพลิเคชั่นจะถือว่าเสร็จสิ้นต่อเมื่อได้รับผลตอบกลับจาก Client ทุกเครื่องเป็นที่เรียบร้อยแล้ว แถบสีเหลืองคือเวลาในการส่งสคริปไฟล์ แถบสีแดงแสดงถึงการรับส่งไฟล์ทั่วไป แถบสีน้ำเงินแสดงถึงเวลาในการประมวลผล

โดยแสดงให้เห็นถึงเวลาในส่วนต่างๆ ดังนี้ เวลาที่เครื่อง Server ใช้ในการส่งไฟล์แบบจำลอง (Model), เวลาที่เครื่อง Server ใช้ในการส่งไฟล์คำสั่ง (Script), เวลาที่เครื่อง Client ใช้ในการประมวลผลแบบจำลองด้วยโปรแกรม Arena (Run Arena), เวลาที่เครื่อง Client ใช้ในการส่งไฟล์ฐานข้อมูลกลับมา (Return), เวลาในการรอของเครื่อง Client แต่ละเครื่อง (Wait Time), เวลาที่ใช้ในการทำงาน (Process Time), เวลาที่มากที่สุดที่ใช้ในแต่ละรอบการกระจาย (Max Time) และ เวลาทั้งหมดที่ใช้การทำงานและแสดงผล (Total Time) ซึ่งสามารถ แบ่งเวลาการทำงานออกเป็น 2 กลุ่ม ดังนี้

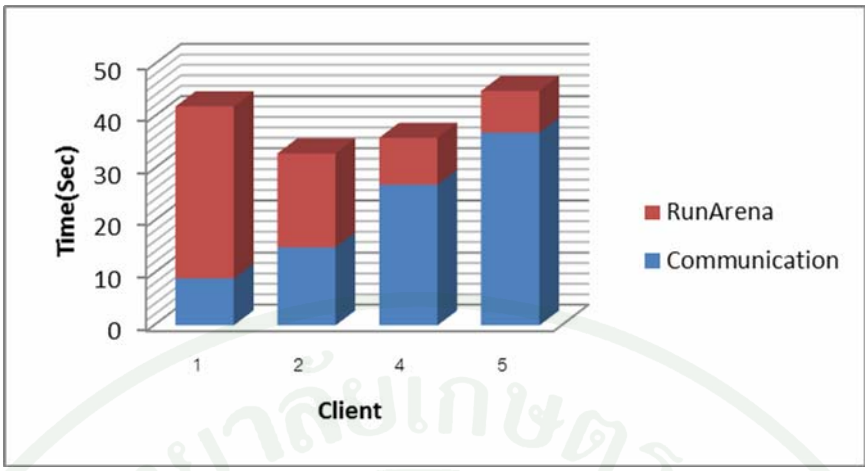
1. เวลาในการรับส่งข้อมูล(Communicate) : Model, Set Arena, Return, Wait time

2. เวลาในการกระมวลผลด้วยโปรแกรม Arena : Run Arena

ตารางที่ 5 การเปรียบเทียบเวลาในการประมวลผลของการกระจายรอบทำซ้ำ (วินาที)

Processor	Reps	Model	Script	Run	Return	Wait Time	Process Time	Max Time	Total Time
1	100	3	3	33	3	0	42	42	42
2	50	3	3	18	3	0	27		
	50	3	3	18	3	6	33	33	33
4	25	3	3	9	3	9	27		
	25	3	3	9	3	12	30		
	25	3	3	9	3	15	33		
	25	3	3	9	3	18	36	36	36
5	20	3	3	8	3	16	33		
	20	3	3	8	3	19	36		
	20	3	3	8	3	22	39		
	20	3	3	8	3	25	42		
	20	3	3	8	3	28	45	45	45

จากตารางที่ 5 แสดงให้เห็นว่าเมื่อจำนวนรอบทำซ้ำลดลง เวลาในการประมวลผลด้วยโปรแกรม Arena จะลดลงด้วย แต่เมื่อดูเวลารวมในการประมวลผลชุดพารามิเตอร์หนึ่งชุดพารามิเตอร์นั้นจะเห็นว่าประมวลผลแบบขนานจะสามารถทำงานได้ดีก็ต่อเมื่อใช้จำนวนเครื่องที่เหมาะสมเนื่องจากเวลาที่เสียไป มีส่วนของเวลาในการรับ-ส่งไฟล์ซึ่งถ้าใช้จำนวนเครื่องมากเกินไปก็จะส่งผลให้เวลาในการรับ-ส่งไฟล์ ใช้เวลามากกว่าในการประมวลผล



ภาพที่ 20 กราฟสัดส่วนเวลาในการทำงานของการกระจายรอบทำซ้ำที่ได้จากระบบ

จากภาพที่ 20 เป็นการแสดงสัดส่วนของเวลาในการทำงานของการกระจายรอบทำซ้ำที่ได้จากระบบซึ่งได้แบ่งเป็นกลุ่มดังที่กล่าวไว้ข้างต้น กราฟแสดงให้เห็นว่า เมื่อใช้จำนวน Client เพิ่มมากขึ้นจะต้องเสียเวลาในการรับส่งไฟล์เพิ่มมากขึ้นตามไปด้วย ถึงแม้ว่าเวลาในการประมวลผลแบบจำลองจะลดลงก็ตาม

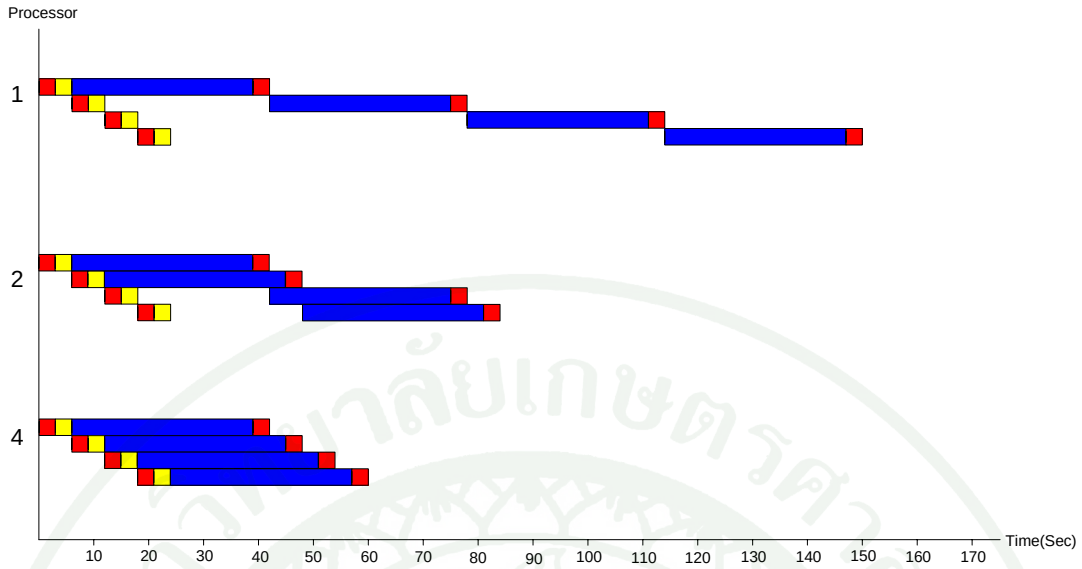
ผลการทดสอบของสมมติฐานที่ 2.2 Client 1 เครื่อง 1 ชุดพารามิเตอร์ โดยการกระจายชุดพารามิเตอร์ให้แต่ละเครื่อง Client

การประยุกต์ของแอพพลิเคชั่นแบบที่ 2 เป็นการกระจายชุดพารามิเตอร์ให้เครื่อง client แต่ละเครื่องช่วยกันประมวลผล โดยผลที่ได้จะเป็นอิสระต่อกัน ไม่มีความเกี่ยวข้องกันแต่อย่างใด

ในการทดลองได้กระจายชุดพารามิเตอร์ 4 ชุดพารามิเตอร์ไปยังเครื่อง Client โดยกระจายชุดพารามิเตอร์ไปยังเครื่อง Client 3 แบบด้วยกันคือ

1. จำนวนเครื่อง Client 1 เครื่อง กระจายครั้งละ 1 ชุดพารามิเตอร์
2. จำนวนเครื่อง Client 2 เครื่อง กระจายครั้งละ 2 ชุดพารามิเตอร์
3. จำนวนเครื่อง Client 4 เครื่อง กระจายครั้งละ 4 ชุดพารามิเตอร์

ทุกชุดพารามิเตอร์มีจำนวนรอบทำซ้ำเท่ากันคือ 100 รอบทำซ้ำ



ภาพที่ 21 ชาร์ตแสดงการทำงานของระบบกรณีกระจายชุดพารามิเตอร์

จากภาพที่ 21 แสดงถึงการทำงานในการกระจายชุดพารามิเตอร์ให้กับเครื่อง Client จำนวน 1, 2 และ 4 เครื่อง ซึ่งในการทำงานของแอพพลิเคชั่นจะถือว่าเสร็จสิ้นต่อเมื่อได้รับผลตอบกลับจาก Client ทุกเครื่องเป็นที่เรียบร้อยแล้ว

ภาพที่ 22 กราฟเวลาในการทำงานกรณีกระจายชุดพารามิเตอร์

ภาพที่ 22 แสดงตัวอย่างผลด้านเวลาในการประมวลผลชุดพารามิเตอร์ 4 ชุดพารามิเตอร์จากระบบ โดยแสดงผลเป็นวินาที โดยเมื่อทำงานด้วยเครื่อง Client 1 เครื่อง ใช้เวลา 150 วินาที ทำงานด้วยเครื่อง Client 2 เครื่อง ใช้เวลา 84 วินาทีและ ทำงานด้วยเครื่อง Client 4 เครื่อง ใช้เวลา 60 วินาที ซึ่งสามารถสรุปได้ว่าถ้าจำนวนเครื่อง Client มีจำนวนมากขึ้น เวลาในการทำงานยิ่งน้อยลง

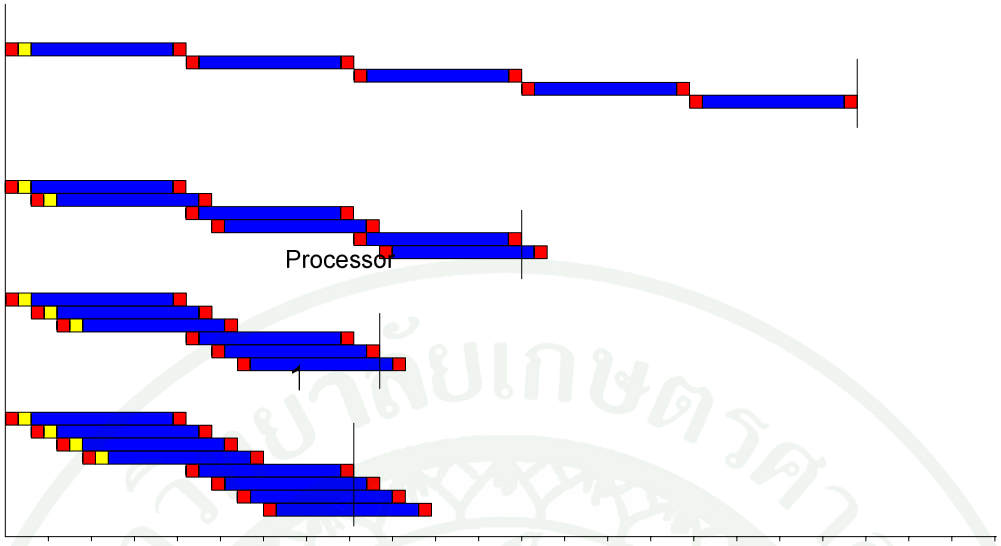
ผลทดลองในส่วนการออกแบบการทดลองไลบารี่สำหรับการเชื่อมต่อคอมพิวเตอร์

ผลการทดสอบสมมุติฐานที่ 1 ไลบารี่สามารถนำมาประยุกต์ใช้สำหรับประมวลผลแบบขนานกับโปรแกรม Arena ในเชิง Logic ได้

จากการทดลองนำไลบารี่ไปสร้างโปรแกรมประยุกต์ใช้งานตามรูปแบบ Logic ที่ออกแบบดังภาพที่ 11 สามารถตรวจสอบได้ว่าโปรแกรมที่สร้างโดยไลบารี่นั้นสามารถทำงานได้ตามกระบวนการที่ผู้วิจัยได้ทำการออกแบบอย่างถูกต้อง พร้อมทั้งได้คำตอบในการค้นหาคือ ต้องใช้ Repair Worker ทั้งหมด 7 คน ซึ่งผู้วิจัยได้นำคำตอบนี้ไปตรวจสอบก็เป็นคำตอบที่ถูกต้อง

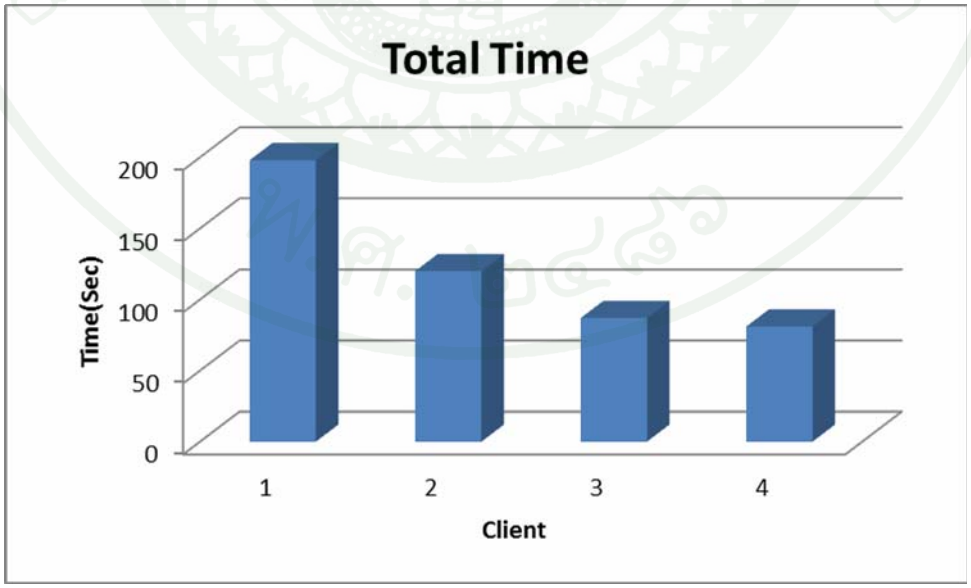
ผลการทดสอบสมมุติฐานที่ 2 การรันแบบขนานโดยใช้ไลบารี่สามารถทำงานได้เร็วกว่าการรันแบบเครื่องเดี่ยว

การเปรียบผลด้านเวลาโดยการประยุกต์ใช้ไลบารี่สร้างโปรแกรมประยุกต์แบบ Logic ระหว่างการประมวลผลแบบเครื่องเดี่ยว และแบบขนานนั้น ในงานวิจัยได้ใช้ Search Logic ดังภาพที่ 16 โดยทำการทดลองประมวลผลด้วยโปรแกรม Arena ระหว่างการประมวลผลแบบเครื่องเดี่ยวและแบบขนานจำนวน 2,3,4 เครื่องโดยใช้การรัน 50 รอบทำซ้ำ

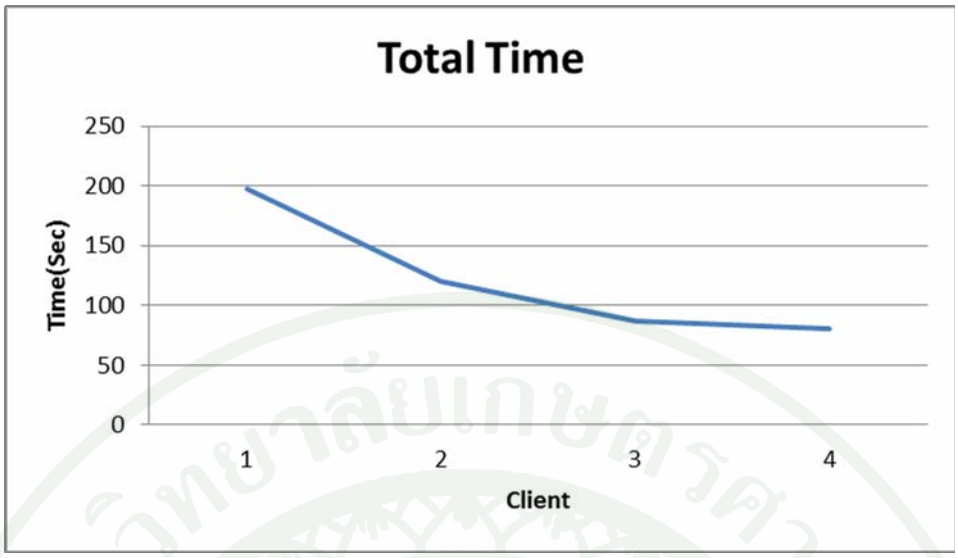


ภาพที่ 23 ชาร์ตแสดงการทำงานกรณี Search Logic

จากภาพที่ 23 แสดงถึงการทำงานในการหาคำตอบของปัญหาโดยมีการทดลองให้เครื่อง Client จำนวน1, 2, 3 และ 4 เครื่อง ช่วยกันหาคำตอบตาม Search Logic ซึ่งในการทำงานของโปรแกรมจะถือว่าเสร็จสิ้นต่อเมื่อได้รับผลตอบตามเงื่อนไขของ Search Logic ที่ตรวจสอบจากเครื่อง Server เท่านั้น



ภาพที่ 24 กราฟแท่งเวลาในการทำงานกรณี Search Logic



ภาพที่ 25 กราฟเส้นแสดงแนวโน้มเวลาในการทำงานกรณี Search Logic

ภาพที่ 25 แสดงตัวอย่างผลด้านเวลาในการประมวลผลตาม Search Logic โดยแสดงผลเป็นวินาที โดยเมื่อทำงานด้วยเครื่อง Client 1 เครื่อง ใช้เวลา 198 วินาที ทำงานด้วยเครื่อง Client 2 เครื่อง ใช้เวลา 120 วินาที ทำงานด้วยเครื่อง Client 3 เครื่อง ใช้เวลา 87 วินาทีและ ทำงานด้วยเครื่อง Client 4 เครื่อง ใช้เวลา 81 วินาที ซึ่งสามารถสรุปได้ว่าถ้าจำนวนเครื่อง Client มีจำนวนมากขึ้น เวลาในการทำงานยิ่งน้อยลง แต่ก็จะมีแนวโน้มหยุดนิ่งเมื่อถึงจุดๆหนึ่งดังภาพที่ 24

ผลการทดสอบสมมุติฐานที่ 3 ไลบารี่สามารถนำมาประยุกต์ใช้สำหรับประมวลผลแบบขนานกับโปรแกรมอื่นๆ ได้

จากการทดลองนำไลบารี่ไปสร้างโปรแกรมประยุกต์ใช้งานกับโปรแกรมค้นหาเส้นทางที่ผู้วิจัยได้เขียนเองตามรูปแบบ Logic ที่ออกแบบดังภาพที่ 17 สามารถตรวจสอบได้ว่าโปรแกรมที่สร้างโดยไลบรารีนั้นสามารถใช้งานกับโปรแกรมค้นหาเส้นทางที่ผู้วิจัยได้เขียนเอง โดยคำตอบที่ได้คือ จุด A ต้องตั้งในตำแหน่ง (7,1) และมีเส้นทางดังนี้ A-L-B-H-C-M-J-E-N-I-F-G-K-D-A มีระยะทาง 45.18

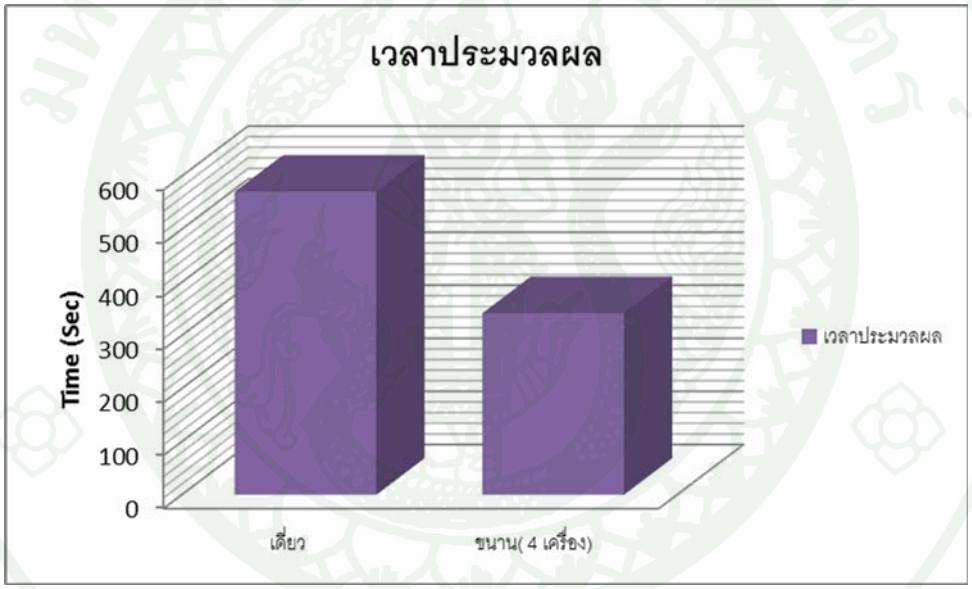
ผลการทดสอบสมมุติฐานที่ 4 ไลบารี่สามารถนำมาประยุกต์ใช้กับปัญหาที่ซับซ้อนและเวลาการประมวลผลแบบเดี่ยวเปรียบเทียบกับเวลาการประมวลผลแบบขนาน

จากการทดลองกรณีปัญหา งาน 5 งานเครื่องจักร 4 ชนิดรันหาคำตอบ 100 รอบทำซ้ำได้คำตอบดังนี้คือ

ตารางที่ 6 แสดงผลการจัดลำดับงานเข้าเครื่องจักรกรณีงาน 5 งานเครื่องจักร 4 ชนิด

Machine	ลำดับงาน				
	1	**2**	**3**	**4**	**5**
1	1	2	4	5	3
2	4	3	1	2	5
3	5	4	1	2	3
4	4	3	5	2	1

จากการจัดลำดับดังตารางที่ 6 ได้เวลาที่งานทั้งหมดทำเสร็จคือ 2.561 ชั่วโมงโดยเวลาเปรียบเทียบระหว่างการประมวลผลแบบเดี่ยว และแบบขนาน 4 เครื่องเป็นภาพที่ 25



ภาพที่ 26 เปรียบเทียบเวลาประมวลผลกรณีงาน 5 งานเครื่องจักร 4 ชนิด

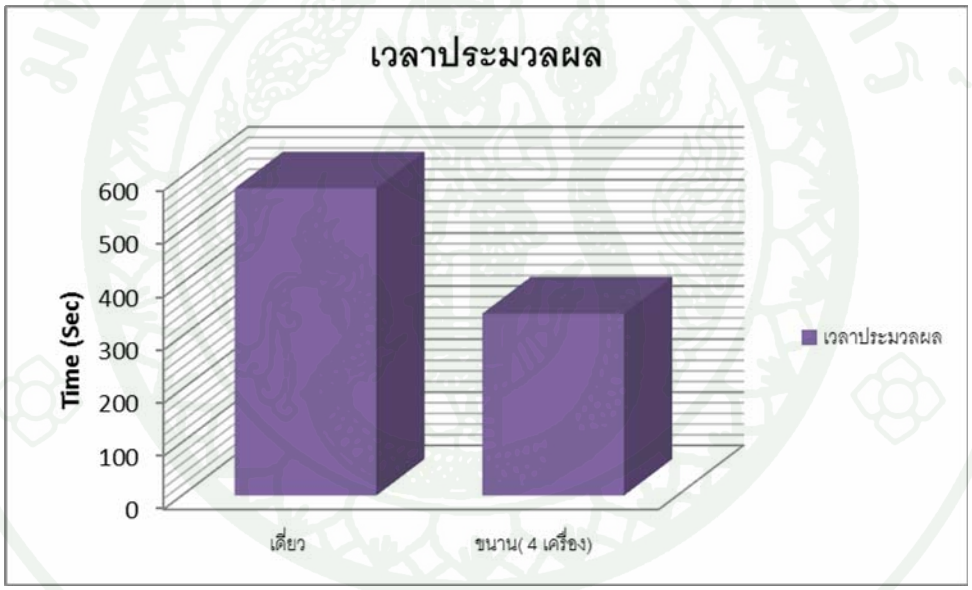
จากภาพที่ 26 การประมวลผลแบบเดี่ยวใช้เวลาการประมวลผล 571 วินาที การประมวลผลแบบขนานใช้เวลา 342 วินาที โดยลดเวลาการประมวลผลลง 229 วินาทีหรือลดเวลาได้ 40% ของเครื่องเดี่ยว

จากการทดลองกรณีปัญหา งาน 10 งานเครื่องจักร 4 ชนิดรันหาคำตอบ 100 รอบทำซ้ำได้คำตอบดังนี้คือ

ตารางที่ 7 แสดงผลการจัดลำดับงานเข้าเครื่องจักรกรณีงาน 10 งานเครื่องจักร 4 ชนิด

Machine	**ลำดับงาน**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	3	2	5	1	4	6	7	8	9	10
2	5	4	6	2	7	9	3	10	8	1
3	10	9	7	8	5	4	2	6	3	1
4	5	4	6	7	9	2	10	3	8	1

จากการจัดลำดับดังตารางที่ 7 ได้เวลาที่งานทั้งหมดทำเสร็จคือ 8.495 ชั่วโมงโดยเวลาเปรียบเทียบระหว่างการประมวลผลแบบเดี่ยว และแบบขนาน 4 เครื่องเป็นภาพที่ 26



ภาพที่ 27 เปรียบเทียบเวลาประมวลผลกรณีงาน 10 งานเครื่องจักร 4 ชนิด

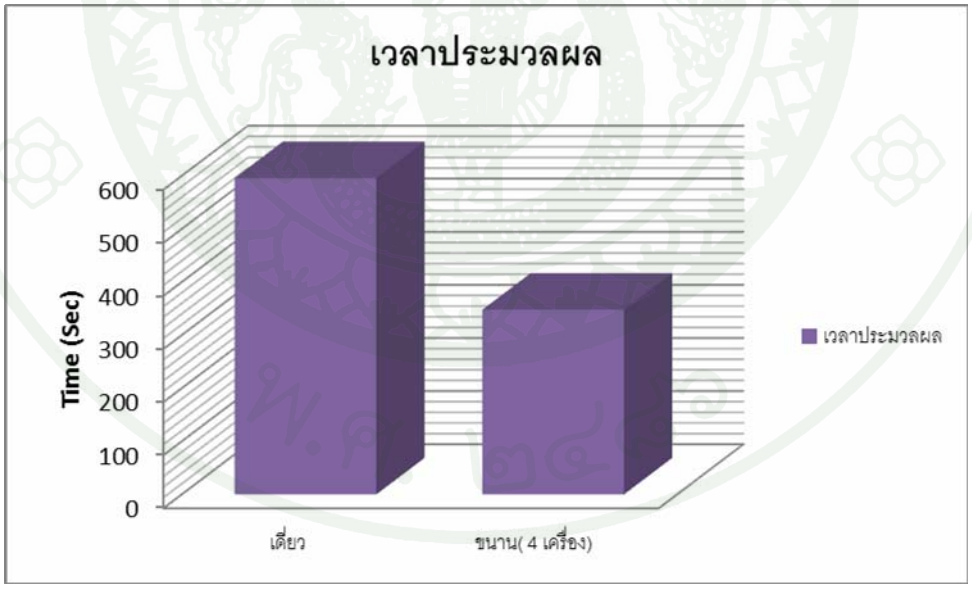
จากภาพที่ 27 การประมวลผลแบบเดี่ยวใช้เวลาการประมวลผล 579 วินาที การประมวลผลแบบขนานใช้เวลา 343 วินาที โดยลดเวลาการประมวลผลลง 236 วินาที หรือลดเวลาได้ 40% จากเครื่องเดี่ยว

จากการทดลองกรณีปัญหา งาน 20 งานเครื่องจักร 4 ชนิดรันหาคำตอบ 100 รอบทำซ้ำได้คำตอบดังนี้คือ

ตารางที่ 8 แสดงผลการจัดลำดับงานเข้าเครื่องจักรกรณีงาน 20 งานเครื่องจักร 4 ชนิด

Machine	**ลำดับงาน**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5	2	8	4	7	3	9	11	12	13
2	8	9	11	12	13	10	14	15	16	17
3	15	16	17	19	18	20	5	7	3	2
4	8	15	16	17	9	11	12	13	18	19
Machine	**ลำดับงาน**									
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	10	6	1	14	16	16	17	18	19	20
2	5	17	2	19	3	7	4	20	1	6
3	4	8	9	11	12	13	10	1	6	14
4	10	20	5	7	3	2	4	14	1	6

จากการจัดลำดับดังตารางที่ 8 ได้เวลาที่งานทั้งหมดทำเสร็จคือ 12.26 ชั่วโมงโดยเวลาเปรียบเทียบระหว่างการประมวลผลแบบเดี่ยว และแบบขนาน 4 เครื่องเป็นภาพที่ 28



ภาพที่ 28 เปรียบเทียบเวลาประมวลผลกรณีงาน 10 งานเครื่องจักร 4 ชนิด

จากภาพที่ 28 การประมวลผลแบบเดี่ยวใช้เวลาการประมวลผล 596 วินาที การประมวลผลแบบขนานใช้เวลา 348 วินาที โดยลดเวลาการประมวลผลลง 248 วินาที หรือลดเวลาได้ 41.6% จากเครื่องเดี่ยว

สรุปและข้อเสนอแนะ

สรุป

จากการศึกษางานวิจัย เมื่อนำแนวคิดของการประมวลผลแบบขนานมาพัฒนาเป็นแอพพลิเคชั่นควบคุมการประมวลผลแบบขนานของ Arena และไลบรารีสำหรับการเชื่อมต่อระหว่างคอมพิวเตอร์โดยภาษา Visual Basic เครื่องมือที่ได้พัฒนาขึ้นสามารถที่จะใช้งานตาม logic ของแอพพลิเคชั่นและไลบรารีที่ได้ออกแบบไว้อย่างถูกต้อง พร้อมทั้งสามารถนำไปประยุกต์การใช้งานตามแนวคิดของการประมวลผลแบบขนานได้เป็นอย่างดี

ในส่วนของความเร็วในการประมวลผลสามารถที่จะสรุปได้ว่า การที่จะให้การประมวลผลแบบขนานมีความเร็วที่ดีกว่าการประมวลผลแบบเดี่ยวนั้นขึ้นอยู่กับระยะเวลาในการประมวลผลปัญหา และระยะเวลาในการติดต่อกันระหว่างเครื่องคอมพิวเตอร์ ซึ่งถ้าหากระยะเวลาในการประมวลผลหาคำตอบใช้เวลาที่น้อยแล้วก็ไม่จำเป็นที่จะต้องประมวลผลแบบขนานเนื่องจากถ้าใช้การประมวลผลแบบขนานก็จะมีส่วนของเวลาในการติดต่อระหว่างเครื่องคอมพิวเตอร์มาเกี่ยวข้องด้วย อันอาจจะส่งผลให้เวลาของการประมวลผลนั้นช้ากว่าที่ควรจะเป็น ดังนั้นการที่จะเลือกใช้หรือไม่เลือกใช้การประมวลผลแบบขนานนั้นผู้ที่จะใช้ต้องพิจารณาแบบของปัญหา ก่อนตัดสินใจว่าจะใช้หรือไม่ใช้ เพราะถ้าหากไม่ทำการพิจารณาแล้วแทนที่จะลดเวลาในการประมวลผลอาจส่งผลให้เพิ่มเวลาในการประมวลผลก็เป็นได้

งานวิจัยนี้เป็นประโยชน์ต่อการลดเวลาในการหาคำตอบประเภทที่ใช้เวลาหาคำตอบนานๆ โดยเฉพาะกับปัญหาการค้นหาคำตอบผ่านการจำลองสถานการณ์ เนื่องจากปัญหาประเภทนี้ต้องใช้ระยะเวลาในการหาคำตอบที่ยาวนานเพราะต้องมีการเปลี่ยนแปลงพารามิเตอร์พร้อมทั้งจำลองสถานการณ์หลังเปลี่ยนพารามิเตอร์ไปเรื่อยๆจนกว่าจะเจอคำตอบที่ต้องการซึ่งไลบรารีที่ได้ออกแบบมาสามารถที่จะประยุกต์ใช้กับปัญหาประเภทนี้ได้ งานวิจัยนี้จึงถือว่าเป็นอีกทางเลือกหนึ่งที่สามารถประยุกต์ใช้สำหรับการลดเวลาและควบคุมการหาคำตอบในการประมวลแบบขนาน

ข้อเสนอแนะ

แอพพลิเคชั่นการควบคุมการประมวลผลแบบขนานและไลบรารีเป็นเครื่องมือพื้นฐานสำหรับนักวิจัยที่มีความรู้การเขียน Visual Basic สามารถนำไปประยุกต์เขียนโปรแกรมการควบคุมการประมวลผลหรือหาคำตอบแบบขนานสำหรับลดเวลาในการหาคำตอบของปัญหาหลาย ๆ ปัญหาได้ แต่ถ้าหากนักวิจัยไม่มีพื้นฐานการในการเขียน Visual Basic แล้วก็ยากที่จะใช้งานไลบรารีได้ นอกจากนั้นแล้วในการใช้เครื่องมือที่ได้พัฒนาขึ้นก็จำเป็นจะต้องมีระบบทาง Network ที่เสถียร เพราะถ้าไม่เสถียรก็ยากที่จะติดต่อสื่อสารได้ พร้อมทั้งระบบปฏิบัติการต้องเป็น Windows XP เท่านั้นถึงแอพพลิเคชั่นและไลบรารีที่สร้างขึ้นจึงจะใช้งานได้ ดังนั้นในการจะต่อยอดของเครื่องมือชิ้นนี้ก็สามารถต่อยอดได้โดยการพัฒนาให้ใช้กับระบบปฏิบัติการอื่นๆได้ เช่น Windows Seven, Android หรืออื่นๆ เป็นต้น

สำหรับผู้ใช้ที่ต้องการนำเครื่องมือที่ผู้วิจัยได้ออกแบบและพัฒนาขึ้นมา ผู้ใช้จำเป็นจะต้องมีความรู้แล้วความเข้าใจใน Visual Basic พื้นฐานบ้างในกรณีที่จะใช้งานไลบรารี โดยผู้ใช้สามารถที่จะศึกษาวิธีใช้งานได้จาก ภาคผนวก ค และ ง นอกจากนั้นแล้ว ก่อนใช้งานผู้ใช้ต้องตรวจสอบระบบรักษาความปลอดภัยของคอมพิวเตอร์ให้สามารถส่งข้อมูลไป-มาระหว่างกันได้ โดยระบบของ Network ต้องสามารถทำงานได้ พร้อมทั้งระบบปฏิบัติการก็ต้องเป็น Windows XP จึงจะสามารถใช้งานเครื่องมือนี้ได้อย่างเต็มประสิทธิภาพ

เอกสารและสิ่งอ้างอิง

ณกร อินทร์พยุง. 2548. **Discrete Optimization in Transport and Logistics.** ซีเอ็ดยูเคชั่น จำกัด (มหาชน), กรุงเทพฯ.

ธีรณี อาจลากุล. 2551. **การประมวลผลแบบขนานและแบบกระจาย.** ท้อปจำกัด, กรุงเทพฯ.

ปิยะวดี บางโม. 2551. **การพัฒนาระบบการจำลองสถานการณ์แบบกระจายในเครือข่ายคอมพิวเตอร์**. วิทยานิพนธ์ปริญญาโท, มหาวิทยาลัยเกษตรศาสตร์.

เอกสิทธิ์ วิริยจารี. 2548. **เรียนรู้ระบบเน็ตเวิร์กจากอุปกรณ์ของ Cisco ภาคปฏิบัติ.** ซีเอ็ดยูเคชั่น จำกัด (มหาชน), กรุงเทพฯ.

Alléon, G., S. Champagneux, G. Chevalier, L. Giraud and G. Sylvand. 2006. Parallel distributed numerical simulations in aeronautic applications. **Applied Mathematical Modelling** 30(8): 714-730.

Bononi, L., M. Bracuto, G, D'Angelo and L. Donatiello 2005. Concurrent Replication of Parallel and Distributed Simulations. pp. 234-243. *In* **Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation 2005**. IEEE Computer Society, USA.

Bruschi, S.M., R.H.C. Santana, M.J. Santana and T.S. Aiza. 2004. An automatic distributed simulation environment, pp. 378 - 385. *In* **Proceedings of the Winter Simulation Conference.** Society for Computer Simulation International, USA.

Fujimoto, M. 2001. Parallel simulation: parallel and distributed simulation systems. pp. 147-157. *In* **Proceedings of the Winter Simulation Conference**. Society for Computer Simulation International, USA.

Kiesling, T. 2006. Progressive Time-Parallel Simulation. pp. 82-91. *In* **Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation 2006**. IEEE Computer Society, USA.

Michael, L.P. 2004. **Planning and Scheduling in Manufacturing and Services.** Springer, USA.

Mota, E., A. Wolisz and K. Pawlikowski. 2000. A perspective of batching methods in a simulation environment of multiple replications in parallel, pp. 761 - 766. *In* **Proceedings of the Winter Simulation Conference.** Society for Computer Simulation International, USA.

Pawlikowski, K., V. Yau and D. McNickle. 1994. Distributed stochastic discrete-event simulation in parallel time streams, pp. 723 - 730. *In* **Proceedings of Winter Simulation Conference.** Society for Computer Simulation International, USA.

Thole, C.A. and K. Stüben. 1999. Industrial simulation on parallel computers. **Parallel Computing** 25 (13-14): 2015-2037.

Yau, V. 1999. Automating Parallel Simulation Using Parallel Time Streams. **ACMTransactions on Modeling and Computer Simulation** 9 (2): 171–201.

ภาคผนวก

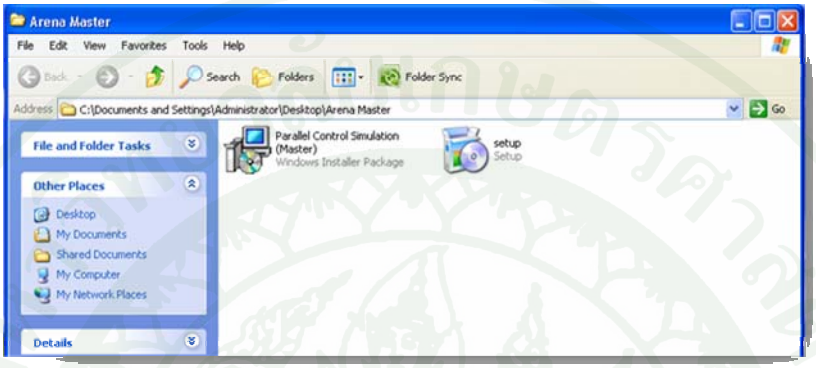
ภาคผนวก ก

การติดตั้งแอพพลิเคชั่น Parallel Control Simulation

การติดตั้งแอพพลิเคชั่น Parallel Control Simulation ฝั่ง Server

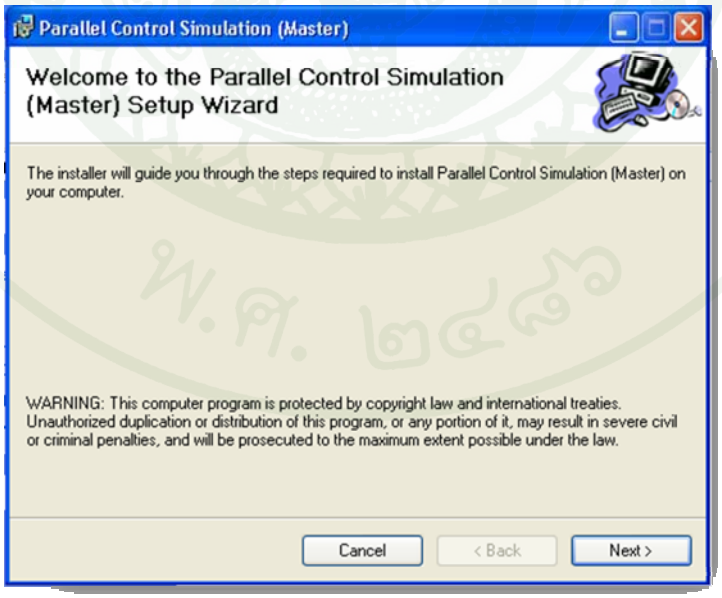
1. Parallel Control Simulation (Master)

1.1 นำ Setup file copy ลงในเครื่องคอมพิวเตอร์ที่จะติดตั้ง ดังภาพที่ ก1



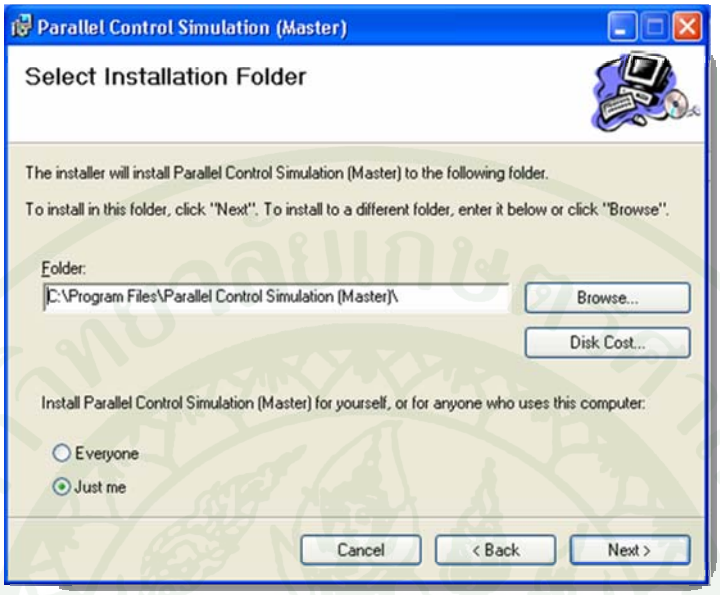
ภาพผนวกที่ ก1 Setup File ฝั่ง Server

1.2 ดับเบิ้ลคลิกที่ Setup File โดยหลังจากดับเบิ้ลคลิก จะปรากฏดังภาพที่ ก2 แล้วกดปุ่ม next เพื่อติดตั้งแอพพลิเคชั่น



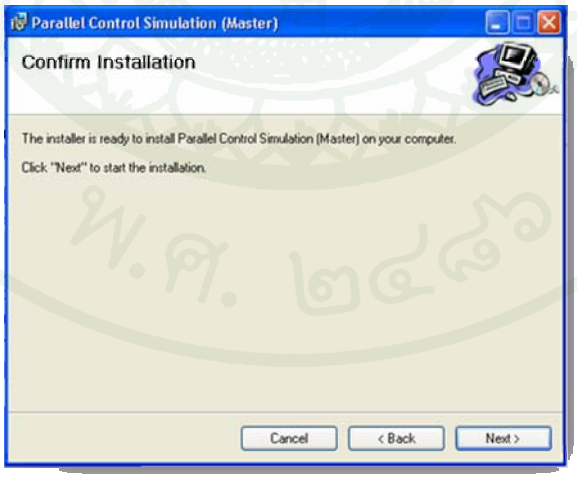
ภาพผนวกที่ ก2 แสดงการติดตั้งของแอพพลิเคชั่นฝั่ง Server

1.3 เลือก path ที่จะติดตั้งแอพพลิเคชั่น แล้วกำหนดสิทธิของผู้ใช้ว่าอนุญาตให้ user ที่ติดตั้งใช้งานเพียง user เดียวเท่านั้น หรืออนุญาตให้ user อื่นสามารถใช้งานได้ ดังภาพที่ ก3



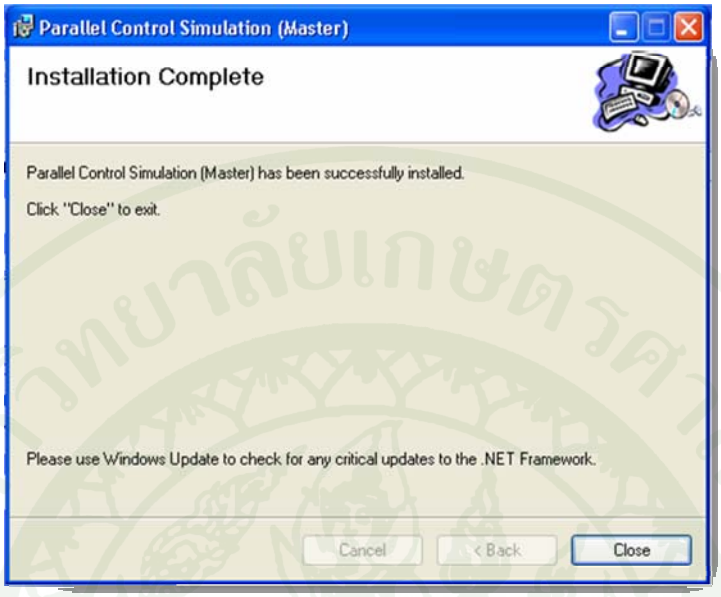
ภาพผนวกที่ ก3 แสดงการกำหนดสิทธิ์และPath ฝั่ง Server

1.4 หลังจากกำหนดสิทธิและ path ที่จะติดตั้งเรียบร้อยแล้ว ให้ทำการคลิกปุ่ม next เพื่อไปยังขั้นตอนถัดไป ดังรูปภาพที่ ก4 แล้วคลิกปุ่ม next



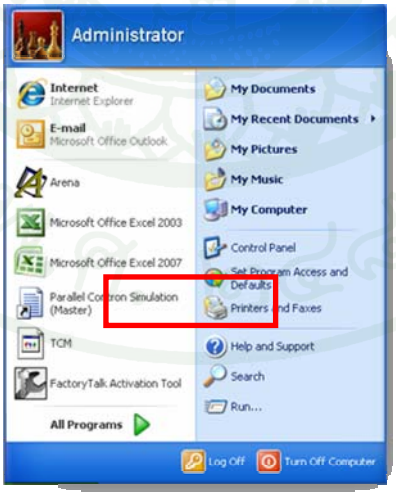
ภาพผนวกที่ ก4 แสดงการตกลงเพื่อติดตั้งแอพพลิเคชั่นฝั่ง Server

1.5 เมื่อเครื่องคอมพิวเตอร์ได้ติดตั้งแอพพลิเคชั่นเสร็จ จะปรากฏรูปดังรูปภาพผนวกที่ ก5 ให้กดปุ่ม close เพื่อเสร็จสิ้นการติดตั้งแอพพลิเคชั่น

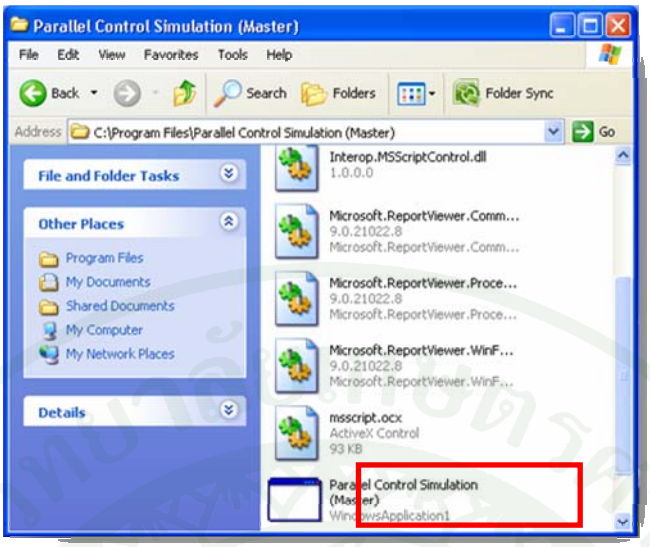


ภาพผนวกที่ ก5 แสดงการติดตั้งเสร็จสิ้นฝั่ง Server

1.6 สามารถเรียกใช้แอพพลิเคชั่นได้จาก path ที่ได้ติดตั้งไว้ โดยดับเบิ้ลคลิกที่ >>> หรือเปิดไปยังเมนู start ดังรูปภาพผนวกที่ ก6 และ ภาพผนวกที่ ก7 ตามลำดับ



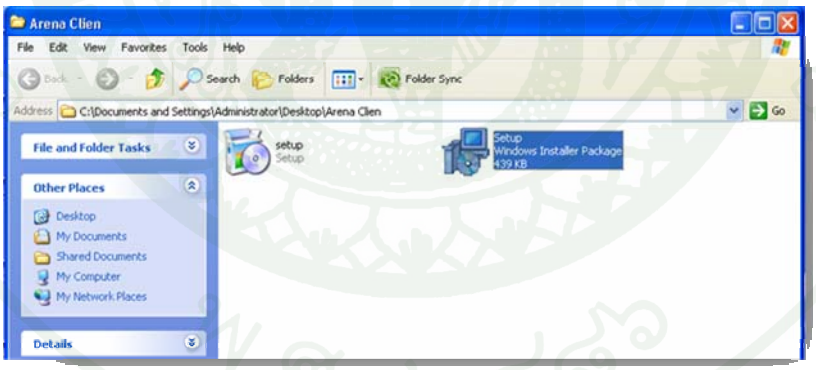
ภาพผนวกที่ ก6 การเปิดแอพพลิเคชั่นผ่าน Short Cut ฝั่ง Server



ภาพผนวกที่ ก7 การเปิดแอพพลิเคชั่นจาก Path ฝั่ง Server

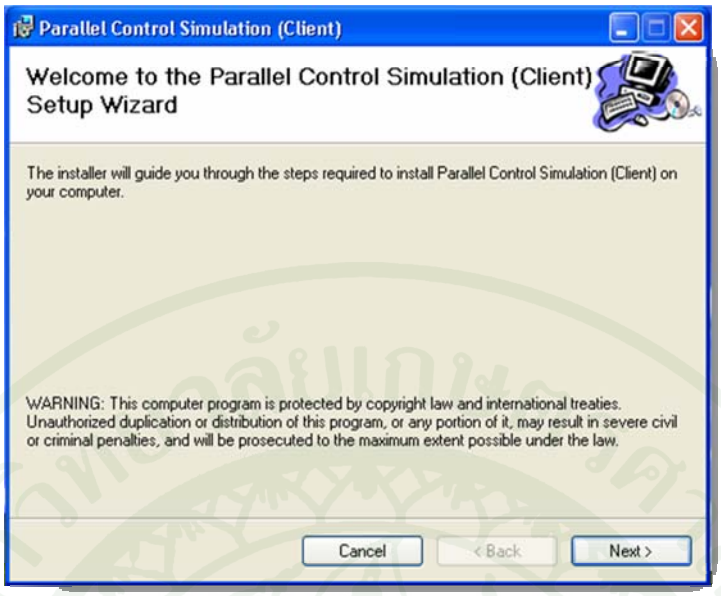
2. Parallel Control Simulation (Client)

2.1 นำ Setup file copy ลงในเครื่องคอมพิวเตอร์ที่จะติดตั้ง ดังภาพผนวกที่ ก8



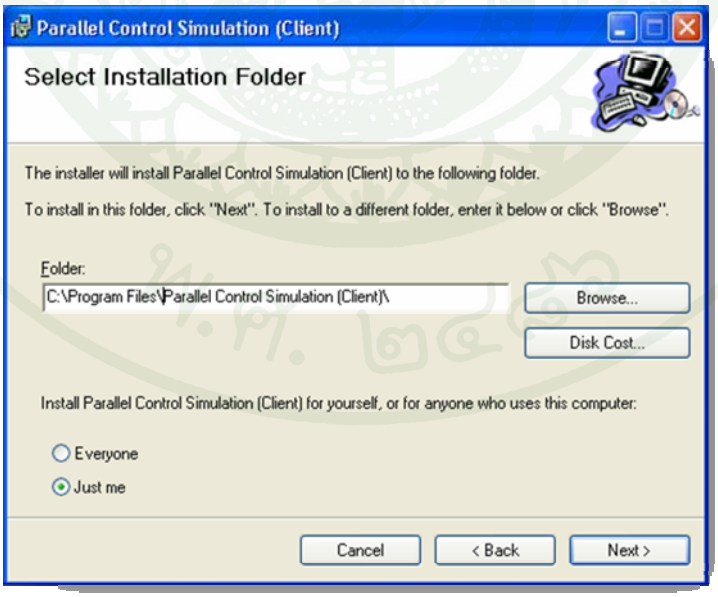
ภาพที่ ก8 Setup File ฝั่ง Client

2.2 ดับเบิ้ลคลิกที่ Setup File โดยหลังจากดับเบิ้ลคลิก จะปรากฏดังภาพผนวกที่ ก9 แล้วกดปุ่ม next เพื่อติดตั้งแอพพลิเคชั่น



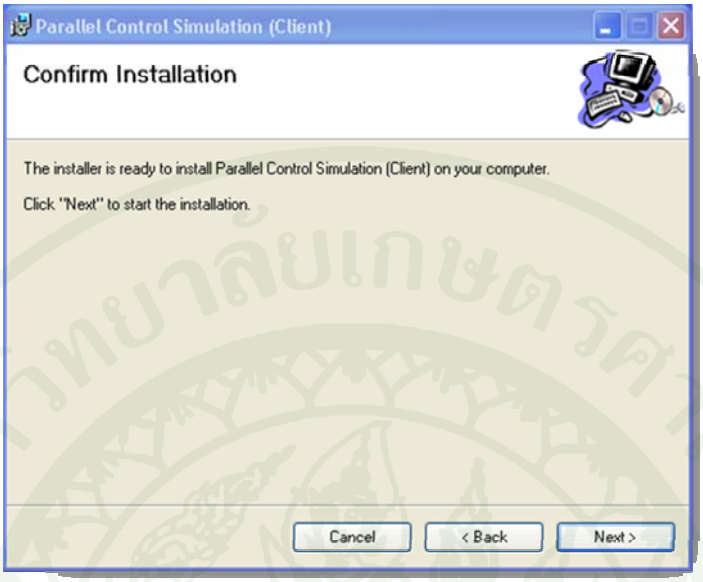
ภาพผนวกที่ ก9 แสดงการติดตั้งของแอพพลิเคชั่นฝั่ง Client

2.3 เลือก path ที่จะติดตั้งแอพพลิเคชั่น และกำหนดสิทธิของผู้ใช้ว่าอนุญาตให้ user ที่ติดตั้งใช้งานเพียง user เดียวเท่านั้น หรืออนุญาตให้ user อื่นสามารถใช้งานได้ ดังรูปภาพผนวกที่ ก 10



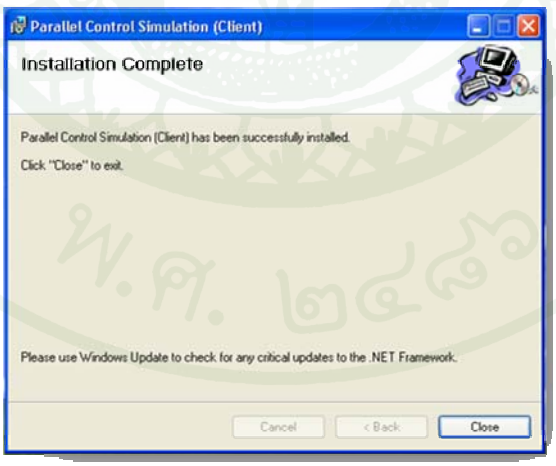
ภาพผนวกที่ ก10 แสดงการกำหนดสิทธิ์และPath ฝั่ง Client

2.4 หลังจากกำหนดสิทธิและ path ที่จะติดตั้งเรียบร้อยแล้ว ให้คลิกปุ่ม next เพื่อไปยังขั้นตอนถัดไป ดังรูปภาพผนวกที่ ก11 แล้ว คลิกปุ่ม next



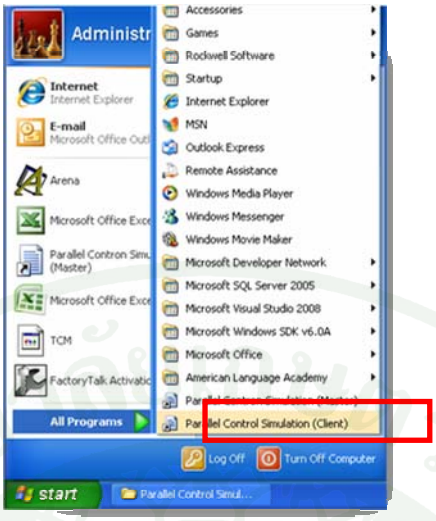
ภาพผนวกที่ ก11 แสดงการยืนยันการติดตั้งแอพพลิเคชั่นฝั่ง Client

2.5 เมื่อเครื่องคอมพิวเตอร์ได้ติดตั้งแอพพลิเคชั่นเสร็จ จะปรากฏรูปดังรูปภาพผนวกที่ ก12 ให้กดปุ่ม close เพื่อเสร็จสิ้นการติดตั้งแอพพลิเคชั่น

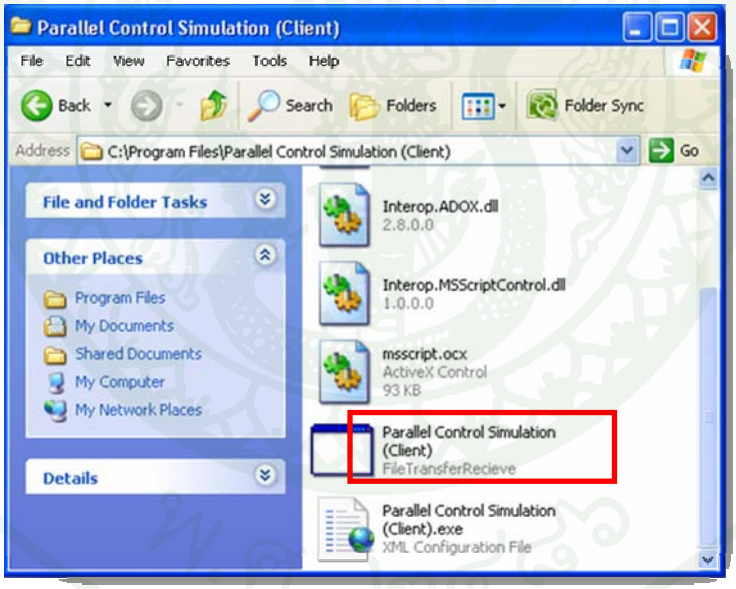


ภาพผนวกที่ ก12 เสร็จสิ้นการติดตั้งแอพพลิเคชั่นฝั่ง Client

2.6 สามารถเรียกใช้แอพพลิเคชั่นได้จาก path ที่ได้ติดตั้งไว้ โดยดับเบิ้ลคลิกที่ >>> หรือเปิดไปยังเมนู start ดังรูปภาพผนวกที่ ก13 และ ก14 ตามลำดับ



ภาพผนวกที่ ก13 เปิดแอพพลิเคชั่นผ่าน Short Cut ฝั่ง Client



ภาพผนวกที่ ก14 เปิดแอพพลิเคชั่นโดยตรงจาก Path ฝั่ง Client

ภาคผนวก ข

การใช้งานแอพพลิเคชั่น Parallel Control Simulation (Master)

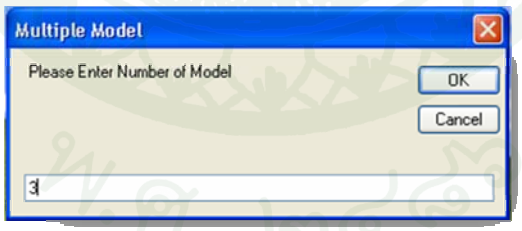
การใช้งานแอพพลิเคชั่น Parallel Control Simulation (Master)

แอพพลิเคชั่น Parallel Control Simulation (Master) ได้มีการแบ่งออกเป็น 2 ส่วน (ดังแสดงในภาพผนวกที่ ข1)

ภาพผนวกที่ ข1 Menu สำหรับการเลือกรูปแบบ

1. Parallel Control เป็นส่วนที่ออกแบบมาสำหรับผู้ใช้งานเบื้องต้นโดยจะให้แอพพลิเคชั่นเขียน Script ออกมาโดยผ่านหน้าจอการใช้งานโดยมีวิธีการใช้งานดังต่อไปนี้

1.1 หลังจากเลือกรูปแบบ Parallel Control แอพพลิเคชั่นจะทำการถามว่ามีกี่ Model ที่ต้องการประมวลผล ดังภาพผนวกที่ ข2 โดยผู้ใช้จะต้องเป็นคนใส่ค่าลงไป

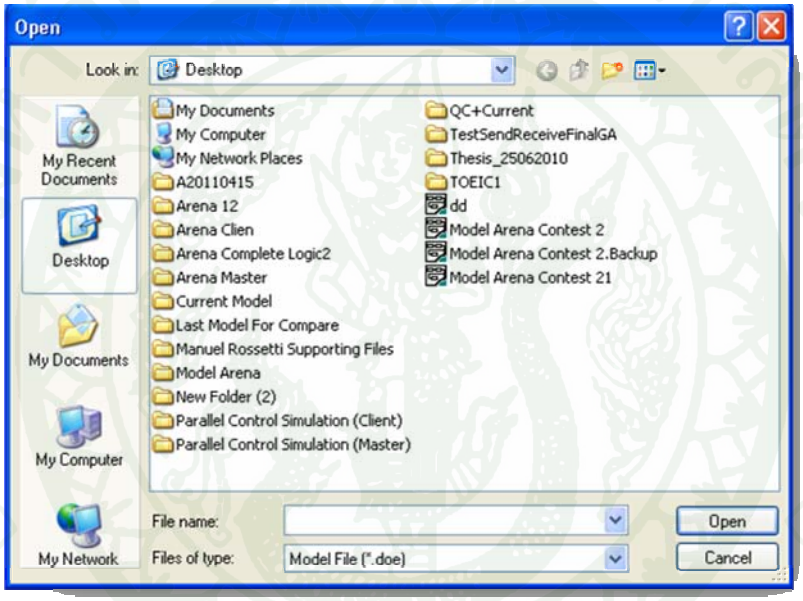


ภาพที่ ข2 กรอกจำนวน Model ที่ต้องการประมวลผล

1.2 เมื่อกรอกจำนวน Model ที่ต้องการประมวลผลแล้วจะแสดงหน้าจอดังภาพที่ ข3 ซึ่งผู้ใช้งานจะต้องเลือก Model ที่จะส่ง, Edit ค่าของ Model ที่ต้องการเปลี่ยนแปลง, กำหนดชื่อ Database ที่จะส่งกลับมายังเครื่อง Master ซึ่งจะต้องกำหนดชื่อไม่ให้ซ้ำกันเพื่อไม่ให้เกิดการซ้อนทับกันของ Database, กำหนดเครื่องที่จะส่งไปประมวลผ่านหมายเลข IP Address และกำหนดจำนวนรอบที่จะประมวลผล

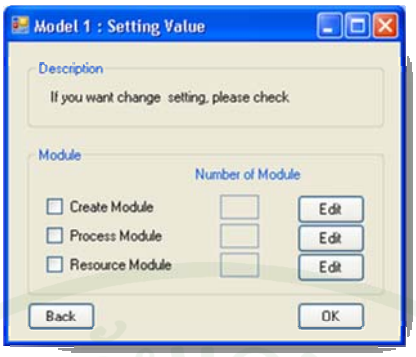
ภาพผนวกที่ ข3 กำหนดค่าต่างๆของแอพพลิเคชั่น

1.3 การเลือกไฟล์ให้กดที่ปุ่ม Browse ของภาพที่ ข3 แอพพลิเคชั่นก็จะขึ้นการเลือกไฟล์ดังภาพผนวกที่ ข4 โดยจะสามารถเลือกได้เฉพาะไฟล์สกุล Doe ซึ่งเป็นไฟล์ Model ของ Arena เท่านั้น



ภาพผนวกที่ ข4 เลือกไฟล์ Model Arena

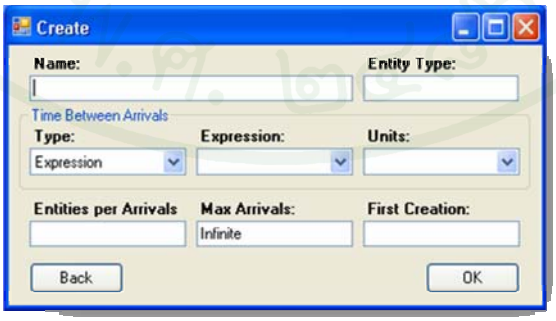
1.4 การแก้ไขค่าต่างๆ ของ Model ให้กดที่ปุ่ม edit ในภาพที่ ข3 ซึ่งแอพพลิเคชั่นกำหนดให้สามารถแก้ไขค่าใน Model ได้แค่ 3 ประเภทดังภาพที่ ข5 ซึ่งถ้าหากต้องการเปลี่ยนแปลงค่าอย่างอื่นมากกว่าที่กำหนดไว้ก็ให้ไปใช้แอพพลิเคชั่นในส่วนของ Script Control แทน



ภาพผนวกที่ ข5 เลือกประเภท Module ที่ต้องการเปลี่ยนแปลงค่า

1.5 ในการเปลี่ยนแปลงค่าต่าง ๆ ของ Module ให้ไปเลือกประเภทของ Module ที่ต้องการเปลี่ยนแปลงพร้อมกำหนดจำนวน Module โดยการที่เราจะเปลี่ยนแปลงค่าได้นั้นเราจะต้องรู้ชื่อของ Module ที่เราจะเปลี่ยนแปลงค่าเราจึงจะสามารถเปลี่ยนแปลงค่าได้

ภาพผนวกที่ ข6 เลือกประเภท Module และกำหนดจำนวน Module

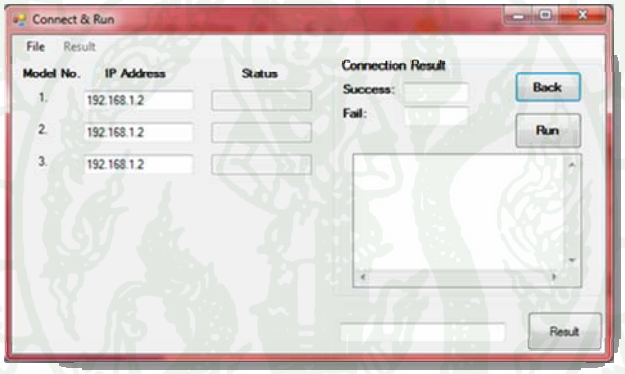


ภาพผนวกที่ ข7 ใส่ชื่อ Module ที่ต้องการเปลี่ยนแปลงค่าและแก้ไข

1.6 หลังจากกำหนดค่าต่าง ๆ ดังภาพที่ ข8 แล้วก็ให้กดปุ่ม ok เพื่อดำเนินการให้แอพพลิเคชั่นทำงานตามที่ตั้งค่าไว้

ภาพผนวกที่ ข8 กำหนดค่าต่างๆให้เรียบร้อย

1.7 หลังจากกดปุ่ม ok แล้วแอพพลิเคชั่นจะขึ้น Display ตามรูปที่ ข9 ให้กด Run เพื่อให้แอพพลิเคชั่นส่งข้อมูลไปยังเครื่อง Client ที่กำหนดไว้

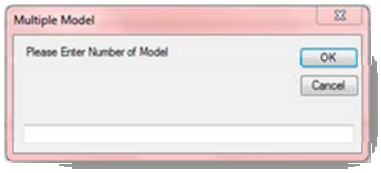


ภาพผนวกที่ ข9 แสดงส่วนการสั่งงานไปยัง Client

1.8 หลังจาก Run เสร็จเรียบร้อยแล้ว Database ที่เป็นไฟล์สกุล .mdb จะถูกเก็บไว้ใน C:\dbarena\ ซึ่งถ้าจะดูข้อมูล ต้องใช้ Microsoft Access ในการเปิดไฟล์

2. Script Control เป็นส่วนที่ออกแบบมาสำหรับผู้ที่ต้องการเขียน Script เองโดยตรงผ่านทาง text ไฟล์ซึ่งมีวิธีการใช้งานดังนี้

2.1 เมื่อเลือกรูปแบบ Script Control แอพพลิเคชั่นจะถามว่ามีกี่ Model ดังภาพผนวกที่ ข10

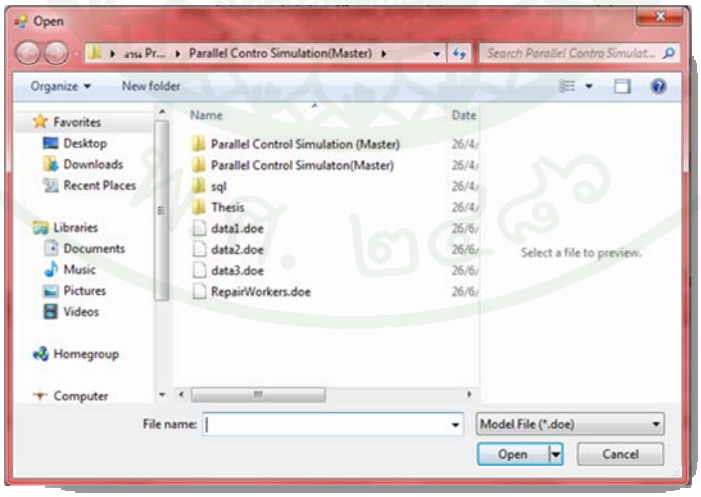


ภาพผนวกที่ ข10 กรอกจำนวน Model ที่ต้องการประมวลผล

2.2 เมื่อกรอก Model ที่ต้องการประมวลผลแล้วจะแสดงดังภาพที่ ข11 ทางผู้ใช้จะต้องเลือกไฟล์ที่จะส่งไปยัง Client , ตั้งชื่อไฟล์ที่จะทำการ save เมื่อไปถึงยังเครื่อง Client และระบุเครื่องที่จะส่งไป Run ผ่าน IP Address

ภาพผนวกที่ ข11 กรอกข้อมูลและรายละเอียด

2.3 การเลือกไฟล์ให้กดที่ปุ่ม Browse ของภาพที่ ข11 แอพพลิเคชั่นก็จะขึ้นการเลือกไฟล์ดังภาพผนวกที่ ข12 โดยจะสามารถเลือกได้เฉพาะไฟล์สกุล Doe ซึ่งเป็นไฟล์ Model ของ Arena เท่านั้น

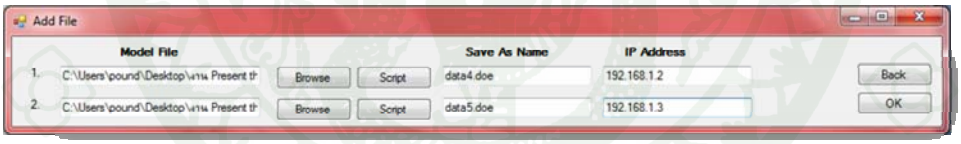


ภาพผนวกที่ ข12 เลือก Model ที่จะส่งไปประมวลผล

2.4 การเขียน Script หรือการเลือก Script ให้กดปุ่ม Script จากภาพผนวกที่ ข11 แอพพลิเคชั่นก็จะให้เขียน Script ดังภาพผนวกที่ ข13 หรือถ้าหากมีการเขียน Script ไว้แล้วก็สามารถกดปุ่ม Browse จากภาพที่ ข13 เพื่อเลือก text ไฟล์

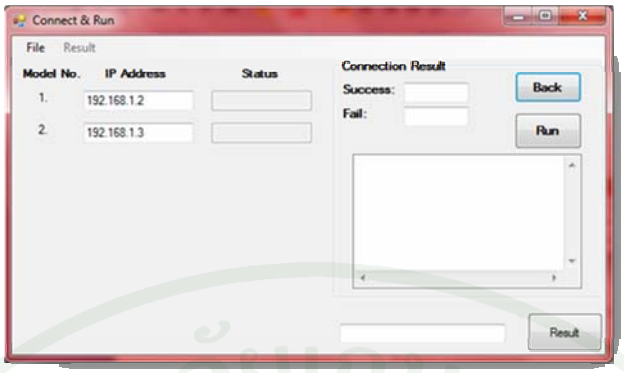
ภาพผนวกที่ ข13 เขียน Script หรือเลือก Script

2.5 หลังจากที่กรอกข้อมูลต่าง ๆ ครบแล้วดังภาพผนวกที่ ข14 ก็ให้กดปุ่ม ok เพื่อเตรียมพร้อมสำหรับการรัน



ภาพผนวกที่ ข14 กำหนดค่า

2.6 หลังจากกดปุ่ม ok แล้วแอพพลิเคชั่นจะขึ้น Display ตามภาพผนวกที่ ข15 ให้ทำการกด Run เพื่อให้แอพพลิเคชั่นส่งข้อมูลไปยังเครื่อง Client ที่กำหนดไว้



ภาพผนวกที่ ข15 แสดงส่วนส่งข้อมูลไปยัง Client

2.7 หลังจาก Run เสร็จเรียบร้อยแล้ว Database ที่เป็นไฟล์สกุล .mdb จะถูกเก็บไว้ใน C:\dbarena\ ซึ่งถ้าจะดูข้อมูลต้องใช้ Microsoft Access ในการเปิดไฟล์

ภาคผนวก ค

ฟังชันก์การใช้งานของไลบรารีและการประยุกต์ไลบรารีใช้งานกับโปรแกรมอัลกอริทึมอื่นๆ

ฟังชันก์การใช้งานของไลบรารี

ในการใช้งานไลบรารีนั้นสามารถที่จะแบ่งออกเป็น 2 คลาสที่จำเป็นต้องนำมาใช้งานคือ

1. Class TransferS โดยมีฟังชันก์ดังตารางผนวกที่ ค1

ตารางผนวกที่ ค1 ตารางแสดงฟังก์การใช้งาน Class TransferS

Class TransferS	
ฟังก์ชัน	**ความหมาย**
Sub IniVar()	กำหนดค่าตัวแปรตั้งต้น
Sub Conn(ByVal ClntIP As String)	สำหรับสร้าง connection โดย ClntIP คือ IPAddress ของเครื่องที่ตั้งการติดต่อ
Sub SendFile(ByVal strFilePath As String, ByVal strFileName As String, ByVal OutPutFile As String, ByVal StrCmd As String)	ใช้สำหรับส่ง file ไปยังเครื่องปลายทาง โดยมี parameter ดังนี้ ● strFilePath = ชื่อ file พร้อม path ● strFileName = ชื่อ file ● OutPutFile = ชื่อ file ที่ต้องการให้ส่งกลับ ● StrCmd = คำสั่งที่ใช้แยกประเภทของ file มี 2 คำสั่งด้วยกัน คือ FLE# ใช้สำหรับ ส่ง file ธรรมดา และ RUN# ใช้สำหรับส่ง Script file และ run script
Sub WriteLog(ByVal msg As String)	ใช้สำหรับเขียน log ของโปรแกรม โดย msg คือ ข้อความที่ต้องการเขียนใน log file
Sub disconnect()	ใช้สำหรับตัด connection

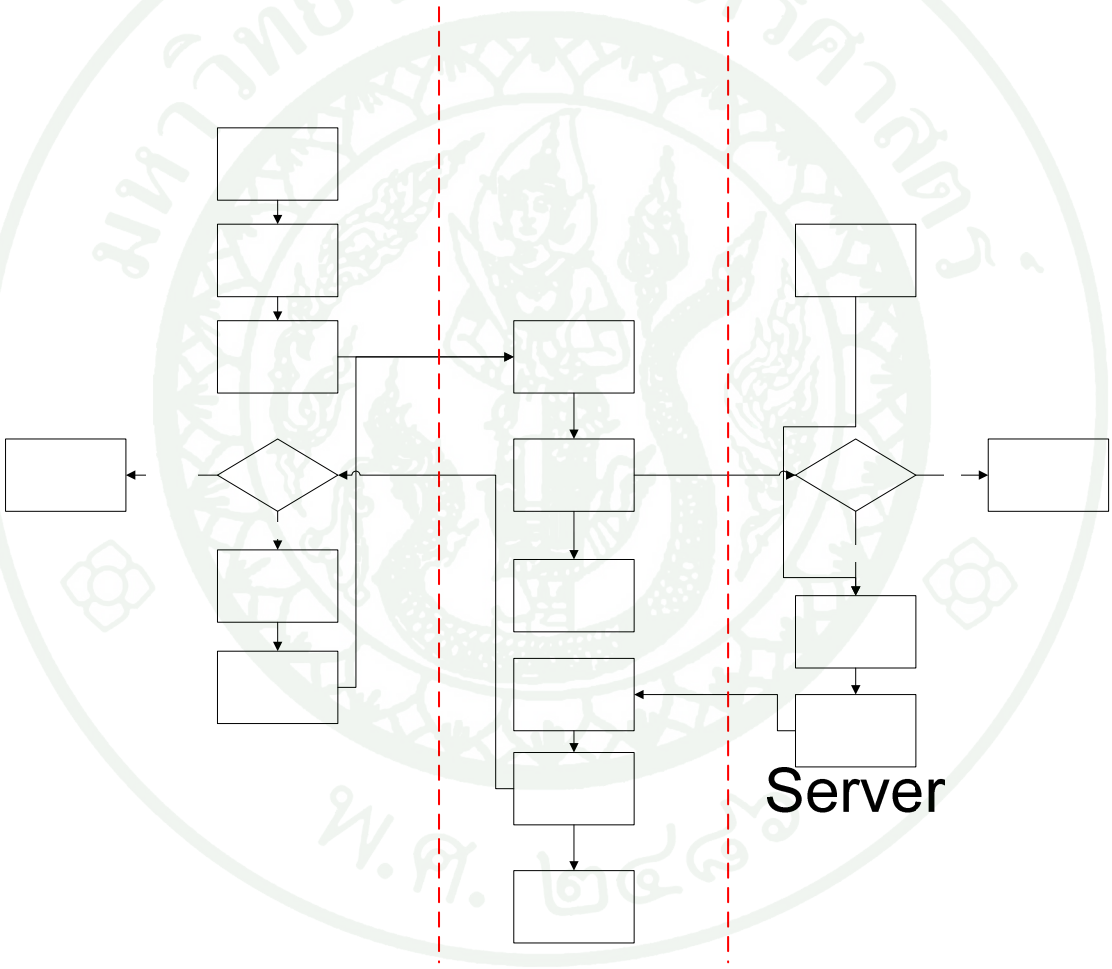
2. Class TransferR โดยมีฟังชั่นก์ดังตารางผนวกที่ ค2

ตารางผนวกที่ ค2 ตารางแสดงฟังก์การใช้งาน Class TransferS

Class TransferR	
ฟังก์ชัน	**ความหมาย**
Sub InitailVar(ByVal VAppPath As String)	ใช้กำหนดค่าตัวแปรตั้งต้น และเปิด port ให้ server ติดต่อ โดยมี VappPath (application path) เป็น parameter
Sub TransferFinishedHandler(ByVal sender As FileTransferReceive, ByVal file As String, ByVal Ipaddr As System.Net.IPAddress, ByVal OutFile As String)	ใช้สำหรับบ่งบอกว่าได้รับ file เรียบร้อยแล้ว
Sub TransferScriptFinishedHandler(ByVal sender As FileTransferReceive, ByVal file As String, ByVal Ipaddr As System.Net.IPAddress, ByVal OutFile As String)	ใช้สำหรับบ่งบอกว่าได้รับ Script file แล้ว
Sub WriteLog(ByVal msg As String)	ใช้สำหรับเขียน log ของโปรแกรม โดย msg คือ ข้อความที่ต้องการเขียนใน log file

การประยุกต์ไลบรารีใช้งานกับโปรแกรมอัลกอริทึมอื่นๆ

ในการประยุกต์ใช้กับอัลกอริทึม ผู้วิจัยได้ใช้ไลบรารีทำหน้าที่ในส่วนของการรับ-ส่งข้อมูลระหว่าง Client และ Server เพื่อให้สามารถประมวลผลแบบขนานได้จากคำสั่งของ VB Script ที่สร้างขึ้นจากฝั่ง Server นอกจากนั้นแล้วโปรแกรมอัลกอริทึมที่จะนำมาใช้งานได้นั้นจำเป็นจะต้องมีข้อมูลใน registry key ผ่านการ register ก่อน จึงจะสามารถเปลี่ยนแปลงค่าตัวแปรต่างๆ ของโปรแกรมอัลกอริทึมผ่าน VB Script ที่สร้างจากเครื่อง Server ได้



ภาพผนวกที่ ค1 ภาพแสดงวิธีการประยุกต์ใช้ไลบรารี

ภาพผนวกที่ ค1 เป็นการประยุกต์ใช้ไลบรารีให้สามารถประมวลผลแบบขนานได้โดยมีขั้นตอนดังต่อไปนี้

1. Set ค่าตั้งต้นในฝั่ง Server โดยใช้ฟังชันก์ IniVar() สำหรับเตรียมติดต่อ Client

2. เปิดเส้นทางรับข้อมูลทางฝั่ง Server และ Client โดยใช้ฟังชันก์ InitialVar()

3. ฝั่ง Server สร้าง Script ตั้งต้นสำหรับส่งไปทางฝั่ง Client

4. ฝั่ง Server ใช้ฟังชันก์ Conn() สำหรับเชื่อมต่อเส้นทางไปยัง Client

5. ฝั่ง Server ใช้ฟังชันก์ SendFile() สำหรับส่งไฟล์ Script หรือไฟล์ Model ไปยัง Client

6. ฝั่ง Server ใช้ฟังชันก์ Disconnect() สำหรับตัดเส้นทางเชื่อมต่อกับ Client

7. ฝั่ง Client สร้าง Logic ตรวจสอบว่าไฟล์ที่เข้ามาเป็นไฟล์ชนิดใดถ้าเป็นไฟล์ Script ก็จะดึงฟังชันก์ TransferScriptFinishedHandler() มาใช้ ถ้าเป็นไฟล์อื่น ๆ ก็จะดึงฟังชันก์ Transfer Finished Handler() มาใช้

8. ฝั่ง Client ใช้ฟังชันก์ IniVar() และ Set ค่าตั้งต้นสำหรับเชื่อมต่อไปยัง Server

9. ฝั่ง Client ใช้ฟังชันก์ Conn() เชื่อมต่อกลับไปยังฝั่ง Server

10. ฝั่ง Client ใช้ฟังชันก์ SendFile() ส่ง Output กลับไปยังฝั่ง Server

11. ฝั่ง Client ใช้ฟังชันก์ Disconnect() สำหรับตัดเส้นทางเชื่อมต่อกับ Server

12. ฝั่ง Server ตรวจสอบคำตอบจาก Output ที่ส่งมาจาก Client ตาม Logic ที่กำหนดถ้าตรงตามเงื่อนไข Server จบการทำงาน ไม่ตรงไปข้อที่ 13 ต่อ

13. ฝั่ง Server ใช้ฟังชันก์ IniVar() และ Set ค่าตั้งต้นสำหรับเชื่อมต่อไปยัง Client

14. ฝั่ง Server สร้าง Script โดยเปลี่ยนแปลงพารามิเตอร์ตามเงื่อนไข Logic ที่ตั้งไว้

15. กลับไปที่ข้อ 4 ใหม่

ภาคผนวก ง

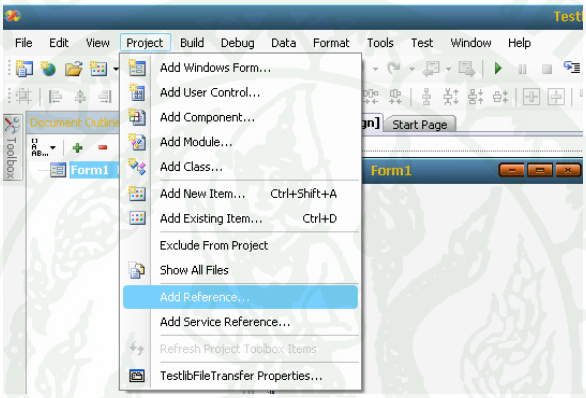
วิธีการเรียกใช้ไลบรารีใน Visual Basic Studio Version 2008 Enterprise

วิธีการใช้เรียกใช้ไลบรารีใน Visual Basic Studio Version 2008 Enterprise

1. Copy ไฟล์ไลบรารีไปยัง project ที่ต้องการเรียกใช้งาน
2. เปิด Project
3. Add Reference

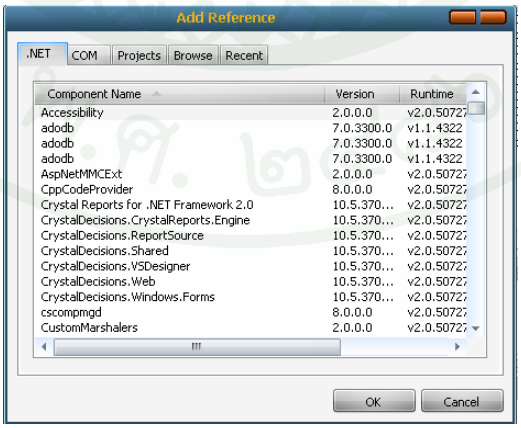
วิธีที่ 1

a. ไปที่หน้าแถบเมนู “Project” ดังภาพผนวกที่ ง1



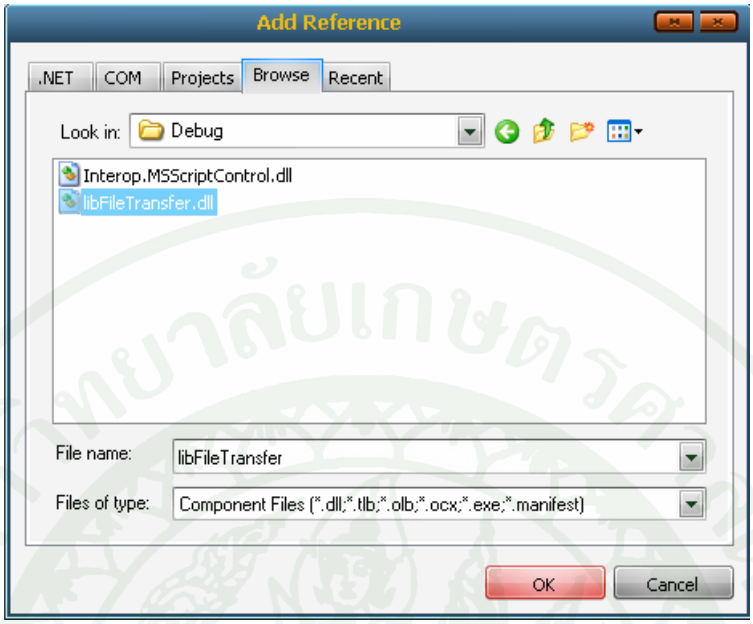
ภาพผนวกที่ ง1 Menu “Project”

b. เลือกเมนู “Add Reference …” จะปรากฏหน้าต่าง ดังภาพผนวกที่ ง2



ภาพผนวกที่ ง2 หน้าต่าง “Add reference”

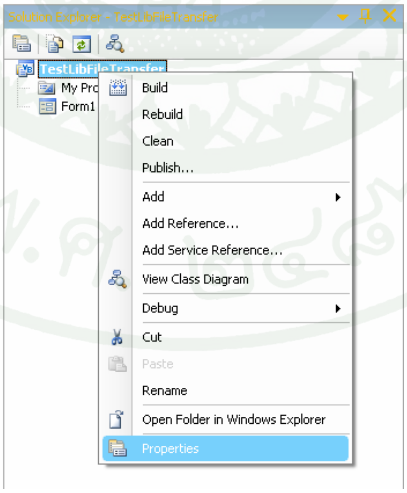
c. เลือก tab “Browse” เพื่อ Browse ไฟล์ library ที่ต้องการใช้งานดังภาพผนวกที่ ง3



ภาพผนวกที่ ง3 การเลือก library ที่ต้องการใช้งาน

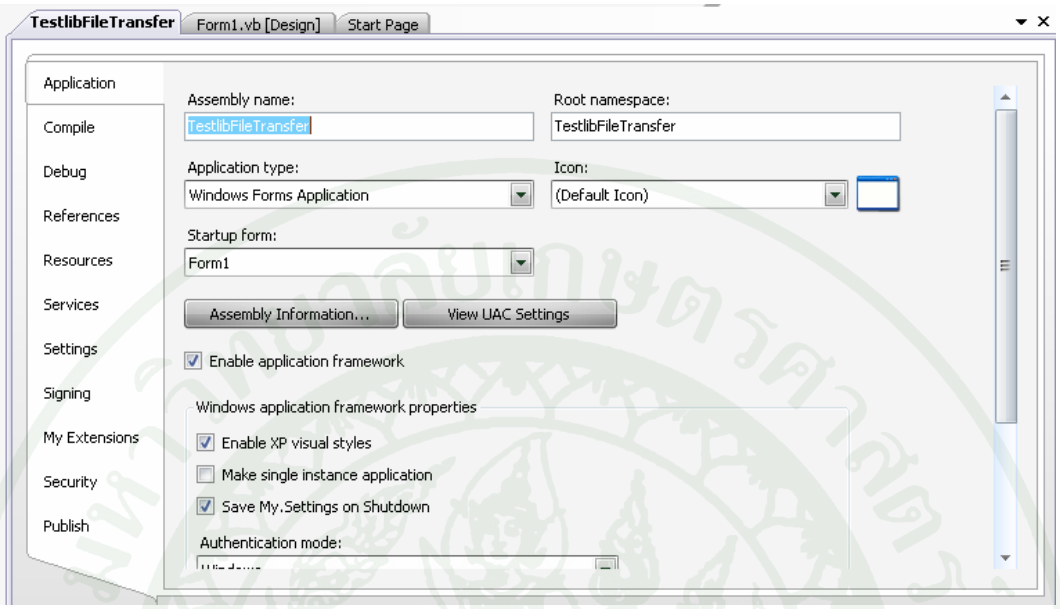
วิธีที่ 2

a. ไปที่หน้าต่าง Solution Explorer คลิกขวา ดังภาพผนวกที่ ง4



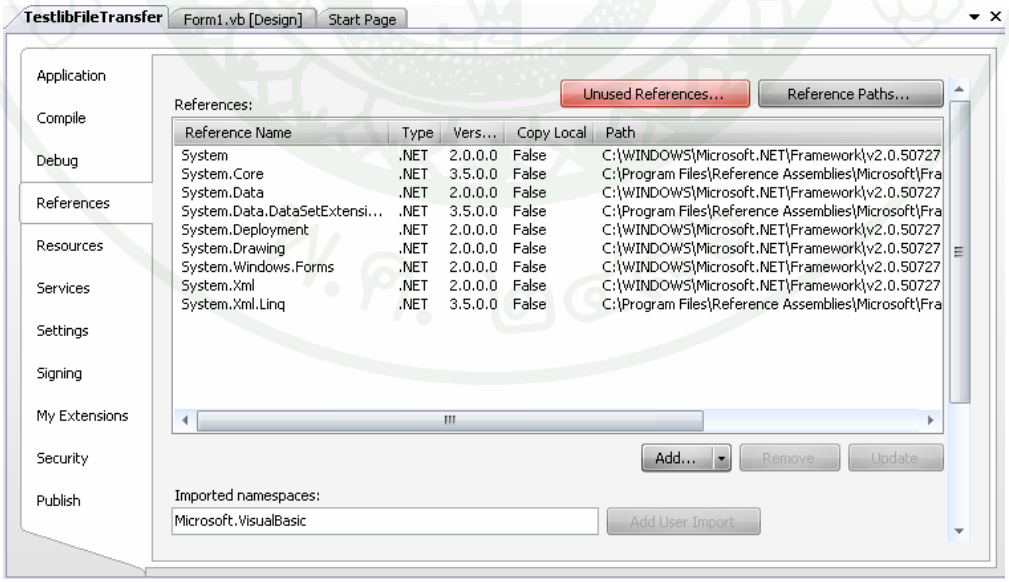
ภาพผนวกที่ ง4 Solution Explorers

b. เลือก properties จะปรากฏหน้าต่าง properties ดังภาพผนวกที่ ง5



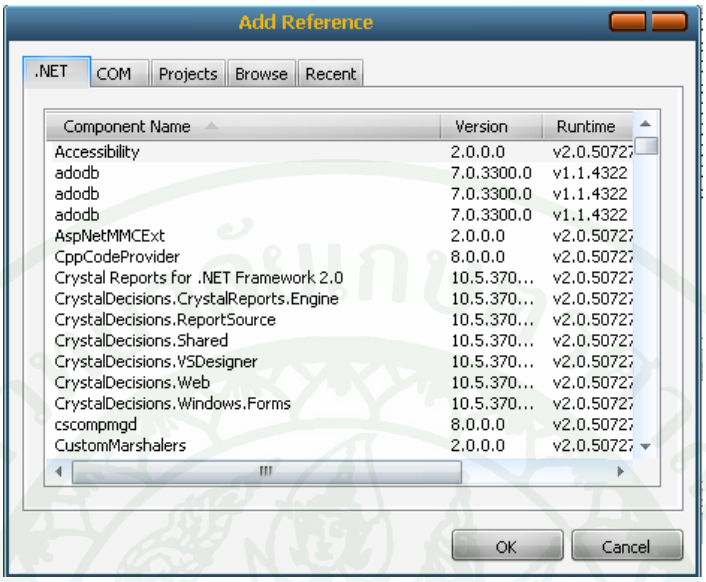
ภาพผนวกที่ ง5 หน้าต่าง properties

c. เลือก tab “Reference” จะปรากฏรายละเอียด ดังภาพผนวกที่ ง6



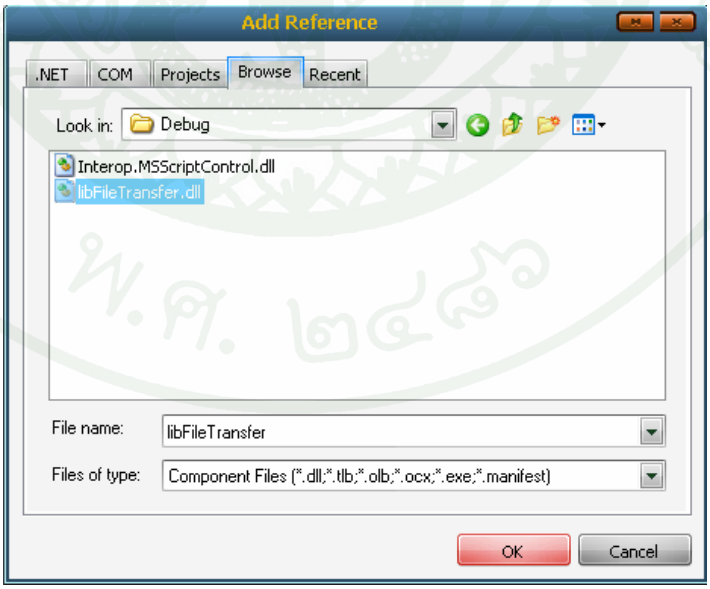
ภาพผนวกที่ ง6 รายละเอียด Reference

d. กดปุ่ม “Add” เพื่อ add library จะปรากฏหน้าจอ ดังภาพผนวกที่ ง7



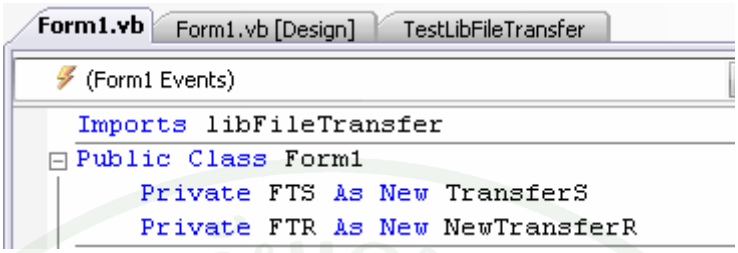
ภาพผนวกที่ ง7 หน้าต่าง “Add reference”

e. เลือก tab “Browse” เพื่อ Browse ไฟล์ library ที่ต้องการใช้งานดังภาพผนวกที่ ง8



ภาพผนวกที่ ง8 การเลือก library ที่ต้องการใช้งาน

2. Import Library ที่ต้องการใช้งาน ด้วยคำสั่ง imports “ชื่อ library”



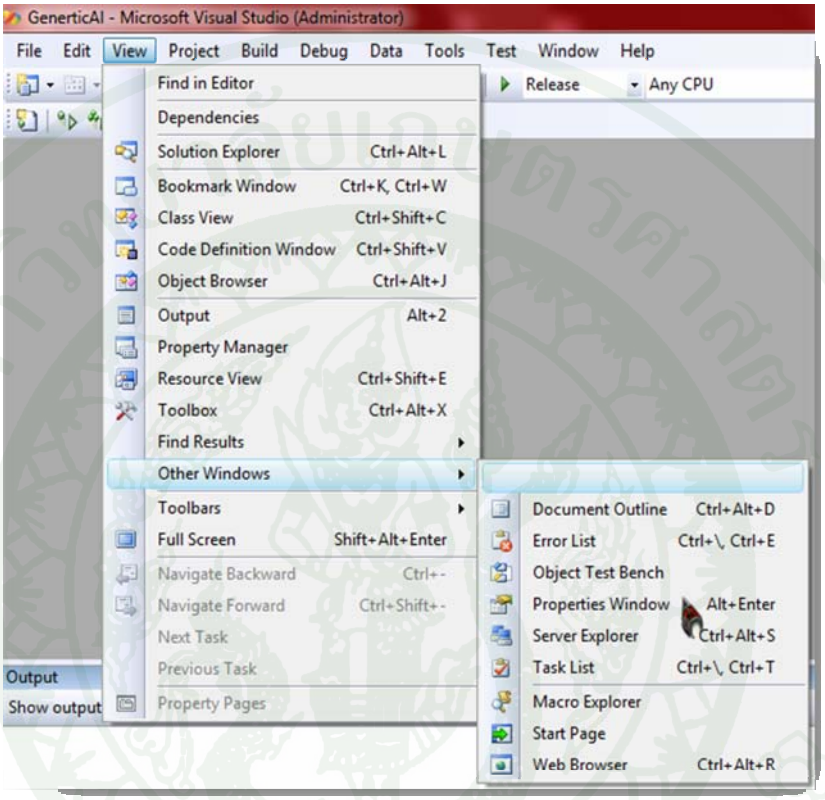
ภาพผนวกที่ ง9 การ Import Library

ภาคผนวก จ

ตัวอย่างวิธีการ register โปรแกรมโดยใช้ Microsoft Visual Studio 2008

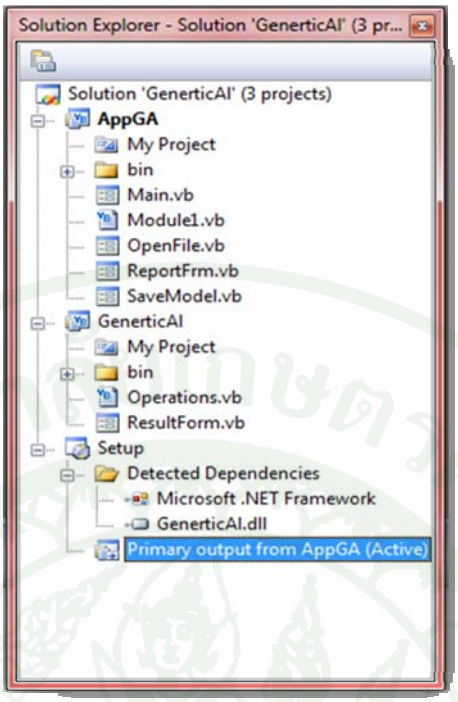
ตัวอย่างวิธีการ register โดยใช้ Microsoft Visual Studio 2008

1. เปิดโปรแกรมที่ต้องการ register โดย Microsoft Visual Studio 2008
2. เข้าไปเลือก Properties Window ดังภาพที่ จ1

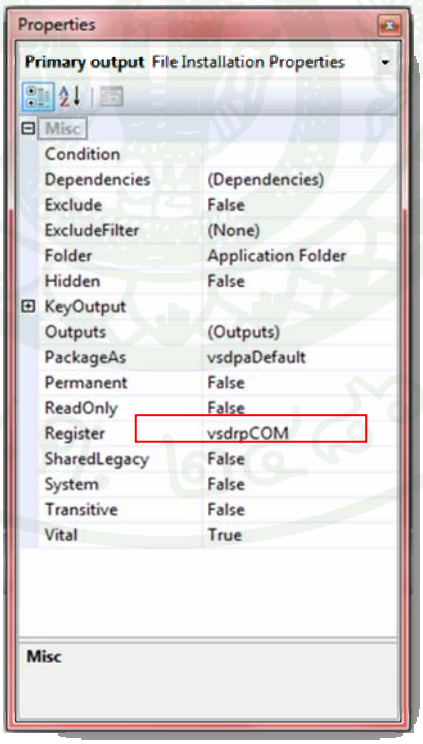


ภาพผนวกที่ จ1 Properties Window

3. เข้าไปเลือกส่วน Primary output ที่แสดงในภาพที่ จ2 แล้วปรับ Properties Window ในส่วนของ Register ให้ vsdraCom ดังแสดงในภาพผนวกที่ จ3 เพื่อทำให้ไฟล์ Setup สามารถ Register ได้

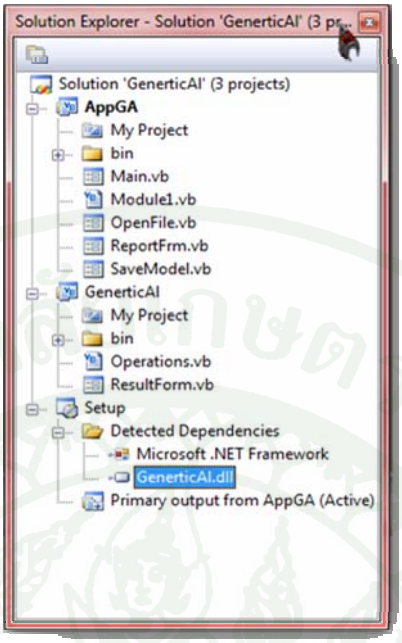


ภาพผนวกที่ จ2 เลือกส่วน Primary output

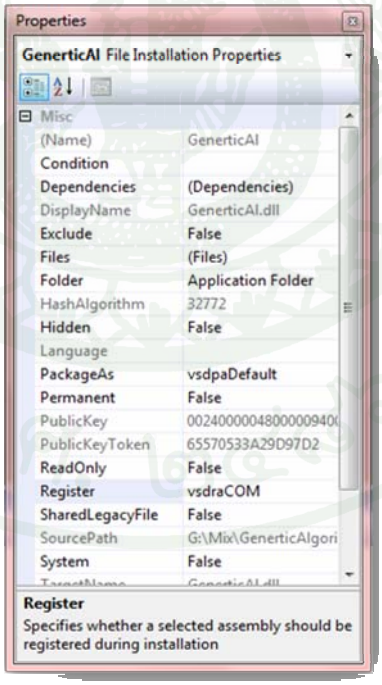


ภาพผนวกที่ จ3 ปรับสถานะของการ Register

3. ในกรณีที่มีไลบรารีให้ทำการ Register ด้วยดังแสดงในภาพที่ จ4 และ จ5 ตามลำดับ



ภาพผนวกที่ จ4 เลือกส่วนไลบรารี



ภาพผนวกที่ จ5 ปรับสถานะของการ Register

ภาคผนวก ฉ

Code โปรแกรมที่ดึงไลบรารีมาใช้ในการทดลอง

Code โปรแกรมที่ดึงไลบรารีมาใช้ในการทดลอง

1. Code ที่ใช้ไลบรารีประยุกต์ในส่วนของ Arena

1.1 Code ในฝั่ง Server

ภาพผนวกที่ ฉ1 หน้าต่างฝั่ง Server ของโปรแกรมควบคุม Arena ที่นำไลบรารีมาใช้

```
Imports libFileTransfer
Imports System.Threading
Imports System.IO
Imports Microsoft.VisualBasic.FileIO

Public Class Form1
Public CountNumb As Integer
Public ResultFlag As String
Public FTS As NewTransferS
Public FTR As NewTransferR
Public FinalResult As String
Private ClientNumb As Integer
```

```
Private ASClient() As String
Private StrFilePath As String
Private StrFileName As String
Private ModelFilePath As String
Private ModelFIleName As String

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        FTS = New NewTransferS
        FTR = New NewTransferR
        FTS.VLogFilePath = "C:\LogFileTransfer"
        FTS.VLogFileName = "C:\LogFileTransfer\SendLogS.txt"
        FTS.IniVar()
        FTR.VLogFilePath = "C:\LogFileTransfer"
        FTR.VLogFileName = "C:\LogFileTransfer\RecieveLogS.txt"
        FTR.InitailVar(Application.StartupPath)
        ModelFilePath = "C:\RepairWorkers.doe"
        ModelFIleName = "RepairWorkers.doe"
        ClientNumb = 0
        CountNumb = 0
        ReDim ASClient(ClientNumb)
        ResultFlag = "N"
        ASClient(0) = "192.168.1.104"
        StrFilePath = "D:\thesis\Mix\TestLib\Model"
        StrFileName = "D:\thesis\Mix\TestLib\Model\RepairWorkers.doe"
End Sub
```

```
Private Sub BtnConn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnConn.Click
        Try
            For i = 0 To ClientNumb
                FTS.WriteLog("Test connecting to Client with" & ASClient(i))
                FTS.Conn(ASClient(i))
                Thread.Sleep(1000)
                FTS.disconnect()
            Next
        Catch ex As Exception
            FTS.WriteLog("Fail to test conection")
        End Try
End Sub

Private Sub BtnCSend_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnCSend.Click
        Dim ScriptFilePath() As String
        Dim ScriptFileName() As String
        Dim OutfilePath() As String
        ReDim ScriptFilePath(ClientNumb)
        ReDim ScriptFileName(ClientNumb)
        ReDim OutfilePath(ClientNumb)
        For i = 0 To ClientNumb
            ScriptFilePath(i) = "c:\ScriptFile_" & i & ".txt"
            ScriptFileName(i) = System.IO.Path.GetFileName(ScriptFilePath(i))
            OutfilePath(i) = "c:\Result_" & i & ".txt"
            WriteScript(ScriptFilePath(i), "RepairWorkers.doe", 1+ CountNumb)
            CountNumb = CountNumb + 1
         Next
```

```
For i = 0 To ClientNumb
        Try
                FTS.WriteLog("Connect to Client with" & ASClient(i))
                FTS.Conn(ASClient(i))
                Thread.Sleep(1000)
        Catch ex As Exception
                FTS.WriteLog("Fail for conection")
                Exit Sub
        End Try
        Try
                FTS.WriteLog("Send Model File for" & ASClient(i))
                FTS.SendFile(ModelFilePath, ModelFIleName, "", "FLE#")
                Thread.Sleep(3000)
        Catch ex As Exception
                FTS.WriteLog("Fail to send model")
                FTS.disconnect()
        Exit Sub
        End Try
        Try
                FTS.WriteLog("Send Script File for" & ASClient(i))
                FTS.SendFile(ScriptFilePath(i), ScriptFileName(i), OutfilePath(i),
                "RUN#")
                Thread.Sleep(500)
        Catch ex As Exception
                FTS.WriteLog("Fail to Send Script File")
                FTS.disconnect()
        Exit Sub
        End Try
                FTS.disconnect()
Next
```

```
End Sub

Public Sub WriteScript(ByVal FileScript As String, ByVal ModlName As String, ByVal Resc As
Integer)
    Dim SwScript As StreamWriter = File.CreateText(FileScript)
    SwScript.WriteLine("Dim xPath, Fname, MArena")
     Try
        SwScript.WriteLine("Sub Main")
        SwScript.WriteLine("   Fname = """ & ModlName & """")
        SwScript.WriteLine("   xPath = ""C:\OutFileDir\""")
        SwScript.WriteLine("   Set AppArena = CreateObject(""Arena.application"")")
        SwScript.WriteLine("   With AppArena")
        SwScript.WriteLine("   .models.Open(xPath & Fname)")
        SwScript.WriteLine("   .Activemodel.DisableRandomness = ""False""")
        SwScript.WriteLine("   .Activemodel.DisplayDefaultReport = True")
        SwScript.WriteLine("    Resource0 =
        .Activemodel.Modules.FindByData(""Name"", ""RepairWorkers"")")
        SwScript.WriteLine("   .Activemodel.Shapes.Item(Resource0).Selected = True")
        SwScript.WriteLine("
        .Activemodel.ActiveView.Selection.Application.Activemodel.Modules.Item(Res
        ource0).Data(""Capacity"") = """ & Resc & """")
        SwScript.WriteLine("     AppArena.Activemodel.go")
        SwScript.WriteLine("     Do Until AppArena.Activemodel.End = True")
        SwScript.WriteLine("       AppArena.Activemodel.SaveAs (xPath & Fname)")
        SwScript.WriteLine("       AppArena.Quit")
        SwScript.WriteLine("       Exit Do")
        SwScript.WriteLine("     Loop")
        SwScript.WriteLine("   End With")
        SwScript.WriteLine("End Sub")
        SwScript.Close()
```

```
        Catch ex As Exception
            FTS.WriteLog("Write Script Fail!!!!")
        End Try
End Sub

Private Sub BtnResult_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnResult.Click
        FTR.WriteLog("===== The result is " & FinalResult & "
        =====")End Sub
End Class

Public Class NewTransferS
Inherits TransferS

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
        Form1.lstData.Items.Add(msg)
        Form1.LstData.SetSelected(Form1.LstData.Items.Count - 1, True)
        Form1.LstData.SetSelected(Form1.LstData.Items.Count - 1, False)
End Sub

Public Overrides Sub TransferFinishedHandler(ByVal sender As
libFileTransfer.FileTransferSend, ByVal file As String)
        MyBase.WriteLog("Transfer Finish")
End Sub
End Class
```

```
Public Class NewTransferR

Inherits TransferR


Public Overrides Sub TransferFinishedHandler(ByVal sender As FileTransferReceive, ByVal file
As String, ByVal IPaddr As System.Net.IPAddress, ByVal OutFile As String)
        Dim FilePathName As String
        Dim FileName As String
        If System.IO.File.Exists(OutFile) Then
            System.IO.File.Delete(OutFile)
        End If
        If System.IO.Directory.Exists(Path.GetDirectoryName(OutFile)) = False Then
            System.IO.Directory.CreateDirectory(Path.GetDirectoryName(OutFile))
        End If
        If Form1.ResultFlag = "N" Then
            System.IO.File.Copy(Application.StartupPath & "\" & file, OutFile)
            ReadResult(OutFile)
            If  Form1.ResultFlag = "N" Then
                Form1.FTS.WriteLog("Generate Script File to " & IPaddr.ToString)
                FilePathName = "c:\ScriptFile_" & Form1.CountNumb & ".txt"
                FileName = "ScriptFile_" & Form1.CountNumb & ".txt"
                Form1.WriteScript(FilePathName, "RepairWorkers.doe", 1 +
                Form1.CountNumb)
                Form1.CountNumb = Form1.CountNumb + 1
                Try
                    Form1.FTS.WriteLog("Connect to Client with" &
                    IPaddr.ToString)
                    Form1.FTS.Conn(IPaddr.ToString)
                    Thread.Sleep(500)
                Catch ex As Exception
                    Form1.FTS.WriteLog("Fail for conection")
```

```
                End Try
                Try
                    Form1.FTS.WriteLog("Send Script File for" &
                    IPaddr.ToString)
                    Form1.FTS.SendFile(FilePathName, FileName, OutFile,
                    "RUN#")
                    Thread.Sleep(500)
                    Form1.FTS.disconnect()
                Catch ex As Exception
                    Form1.FTS.WriteLog("Fail to Send Script File")
                End Try
            End If
        End If
        MyBase.TransferFinishedHandler(sender, file, IPaddr, OutFile)
End Sub

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
        Form1.lstData.Items.Add(msg)
        Form1.lstData.SetSelected(Form1.lstData.Items.Count - 1, True)
        Form1.lstData.SetSelected(Form1.lstData.Items.Count - 1, False)
End Sub

Private Sub ReadResult(ByVal Outfile As String)
        Dim Result(2) As String
        Try
            If File.Exists(Outfile) Then
                Using MyReader As New TextFieldParser(Outfile)
                    MyReader.TextFieldType = FileIO.FieldType.Delimited
                    MyReader.SetDelimiters("|")
```

```
                    Dim currentRow As String()
                    Dim i = 0
                    While Not MyReader.EndOfData
                        Try
                            currentRow = MyReader.ReadFields()
                            For Each currentField In currentRow
                                Result(i) = currentField
                                i = i + 1
                            Next
                        Catch ex As MalformedLineException
                            MsgBox("Line " & ex.Message & "is not valid
                            and will be skipped.")
                        End Try
                    End While
                    MyReader.Close()
                End Using
            Else
                WriteLog("Result File does not exit")
            End If
        Catch ex As Exception
            MsgBox("Error:  " & ex.Message & "from Read Result File")
            Exit Sub
        End Try
        If (Result(0) >= 0.4 And Result(0) <= 0.6) Then
            Form1.ResultFlag = "Y"
            WriteLog("===== The result is " & Result(2) & " =====")
            Form1.FinalResult = Result(0)
        End If
End Sub
```

End Class

1.2 Code ฝั่ง Client

ภาพผนวกที่ ฉ2 หน้าต่างฝั่ง Client ของโปรแกรมควบคุมที่นำไลบรารีมาใช้

```
Imports libFileTransfer
Imports System.IO
Imports System.Threading

Public Class Form1
Private FTR As NewTransferR

Public Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        If System.IO.Directory.Exists("C:\OutFileDir") = False Then
                System.IO.Directory.CreateDirectory("c:\OutfileDir")
        End If
        FTR = New NewTransferR
        FTR.VLogFilePath = "C:\LogFileTransfer"
        FTR.VLogFileName = "C:\LogFileTransfer\RecieveLogC.txt"
        FTR.InitailVar(Application.StartupPath)
```

```
End Sub
```

End Class

Public Class NewTransferR

```
Inherits TransferR
Public FTS As NewTransferS
Public ModelPathName As String
Public ModelName As String
Public Model As String

Private Sub RetrieveR(ByVal DBName As String, ByVal Outfile As String)
        Dim conStr As String
        conStr = "provider=Microsoft.jet.Oledb.4.0;" & _
        "Data Source=C:\OutFileDir\" & DBName ‘เป็นคำสั่งในการติดต่อ access
        Dim conn As New OleDb.OleDbConnection(conStr)
        conn.Open() ‘เป็นการรันคำสั่ง
        Dim sql As String
        sql = " SELECT AVG(AvgObs) AS Average" & _
          " FROM(StatsAndOutputQry) " & _
          " WHERE SourceCategory.Name = 'Resource' " & _
          " AND SourceProcess.Name = 'RepairWorkers' " & _
          " AND SourceDataType.Name not in ('Scheduled Utilization', 'Total Number Seized') "
        & _
          " GROUP BY  SourceCategory.Name, SourceProcess.Name, SourceDataType.Name "
        Dim cmd As New OleDb.OleDbCommand(sql, conn)
        Dim adapter As New OleDb.OleDbDataAdapter(cmd)
        Dim data As New DataSet
        adapter.Fill(data, "AVG")
        Dim AVG(2) As String
        For i = 0 To data.Tables("AVG").Rows.Count - 1
```

```
                AVG(i) = data.Tables("AVG").Rows(i)("Average")
        Next
        conn.Close()
        If File.Exists(Outfile) Then
                File.Delete(Outfile)
        End If
        Dim swOutfile As StreamWriter = File.CreateText(Outfile)
        For i = 0 To 2
                swOutfile.WriteLine(AVG(i))
        Next
        swOutfile.Close()
End Sub

Public Overrides Sub TransferScriptFinishedHandler(ByVal sender As FileTransferReceive,
ByVal file As String, ByVal IPaddr As System.Net.IPAddress, ByVal OutFile As String)
        MyBase.TransferScriptFinishedHandler(sender, file, IPaddr, OutFile)
        Dim DBName As String
        DBName = Model & ".mdb"
        RetrieveR(DBName, OutFile)
        FTS = New NewTransferS
        FTS.VLogFilePath = "C:\LogFileTransfer"
        FTS.VLogFileName = "C:\LogFileTransfer\SendLogC.txt"
        FTS.IniVar()
        Dim OutFileName As String
        OutFileName = Path.GetFileName(OutFile)
        FTS.Conn(IPaddr.ToString)
        Thread.Sleep(500)
        FTS.SendFile(OutFile, OutFileName, OutFileName, "FLE#")
        Thread.Sleep(500)
        FTS.disconnect()
```

```
End Sub

Public Overrides Sub TransferFinishedHandler(ByVal sender As FileTransferReceive, ByVal file
As String, ByVal IPaddr As System.Net.IPAddress, ByVal OutFile As String)
        ModelPathName = "C:\OutFileDir\" & file
        ModelName = Path.GetFileName(ModelPathName)
        Model = Path.GetFileNameWithoutExtension(ModelPathName)  'ตัดแต่ชื่อไฟล์
        If System.IO.File.Exists(ModelPathName) Then
            System.IO.File.Delete(ModelPathName)
        End If
        If System.IO.Directory.Exists("C:\OutFileDir") = False Then
            System.IO.Directory.CreateDirectory("c:\OutfileDir")
        End If
        System.IO.File.Copy(Application.StartupPath & "\" & file, ModelPathName)
        MyBase.TransferFinishedHandler(sender, file, IPaddr, OutFile)
End Sub

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
End Sub
End Class

Public Class NewTransferS
Inherits TransferS
Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
End Sub
End Class
```

2. Code ที่ใช้ไลบรารีประยุกต์ในส่วนของ Genatic Algorithms

2.1 Code ในฝั่ง Server

ภาพผนวกที่ ฉ3 หน้าต่างฝั่ง Server ของโปรแกรมควบคุม Genetic Algorithms ที่นำไลบรารีมาใช้

Module Module1

```
Public SendCounter As Integer
Public MaxCounter As Integer = 10
Public Result(,) As String
Public FResult() As String
Public CCounter As Integer
Public TResult(,,) As String
Public TBResult(,,) As String
Public TotalC As Integer
Public FTS As New NewTransferS
Public FTR As New NewTransferR
Public ClientNumb As Integer
Public IncNumber As Integer
Public Sub ShowMsg(ByVal msg As String)
```

```
        Main.lstData.Items.Add(msg)
        Main.lstData.SetSelected(Main.lstData.Items.Count - 1, True) 'scrolls to last nitem
        Main.lstData.SetSelected(Main.lstData.Items.Count - 1, False)
End Sub
```

End Module

```
Imports libFileTransfer
Imports System.Threading
Imports System.IO
Imports Microsoft.VisualBasic.FileIO
```

Public Class Main

```
Private ASClient() As String
Private ModelFilePath As String
Private ModelFileName As String
Public TotlLine As Integer

Private Sub Main_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        FTS.VLogFilePath = "C:\LogFileTransfer"
        FTS.VLogFileName = "C:\LogFileTransfer\SendLogS.txt"
        FTS.IniVar()
        FTR.VLogFilePath = "C:\LogFileTransfer"
        FTR.VLogFileName = "C:\LogFileTransfer\RecieveLogS.txt"
        FTR.InitailVar(Application.StartupPath)
        If System.IO.Directory.Exists("C:\OutFileDir") Then
        Else
                System.IO.Directory.CreateDirectory("c:\OutFileDir")
        End If
```

```
        ClientNumb = 2
        CCounter = 0
        ReDim ASClient(ClientNumb)
        ASClient(0) = "192.168.1.104"
        ASClient(1) = "192.168.1.105"
        ASClient(2) = "192.168.1.106"
End Sub

Private Sub Browse_Click_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        With OpenFileDialog1
            .FileName = ""
            .Filter = "Text File (*.txt)|*.txt"
         End With
        If (OpenFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK) Then
            Dim temp_name As String
            temp_name = OpenFileDialog1.FileName
            ShowMsg(temp_name)
            ModelFilePath = temp_name
            ModelFIleName = Path.GetFileName(ModelFilePath)
        End If
        TotlLine = 0
        Using MyReader As New TextFieldParser(ModelFilePath)
            Dim LineStr As String
            While Not MyReader.EndOfData
                Try
                    LineStr = MyReader.ReadLine()
                    If (LineStr.Trim().Length > 0) Then
                        TotlLine += 1
                    End If
```

```
                    Catch ex As MalformedLineException
                        MsgBox("Line " & ex.Message & "is not valid and will be
                        skipped.")
                    End Try
                End While
                MyReader.Close()
            End Using
            TotalC = TotlLine + 4
            ReDim TResult(MaxCounter, ClientNumb, TotalC)
            ReDim TBResult(MaxCounter, ClientNumb, TotalC)
            ReDim Result(MaxCounter, TotalC)
End Sub

Public Sub WriteScript(ByVal FileScript As String, ByVal ModlName As String, ByVal
IncNumb As Integer)
            Dim SwScript As StreamWriter = File.CreateText(FileScript)
            SwScript.WriteLine("Dim xPath, Fname")
            Try
                SwScript.WriteLine("Sub FindPath()")
                SwScript.WriteLine("   Dim fs, f, s, xModelPath, i, tempStr")
                SwScript.WriteLine("   Set fs = CreateObject(""Scripting.FileSystemObject"")")
                SwScript.WriteLine("   Set f = fs.GetFile(Fname)")
                SwScript.WriteLine("   xModelPath = f.ShortPath")
                SwScript.WriteLine("   i = 1")
                SwScript.WriteLine("   Do Until i = Len(xModelPath)")
                SwScript.WriteLine("     tempStr = Right(xModelPath, i)")
                SwScript.WriteLine("     If Left(tempStr, 1) = ""\"" Then")
                SwScript.WriteLine("       xPath = Mid(xModelPath, 1, Len(xModelPath) –
                Len(tempStr) + 1)")
                SwScript.WriteLine("       Exit Do")
```

```
                SwScript.WriteLine("      End If")
                SwScript.WriteLine("      i = i + 1")
                SwScript.WriteLine("   Loop")
                SwScript.WriteLine("End Sub")
                SwScript.WriteLine("")
                SwScript.WriteLine("Sub Main")
                SwScript.WriteLine("   Fname = """ & ModlName & """")
                SwScript.WriteLine("   xPath = ""C:\OutFileDir\""")
                SwScript.WriteLine("   FindPath()")
                SwScript.WriteLine("   Set AppGA =
                CreateObject(""GenerticAL.Operations"")")
                SwScript.WriteLine("   With AppGA")
                SwScript.WriteLine("   .filename = xPath & Fname")
                SwScript.WriteLine("   .open(xPath & Fname)")
                SwScript.WriteLine("   .ANode(0).idx =  ANode(0).idx  + " & IncNumb)
                SwScript.WriteLine("   .GenNumb = 1000")
                SwScript.WriteLine("   .run()")
                SwScript.WriteLine("   .report(""C:\test.txt"")")
                SwScript.WriteLine("   .AppGA.Quit")
                SwScript.WriteLine("   End With")
                SwScript.WriteLine("End Sub")
                SwScript.Close()
        Catch ex As Exception
                FTS.WriteLog("Write Script Fail!!!!")
        End Try
    End Sub

Private Sub BtnConn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnConn.Click
        Try
```

```
                For i = 0 To ClientNumb
                    FTS.WriteLog("Test connecting to Client with" & ASClient(i))
                    FTS.Conn(ASClient(i))
                    Thread.Sleep(3000)
                    FTS.disconnect()
                Next
            Catch ex As Exception
                FTS.WriteLog("Fail to test conection")
            End Try
End Sub

Private Sub BtnCSend_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnCSend.Click
            SendCounter = 0
            IncNumber = 0
            SendsCript(IncNumber)
End Sub

Public Sub SendsCript(ByVal INumber As Integer)
            Dim ScriptFilePath() As String
            Dim ScriptFileName() As String
            Dim OutfilePath() As String
            ReDim ScriptFilePath(ClientNumb)
            ReDim ScriptFileName(ClientNumb)
            ReDim OutfilePath(ClientNumb)
            For i = 0 To ClientNumb
                ScriptFilePath(i) = "c:\ScriptFile_" & i & ".txt"
                ScriptFileName(i) = System.IO.Path.GetFileName(ScriptFilePath(i))
                OutfilePath(i) = "c:\Result_" & i & ".txt"
                WriteScript(ScriptFilePath(i), ModelFIleName, INumber)
```

```
Next
For i = 0 To ClientNumb
        Try
                FTS.WriteLog("Connect to Client with" & ASClient(i))
                FTS.Conn(ASClient(i))
                Thread.Sleep(3000)
        Catch ex As Exception
                FTS.WriteLog("Fail for conection")
                Exit Sub
        End Try

        Try
                FTS.WriteLog("Send Model File for" & ASClient(i))
                FTS.SendFile(ModelFilePath, ModelFileName, "", "FLE#")
                Thread.Sleep(3000)
        Catch ex As Exception
                FTS.WriteLog("Fail to send model")
                FTS.disconnect()
                Exit Sub
        End Try
        Try
                FTS.WriteLog("Send Script File for" & ASClient(i))
                FTS.SendFile(ScriptFilePath(i), ScriptFileName(i), OutfilePath(i),
                "RUN#")
                Thread.Sleep(500)
        Catch ex As Exception
                FTS.WriteLog("Fail to Send Script File")
                FTS.disconnect()
                Exit Sub
        End Try
```

```
                FTS.disconnect()
        Next
End Sub
```

End Class

Public Class NewTransferS

```
Inherits TransferS

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
        ShowMsg(msg)
End Sub

Public Overrides Sub TransferFinishedHandler(ByVal sender As
libFileTransfer.FileTransferSend, ByVal file As String)
        MyBase.WriteLog("Transfer Finish")
End Sub
```

End Class

Public Class NewTransferR

```
Inherits TransferR
Public ClientCount As Integer = 0

Public Overrides Sub TransferFinishedHandler(ByVal sender As FileTransferReceive, ByVal file
As String, ByVal IPaddr As System.Net.IPAddress, ByVal OutFile As String)
        If System.IO.File.Exists(OutFile) Then
                System.IO.File.Delete(OutFile)
        End If
        If System.IO.Directory.Exists(Path.GetDirectoryName(OutFile)) = False Then
```

```
            System.IO.Directory.CreateDirectory(Path.GetDirectoryName(OutFile))
End If
System.IO.File.Copy(Application.StartupPath & "\" & file, OutFile)
ReadResult(OutFile)

If CCounter = ClientNumb Then
            calmin()
            If SendCounter = MaxCounter Then
                        CalcFMin()
                        FTR.WriteLog("The Result is :" & FResult(2))
                        Dim TempR As String
                        TempR = "Format is "
                        For i = 3 To TotalC
                                    TempR = TempR & "|" & FResult(i)
                        Next
                        FTR.WriteLog(TempR)
            Else
                        If Result(SendCounter, 2) < 40 Then
                                    FTR.WriteLog("The Result is :" & Result(SendCounter, 2))
                                    Dim TempR As String
                                    TempR = "Format is "
                                    For i = 3 To TotalC
                                                TempR = TempR & "|" & Result(SendCounter, i)
                                    Next
                                    FTR.WriteLog(TempR)
                        Else
                                    IncNumber = IncNumber + 1
                                    Main.SendsCript(IncNumber)
                        End If
            End If
```

```
        End If
        MyBase.TransferFinishedHandler(sender, file, IPaddr, OutFile)
        CCounter = CCounter + 1
End Sub

Sub CalcFMin()
        Dim Cbuff() As Double
        Dim Tbuff As Double
        ReDim Cbuff(MaxCounter)
        For i = 0 To MaxCounter
                Cbuff(i) = Result(i, 2)
        Next
        For i = 0 To MaxCounter
                For j = 0 To MaxCounter - 1
                        If Cbuff(j) > Cbuff(j + 1) Then
                                Tbuff = Cbuff(j)
                                Cbuff(j) = Cbuff(j + 1)
                                Cbuff(j + 1) = Tbuff
                        End If
                Next
        Next
        For i = 0 To MaxCounter
                If Result(i, 2) = Cbuff(0) Then
                        For j = 0 To TotalC
                                FResult(j) = Result(i, j)
                        Next
                        Exit For
                End If
        Next
End Sub
```

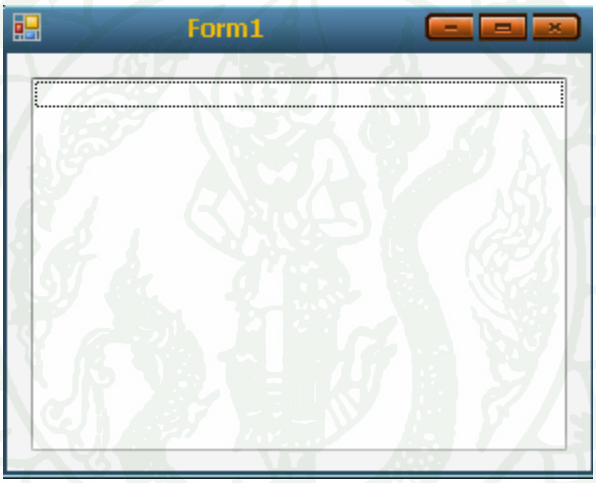
```
Sub calmin()
        Dim Tbuff As Double
        Dim Cbuff() As Double
        ReDim Cbuff(ClientNumb)
        For i = 0 To ClientNumb
                Cbuff(i) = TResult(IncNumber, i, 2)
        Next
        For i = 0 To ClientNumb
                For j = 0 To ClientNumb - 1
                        If Cbuff(j) > Cbuff(j + 1) Then
                                Tbuff = Cbuff(j)
                                Cbuff(j) = Cbuff(j + 1)
                                Cbuff(j + 1) = Tbuff
                        End If
                Next
        Next
        For i = 0 To ClientNumb
                If TResult(IncNumber, i, 2) = Cbuff(0) Then
                        For j = 0 To TotalC
                                Result(IncNumber, j) = TResult(IncNumber, i, j)
                        Next
                        Exit For
                End If
        Next
End Sub

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
        ShowMsg(msg)
End Sub
```

```
Private Sub ReadResult(ByVal Outfile As String)
    Try
        Dim Colm As Integer = 0
        Dim i As Integer
        If File.Exists(Outfile) Then
            Using MyReader As New TextFieldParser(Outfile)
                MyReader.TextFieldType = FileIO.FieldType.Delimited
                MyReader.SetDelimiters("|")
                Dim currentRow As String()
                While Not MyReader.EndOfData
                    Try
                        currentRow = MyReader.ReadFields()
                        i = 0
                        If MyReader.LineNumber = 2 Then
                            For Each currentField In currentRow
                                TResult(IncNumber,
                                CCounter, i) = currentField
                                i = i + 1
                            Next
                            Exit While
                        End If
                    Catch ex As MalformedLineException
                        MsgBox("Line " & ex.Message & "is not valid
                        and will be skipped.")
                    End Try
                End While
                MyReader.Close()
            End Using
        Else
            WriteLog("Result File does not exit")
```

```
            End If
        Catch ex As Exception
            MsgBox("Error:  " & ex.Message & "from Read Result File")
            Exit Sub
        End Try
End Sub
End Class
```

2.2 Code ในฝั่ง Client



ภาพผนวกที่ ฉ4 หน้าต่างฝั่ง Client ของโปรแกรมควบคุม Genetic Algorithms ที่นำไลบรารีมาใช้

```
Module Module1
Public Sub ShowMsg(ByVal msg As String)
Form1.LstData.Items.Add(msg)
Form1.LstData.SetSelected(Form1.LstData.Items.Count - 1, True)
Form1.LstData.SetSelected(Form1.LstData.Items.Count - 1, False)
End Sub
End Module
```

```
Imports libFileTransfer
Imports System.IO
Imports System.Threading
Imports Microsoft.VisualBasic.FileIO

Public Class Form1
Private FTR As NewTransferR

Public Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        If System.IO.Directory.Exists("C:\OutFileDir") = False Then
            System.IO.Directory.CreateDirectory("c:\OutfileDir")
        End If
        FTR = New NewTransferR
        FTR.VLogFilePath = "C:\LogFileTransfer"
        FTR.VLogFileName = "C:\LogFileTransfer\RecieveLogC.txt"
        FTR.InitailVar(Application.StartupPath)
End Sub
End Class

Public Class NewTransferR
Inherits TransferR
Public FTS As NewTransferS
Public ModelPathName As String
Public ModelName As String
Public Model As String
Private Sub RetrieveR(ByVal Outfile As String)
        Dim RTemp As String
        Dim Fcount As Integer
```

```
RTemp = "Temp_" & Outfile
Using MyReader As New TextFieldParser(Outfile)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters("|")
        Dim currentRow As String()
        Dim j = 0
        Fcount = 0
        While Not MyReader.EndOfData
            Try
                If MyReader.LineNumber = 2 Then
                    currentRow = MyReader.ReadFields()
                    Dim currentField As String
                    For Each currentField In currentRow
                        Fcount = Fcount + 1
                    Next
                    Exit While
                End If
            Catch ex As MalformedLineException
                MsgBox("Line " & ex.Message & "is not valid and will be
                skipped.")
            End Try
        End While
        MyReader.Close()
End Using
Using MyReader As New TextFieldParser(Outfile)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters("|")
        Dim currentRow As String()
        Dim j = 0
        While Not MyReader.EndOfData
```

```
                Try
                    If MyReader.LineNumber = 2 Then
                        currentRow = MyReader.ReadFields()
                        Dim i = 0
                        Dim StrTemp() As String
                        ReDim StrTemp(Fcount)
                            For Each currentField In currentRow
                                StrTemp(i) = currentField
                                i = i + 1
                            Next
                        Dim swOutfile As StreamWriter =
                        File.CreateText(RTemp)
                        For i = 0 To 2
                            swOutfile.WriteLine(StrTemp(i))
                        Next

                        swOutfile.Close()
                    End If
                Catch ex As MalformedLineException
                    MsgBox("Line " & ex.Message & "is not valid and will be
                    skipped.")
                End Try
            End While
            MyReader.Close()
        End Using
        System.IO.File.Delete(Outfile)
        System.IO.File.Copy(RTemp, Outfile)
End Sub
```

```
Public Overrides Sub TransferScriptFinishedHandler(ByVal sender As FileTransferReceive,
ByVal file As String, ByVal IPaddr As System.Net.IPAddress, ByVal OutFile As String)
        MyBase.TransferScriptFinishedHandler(sender, file, IPaddr, OutFile)
        FTS = New NewTransferS
        FTS.VLogFilePath = "C:\LogFileTransfer"
        FTS.VLogFileName = "C:\LogFileTransfer\SendLogC.txt"
        FTS.IniVar()
        Dim OutFileName As String
        OutFileName = Path.GetFileName(OutFile)
        RetrieveR(OutFile)
        FTS.Conn(IPaddr.ToString)
        Thread.Sleep(500)
        FTS.SendFile(OutFile, OutFileName, OutFileName, "FLE#")
        Thread.Sleep(500)
        FTS.disconnect()
End Sub

Public Overrides Sub TransferFinishedHandler(ByVal sender As FileTransferReceive, ByVal file
As String, ByVal IPaddr As System.Net.IPAddress, ByVal OutFile As String)
        ModelPathName = "C:\OutFileDir\" & file
        ModelName = Path.GetFileName(ModelPathName)
        Model = Path.GetFileNameWithoutExtension(ModelPathName)
        If System.IO.File.Exists(ModelPathName) Then
                System.IO.File.Delete(ModelPathName)
        End If
        If System.IO.Directory.Exists("C:\OutFileDir") = False Then
                System.IO.Directory.CreateDirectory("c:\OutfileDir")
        End If
        System.IO.File.Copy(Application.StartupPath & "\" & file, ModelPathName)
        MyBase.TransferFinishedHandler(sender, file, IPaddr, OutFile)
```

```
End Sub

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
        ShowMsg(msg)
End Sub
End Class

Public Class NewTransferS
Inherits TransferS

Public Overrides Sub WriteLog(ByVal msg As String)
        MyBase.WriteLog(msg)
        ShowMsg(msg)
End Sub
End Class
```

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายอภิศักดิ์ วิทยาประภากร
วัน เดือน ปี ที่เกิด	วันที่ 20 พฤษภาคม 2527
สถานที่เกิด	ระนอง
ประวัติการศึกษา	วศ.บ. (วิศวอุตสาหการ) มหาวิทยาลัยสงขลานครินทร์
ผลงานดีเด่นและรางวัลทางวิชาการ	รองชนะเลิศการแข่งขันโปรแกรม Arena ปี 2553