การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์

TWO-DIMENSIONAL PARALLEL FAST FOURIER TRANSFORM ON A CLUSTER OF PCs

อภินันทน์ เสริมศรี

APINAN SERMSRI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2550

การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์

TWO-DIMENSIONAL PARALLEL FAST FOURIER TRANSFORM ON A CLUSTER OF PCs

อภินันทน์ เสริมศรี

APINAN SERMSRI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2550

TWO-DIMENSIONAL PARALLEL FAST FOURIER TRANSFORM ON A CLUSTER OF PCs

APINAN SERMSRI

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2007

หัวข้อวิทยานิพนธ์	การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์
นักศึกษา	นายอภินันทน์ เสริมศรี
รหัสประจำตัว	47063708
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2550
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ. ดร. จีรพร ศรีสวัสดิ์

บทคัดย่อ

การแปลงฟูเรียร์ (Fourier Transform : FT) เป็นเทคนิคพื้นฐานที่สำคัญต่องานวิจัยและพัฒนาทางเทคโนโลยีในหลายสาขา ตัวอย่างเช่น งานทางด้านวิทยาศาสตร์ และทางด้านวิศวกรรมศาสตร์ โดยเฉพาะงานทางด้านการประมวลผลสัญญาณ เป็นต้น งานวิจัยนี้ได้นำเสนอวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน (2-D Parallel Fast Fourier Transform : 2-D PFFT) ทั้งหมด 3 วิธี คือ 1) วิธีอาร์ซีทูดีพีเอฟเอฟที (RC 2-D PFFT) ซึ่งเป็นวิธีที่ใช้ในงานวิจัยส่วนมาก เนื่องจากเป็นวิธีพัฒนาง่ายและสามารถคำนวณได้เร็วกว่าวิธีอื่น 2) วิธีวีอาร์ทูดีพีเอฟเอฟที (VR 2-D PFFT) เป็นวิธีที่การแปลงฟูเรียร์แบบเร็วที่ถูกพัฒนาขึ้นโดยมีรูปแบบการติดต่อระหว่างหน่วยประมวลผลที่ง่ายกว่าวิธีที่ 1 และ 3) วิธีพีบีอาร์ซีทูดีพีเอฟเอฟที (PBRC 2-D PFFT) ซึ่งเป็นวิธีที่นำเอาข้อดีของทั้ง 2 วิธีข้างต้นมาประยุกต์รวมกัน โดยนำเอาขั้นตอนในการติดต่อสื่อสารระหว่างหน่วยประมวลผลของวิธีวีอาร์ทูดีพีเอฟเอฟทีมาประยุกต์กับขั้นตอนการคำนวณของวิธีอาร์ซีทูดีพีเอฟเอฟที โดยจากผลการทดลองบนระบบพีซีคลัสเตอร์เพื่อวัดสมรรถนะของวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี พบว่าวิธีพีบีอาร์ซีทูดีพีเอฟเอฟที (PBRC 2-D PFFT) มีสมรรถนะในด้านการติดต่อสื่อสารระหว่างหน่วยประมวลผลดีกว่าวิธีอาร์ซีทูดีพีเอฟเอฟที (RC 2-D PFFT) และ มีสมรรถนะในด้านการคำนวณดีกว่าวิธีวีอาร์ทูดีพีเอฟเอฟที (VR 2-D PFFT)

Thesis	Two-Dimensional Parallel Fast Fourier Transform on a Cluster of PCs
Student	Mr. Apinan Sermsri
Student ID	47063708
Degree	Master of Science
Program	Computer Science
Year	2007
Thesis Advisor	Asst. Prof. Dr. Jeeraporn Srisawat

ABSTRACT

Fourier Transformation (FT) is a fundamentally important technique in various fields of technological research and development. Several applications of FT include scientific researches, engineering works and signal processing development. This study aims toward 2-dimensional parallel Fast Fourier Transformation (2-D PFFT), which can be classified into 3 methods: The Row-Column method, The Vector Radix method and The Partition-Block Row-Column method. The Row-Column (RC 2-D PFFT) method is practically used in many applications because it offers a straightforward development and faster processing time than other methods. The Vector Radix (VR 2-D PFFT) method is a fast Fourier transformation which offers a more direct response among the central processing units. The Partition-Block Row-Column (PBRC 2-D PFFT) method is a combined method of the first two methods, by utilizing the communication procedure from the VR method and the CPU response from the RC method. Performance of three 2-D PFFT algorithms was experimentally investigated on the PCs-cluster system. The results showed that CPU communication of the PBRC 2-D PFFT is better than that of the RC 2-D PFFT. In addition, its computation outperformed that of the VR 2-D PFFT.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้มิอาจจะสำเร็จลุล่วงไปได้ด้วยดี หากมิได้รับคำแนะนำ คำชี้แจง ความรู้ และความเอาใจใส่จาก ผศ.ดร.จีรพร ศรีสวัสดิ์ ผู้เป็นอาจารย์ที่ปรึกษา ซึ่งท่านได้สละเวลาให้กับข้าพเจ้าอย่างเต็มที่ จึงใคร่ขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ รศ.ดร.วีระ บุญจริง ผศ.ดร.ศรัณย์ อินทโกสุม และดร.เฉลิมศักดิ์ เลิศวงศ์เสถียร คณะกรรมการสอบหัวข้อ และโครงร่างวิทยานิพนธ์ ที่กรุณาให้คำแนะนำตลอดจนข้อชี้แนะจนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบพระคุณโครงการส่งเสริมการผลิตครูที่มีความสามารถพิเศษทางวิทยาศาสตร์และคณิตศาสตร์ (สควค.) โครงการพัฒนาครูแกนนำในสาขาคณิตศาสตร์ วิทยาศาสตร์และเทคโนโลยีและสิ่งแวดล้อม (Project Guiding Light) และมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสาร ที่ให้ทุนการศึกษาตลอดมา

ขอขอบพระคุณบิดา มารดา และพี่ๆ ที่สนันสนุนให้ได้เรียนในระดับที่ได้ตั้งใจ อีกทั้งยังได้ดูแลเรื่องค่าใช้จ่ายต่างๆระหว่างศึกษาเป็นอย่างดีอีกด้วย

ขอขอบคุณ นายนพรัตน์ พันธุ์เสนา และสำนักวิจัยและบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้ใช้เครื่องคลัสเตอร์ในการทำงานวิจัยรวมถึงให้คำแนะนำต่างๆที่เป็นประโยชน์ส่งผลให้วิทยานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณเพื่อนๆรวมรุ่นอันได้แก่ นายอัศวิน นิ่มกร นายองอาจ คูศรี นายทศนัย ชุ่มวัฒนะ นายมารุต ณ พัทลุง นายโยธิน เทียนดี นายสุเมธ รัตนจันทร์ นายพสิษฐ์ สสิกาญจน์ และพี่ๆน้องทุกคนที่ให้คำปรึกษา และช่วยอำนวยความสะดวกในด้านต่างๆ

สำหรับคุณงามความดีและประโยชน์อันใดที่เกิดขึ้นจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดา มารดา อาจารย์ทุกท่านซึ่งเป็นที่เคารพรักยิ่ง ตลอดจนญาติพี่น้อง รุ่นพี่และเพื่อนๆที่รักทุกคน

อภินันทน์ เสริมศรี

มีนาคม 2549

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I

บทคัดย่อภาษาอังกฤษ.....II

กิตติกรรมประกาศ.....III

สารบัญ.....IV

สารบัญตาราง.....VII

สารบัญรูป.....IX

บทที่ 1 บทนำ..... 1

1.1 ความเป็นมาและความสำคัญของปัญหา..... 1

1.2 วัตถุประสงค์ของการศึกษา..... 2

1.3 สมมติฐานการศึกษา..... 3

1.4 ขอบเขตการวิจัย..... 3

1.5 ขั้นตอนการศึกษาและการดำเนินงานวิจัย..... 3

1.6 ประโยชน์ที่คาดว่าจะได้รับ..... 4

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง..... 5

2.1 ทฤษฎีบทที่เกี่ยวกับจำนวนเชิงซ้อน..... 5

2.2 ทฤษฎีที่เกี่ยวข้องกับรากที่ N ของ 1 (Complex nth of root of unity : ω)..... 7

2.3 การแปลงฟูเรียร์(Fourier Transform)..... 10

2.3.1 การแปลงฟูเรียร์แบบต่อเนื่อง (Continuous Fourier Transform)..... 12

2.3.2 การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform)..... 12

2.3.3 การแปลงฟูเรียร์แบบเร็ว (Fast Fourier Transform)..... 13

2.4 การแปลงฟูเรียร์แบบไม่ต่อเนื่อง 2 มิติ (2-D Discrete Fourier Transform)..... 19

2.4.1 การแปลงฟูเรียร์แบบเร็ว 2 มิติโดยวิธีแถวและหลัก
(Row-Coloumn Two-Dimensional Fast Fourier Transform)..... 21

2.4.2 การแปลงฟูเรียร์แบบเร็ว 2 มิติโดยวิธีเวกเตอร์เรดิกซ์
(Vector radix Fast Fourier Transform)..... 24

2.5 การแปลงฟูเรียร์แบบเร็วแบบขนาน (Parallel Fast Fourier Transform)..... 32

2.5.1 การแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบไฮเปอร์คิว
(Fast Fourier Transform On Hypercube Network)..... 33

สารบัญ (ต่อ)

หน้า

2.5.2 การแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบเมสช์ (Fast Fourier Transform On Mash Network) 38

2.6 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน (2-D Parallel Fast Fourier Transform)..... 42

2.7 การวัดสมรรถนะของการประมวลผลแบบขนาน..... 45

2.7.1 เวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time)..... 46

2.7.2 อัตราการเพิ่มของความเร็ว (Speedup) 47

2.7.3 ประสิทธิภาพ (Efficiency) 47

2.8 งานวิจัยที่เกี่ยวข้อง 48

บทที่ 3 การแปลงฟูเรียร์แบบเร็วแบบขนานบนระบบพีซีคลัสเตอร์ 50

3.1 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยวิธี RC 2D-PFFT (Row-Column Two-Dimensional Fast Fourier Transform) 52

3.2 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยวิธี VR 2D-PFFT (Vector Radix Two-Dimensional Fast Fourier Transform) 57

3.3 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยวิธี PBRC 2D-PFFT (Partition Block Row-Column Two-Dimensional Fast Fourier Transform) 63

3.4 การวิเคราะห์ซับซ้อนด้านเวลา (Time Complexity Analysis)..... 70

3.4.1 ความซับซ้อนด้านเวลาของขั้นตอนวิธี RC 2-D PFFT..... 70

3.4.2 ความซับซ้อนด้านเวลาของขั้นตอนวิธี VR 2-D PFFT..... 71

3.4.3 ความซับซ้อนด้านเวลาของขั้นตอนวิธี PBRC 2-D PFFT 73

บทที่ 4 การทดลองและผลการทดลอง..... 76

4.1 ระบบคอมพิวเตอร์คลัสเตอร์ที่ใช้ในการทดลอง 76

4.2 ลักษณะข้อมูลที่ใช้ในการทดลอง..... 79

4.3 ผลการทดลอง..... 80

4.3.1 ผลการทดลองเปรียบเทียบกับเวลาในอุดมคติ 81

4.3.2 ผลการทดลองเปรียบเทียบเมื่อขนาดข้อมูลที่ใช้ทดลองต่างกัน 85

4.3.3 ผลการทดลองเปรียบเทียบเมื่อจำนวนหน่วยประมวลผลที่ใช้ในการทดลองต่างกัน..... 91

บทที่ 5 สรุปผลการทดลองและแนวทางการพัฒนา..... 96

5.1 สรุปและวิเคราะห์ผลการทดลอง..... 96

สารบัญ (ต่อ)

หน้า

5.2 แนวทางการพัฒนางานวิจัย 98

เอกสารอ้างอิง 99

ภาคผนวก ก. 101

ประวัติผู้ทำวิจัย 103

สารบัญตาราง

ตารางที่ หน้า

3.1 แสดงการเปรียบเทียบจำนวนการดำเนินการคำนวณ (Number of Arithmetic Operation) ของขั้นตอนวิธีทั้ง 3 74

3.2 แสดงเวลาทั้งหมดที่ใช้ในการประมวลผล (Total Parallel Execution Time) ของทั้ง 3 ขั้นตอนวิธี โดยแบ่งเป็นเวลาที่ใช้ในการคำนวณ และ เวลาที่ใช้ในการติดต่อสื่อสาร 75

4.1 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ของทั้ง 3 วิธีโดยทำการเปรียบเทียบกับเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติ (Response Time of Ideal Case) เมื่อข้อมูลมีขนาด 1024 × 1024 และใช้หน่วยประมวลผลตั้งแต่ 2 และ 4 หน่วยประมวลผล ที่มีเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติเป็น 2.56385 และ 1.37819 ตามลำดับ 82

4.2 แสดงอัตราการเพิ่มของความเร็วในอุดมคติ (Speedup) ของทั้ง 3 วิธีโดยทำการเปรียบเทียบกับเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติ (Response Time of Ideal Case) เมื่อข้อมูลมีขนาด 1024 × 1024 และใช้หน่วยประมวลผลตั้งแต่ 2 และ 4 หน่วยประมวลผล ที่มีเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติเป็น 2.56385 และ 1.37819 ตามลำดับ 83

4.3 แสดงแสดงประสิทธิภาพ (Efficiency) ของทั้ง 3 วิธีโดยทำการเปรียบเทียบกับเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติ (Response Time of Ideal Case) เมื่อข้อมูลมีขนาด 1024 × 1024 และใช้หน่วยประมวลผลตั้งแต่ 2 และ 4 หน่วยประมวลผล ที่มีเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติเป็น 2.56385 และ 1.37819 ตามลำดับ 84

4.4 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ 85

4.5 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ 86

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

4.6 แสดงแสดงประสิทธิภาพ (Efficiency) ทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล
และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 ,
2048 × 2048 และ 4096 × 4096 ตามลำดับ 87

4.7 แสดงเวลาที่ใช้ในการคำนวณ (Computation Time) ทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล
และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 ,
2048 × 2048 และ 4096 × 4096 ตามลำดับ 89

4.8 แสดงเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล (Communication Time)ทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล
และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 ,
2048 × 2048 และ 4096 × 4096 ตามลำดับ 90

4.9 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ทั้ง 3 วิธี
เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048
และใช้หน่วยประมวลผลเท่ากับ 2 และ4 หน่วยประมวลผล 91

4.10 แสดงอัตราการเพิ่มของความเร็วในอุดมคติ (Speedup) ทั้ง 3 วิธี
เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048
และใช้หน่วยประมวลผลเท่ากับ 2 และ4 หน่วยประมวลผล 92

4.11 แสดงแสดงประสิทธิภาพ (Efficiency) ทั้ง 3 วิธี
เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048
และใช้หน่วยประมวลผลเท่ากับ 2 และ4 หน่วยประมวลผล 93

5.1 แสดงการเปรียบเทียบข้อดี-ข้อเสียของวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติ
แบบขนานทั้ง 3 วิธีบนระบบพีซีคลัสเตอร์ 97

สารบัญรูป

รูปที่ หน้า

2.1 ค่าของจำนวนเชิงซ้อนในรูปคู่อันดับของจำนวนจริง 6

2.2 แสดงค่ารากที่หนึ่งของ 8 ของจำนวนเชิงซ้อน 7

2.3(a) แสดงข้อมูลของของฟังก์ชั่นในโดเมนเชิงเวลา (Time Domain) 11

2.3(b) แสดงข้อมูลของของฟังก์ชั่นในโดเมนเชิงความถี่ (Frequency Domain) 11

2.4(a) แสดงการขั้นตอนคำนวณการแปลงฟูเรียร์โดยวิธี Recursive 16

2.4(b) แสดงการขั้นตอนคำนวณการแปลงฟูเรียร์โดยวิธี Recursive 16

2.5 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบ Recursive 17

2.6 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็วแบบ Iterative 18

2.7 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบ Iterative 19

2.8 แสดงขั้นตอนวิธีแถวและหลัก (Row-Column Algorithm) 22

2.9 แสดงขั้นตอนการทำงาน 1 ขั้นของวิธีเวกเตอร์เรดิกซ์ของข้อมูลขนาด 2× 2 27

2.10 แสดงขั้นตอนการคำนวณแบบ Butterfly ของวิธีเวกเตอร์เรดิกซ์ 28

2.11 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็วแบบขนานมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบไฮเปอร์คิว กรณีที่มีหนึ่งข้อมูลต่อหนึ่งหน่วยประมวลผล 33

2.12 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็วแบบขนานมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบไฮเปอร์คิว กรณีที่มีหลายข้อมูลต่อหนึ่งหน่วยประมวลผล 35

2.13 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบไฮเปอร์คิว 37

2.14 แสดงการติดต่อระหว่างหน่วยประมวลผลแบบเมสช์ 38

2.15 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วบนการติดต่อแบบเมสช์ 39

2.16 แสดงขั้นตอนที่ 1 ในการคำนวณการแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบเมสช์ 39

2.17 แสดงขั้นตอนที่ 2 ในการคำนวณการแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบเมสช์ 40

2.18 แสดงขั้นตอนการคำนวณการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติของข้อมูลภายในหน่วยประมวลผลของตัวเอง 43

2.19 แสดงขั้นตอนการทรานสโพสของข้อมูลระหว่างหน่วยประมวลผลของตัวเอง 44

สารบัญรูป (ต่อ)

รูปที่ หน้า

2.20 แสดงขั้นตอนการคำนวณการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติของข้อมูล ภายในหน่วยประมวลผลของตัวเอง 45

3.1 แสดงเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) 53

3.2 แสดงการแบ่งข้อมูลเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) 54

3.3 แสดงขั้นตอนการทรานสโพสระหว่างหน่วยประมวลผล 55

3.4 แสดงขั้นตอนการทรานสโพสภายในหน่วยประมวลผล 55

3.5 แสดงตัวอย่างรหัสเหมือนเอ็มพีไอ-ซี (MPI-C like) วิธี RC 2-D PFFT 56

3.6 แสดงเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) 58

3.7 แสดงการสลับบิต(Bit Reverse Order) เมตริกซ์ A 58

3.8 แสดงการแบ่งข้อมูลเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) 59

3.9 แสดงการรวมข้อมูลจากแต่ละหน่วยประมวลผลทำงาน เพื่อให้หน่วยประมวลผลหลักทำการ คำนวณด้วยวิธี VR 2-D PFFT 61

3.10 แสดงตัวอย่างรหัสเหมือนเอ็มพีไอ-ซี (MPI-C like) วิธี VR 2-D PFFT 62

3.11 แสดงขั้นตอนการทำงานของวิธี PBRC 2-D PFFT 63

3.12 แสดงเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) 65

3.13 แสดงการสลับบิต(Bit Reverse Order) เมตริกซ์ A 65

3.14 แสดงการแบ่งข้อมูลเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) 66

3.15 แสดงการคำนวณการแปลงฟูเรียร์ด้วยวิธี RC ของหน่วยประมวลผลที่ 1 67

3.16 แสดงการรวมข้อมูลจากแต่ละหน่วยประมวลผลทำงาน เพื่อให้หน่วยประมวลผลหลักทำการ คำนวณด้วยวิธี Vector Radix 68

3.17 แสดงตัวอย่างรหัสเหมือนเอ็มพีไอ-ซี (MPI-C like) วิธี PBRC 2-D PFFT 69

4.1 ระบบคลัสเตอร์ที่มีการเก็บข้อมูลแบบรวมศูนย์กลาง 80

4.2 แสดงกราฟเปรียบเทียบเวลาทั้งหมดที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน กับเวลาในอุดมคติ 82

สารบัญรูป (ต่อ)

รูปที่ หน้า

4.3 แสดงกราฟเปรียบเทียบอัตราการเพิ่มของความเร็ว
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานกับเวลาในอุดมคติ 83

4.4 แสดงกราฟเปรียบเทียบประสิทธิภาพ
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานกับเวลาในอุดมคติ 84

4.5 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการประมวลผลทั้งหมด
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล 86

4.6 แสดงกราฟเปรียบเทียบอัตราการเพิ่มของความเร็ว
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล 87

4.7 แสดงกราฟเปรียบเทียบประสิทธิภาพ
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล 88

4.8 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการคำนวณ
การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล 89

4.9 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล
การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล 90

4.10 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการประมวลผลทั้งหมด
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 92

4.11 แสดงกราฟเปรียบเทียบอัตราการเพิ่มของความเร็ว
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 93

4.12 แสดงกราฟเปรียบเทียบประสิทธิภาพ
ที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี
เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 94

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันนี้ได้มีการศึกษาและประยุกต์ใช้วิธีการแปลงข้อมูล (Data Tranformation) แบบต่างๆอย่างกว้างขวาง ซึ่งวิธีการแปลงข้อมูลจะใช้หลักการทางคณิตศาสตร์เฉพาะแบบต่างๆเพื่อนำไปใช้ในการช่วยวิเคราะห์ และหาคำตอบของปัญหาต่างๆที่มีความยุ่งยากและซับซ้อน โดยเฉพาะอย่างยิ่งการแปลงข้อมูลที่เกี่ยวข้องกับจำนวนเชิงซ้อน เช่น การแปลงฟูเรียร์ (Fourier Tranfrom) ซึ่งถูกนำไปใช้อย่างแพร่หลายในด้านการวิเคราะห์ระบบเชิงเส้นที่ไม่มีการเปลี่ยนค่าตัวแปรใดๆในแต่ละอาณาเขตของการเปลี่ยนแปลง เช่น ตัวแปรทางเวลา ตัวแปรทางความถี่ เป็นต้น แต่จากความก้าวหน้าทางด้านเทคโนโลยี ทำให้มีการนำอุปกรณ์ที่มีประสิทธิภาพทางด้านการคำนวณ เช่น คอมพิวเตอร์ มาช่วยในการคำนวณเกี่ยวการแปลงฟูเรียร์ ซึ่งภายหลังได้ถูกพัฒนามาเป็นการแปลงฟูเรียร์ที่เรียกว่า การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform : DFT) สำหรับประมวลผลโดยคอมพิวเตอร์และในปัจจุบันการแปลงฟูเรียร์ดังกล่าวมีความสำคัญอย่างมากต่องานสำคัญต่างๆ ทางด้านวิศวกรรมศาสตร์และวิทยาศาสตร์ ที่พบได้บ่อยๆเช่น การประมวลผลสัญญาณดิจิทัล (Digital Signal Processing) การประมวลผลการรู้จำเสียงพูด (Voice Recognition) การแก้ระบบสมการเชิงอนุพันธ์ย่อย และการประมวลผลภาพ (Image Processing) โดยเฉพาะ การกรองภาพ (Image Filtering) หรือการบีบอัดสัญญาณภาพดิจิตอล ซึ่งเป็นพื้นฐานการเข้ารหัสทั้งสัญญาณภาพดิจิตอล และสัญญาณวีดิทัศน์ดิจิตอลที่ใช้ในปัจจุบัน และแม้แต่ในการประยุกต์ใช้ด้านการแพทย์ ตัวอย่างเช่น MRI Ultrasound Image หรือการสนับสนุนแพทย์ในการตรวจวินิจฉัยข้อมูลที่ได้จากการวัดและเก็บบันทึกจากผู้ป่วย ก็มีการนำความรู้เกี่ยวกับการแปลงฟูเรียร์แบบไม่ต่อเนื่องไปใช้ในการพัฒนาทั้งสิ้น ซึ่งต่อมาการแปลงดังกล่าวได้มีการศึกษาและพัฒนาต่อมาเป็นวิธีการแปลงฟูเรียร์แบบใหม่ที่มีสมรรถนะมากกว่าเดิมที่เรียกว่า การแปลงฟูเรียร์แบบเร็ว (Fast Fourier Transform : FFT) ซึ่งมีประสิทธิภาพในการทำงานที่เร็วกว่าการแปลงฟูเรียร์แบบเดิม [24]

แต่เนื่องจากการแปลงฟูเรียร์แบบไม่ต่อเนื่องมีขั้นตอนการทำงานที่ซับซ้อนเพราะการแปลงฟูเรียร์จะต้องทำการคูณและบวกจำนวนเชิงซ้อน ซึ่งทำให้คอมพิวเตอร์ที่มีเพียงหนึ่งหน่วยประมวลผลใช้เวลาในการประมวลผลมาก ถึงแม้ว่าจะประยุกต์ใช้การแปลงฟูเรียร์แบบเร็ว (Fast Fourier Transform : FFT) แล้วก็ตาม ดังนั้น จึงมีการวิจัยและพัฒนาที่มีประสิทธิภาพมากขึ้นโดยนำวิธีการแปลงฟูเรียร์มาพัฒนาบนเครื่องคอมพิวเตอร์ที่มีสมรรถนะสูง เช่น ซุปเปอร์คอมพิวเตอร์ โดย

ระบบซุปเปอร์คอมพิวเตอร์ ที่ใช้คอมพิวเตอร์แบบขนานที่มีหลายหน่วยประมวลผลและเทคนิคการประมวลผลแบบขนานเข้ามาเพิ่มสมรรถนะทางด้านความเร็วในการประมวลผล

ในอดีตที่ผ่านมาได้มีการศึกษางานวิจัยเกี่ยวกับการแปลงฟูเรียร์แบบเร็วแบบขนาน (Parallel Fast Fourier Transform) มากมายหลายด้านไม่ว่าจะเป็นทางด้านขั้นตอนวิธี (Parallel Computer Algorithm) หรือ ทางด้านการประยุกต์ (Parallel Computer Application) เช่น การศึกษาและพัฒนาวิธีการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติบนเครื่องคอมพิวเตอร์แบบขนานที่มีโครงสร้างของหน่วยประมวลผลติดต่อกันแบบไฮเปอร์คิว [2] หรือ การศึกษาและพัฒนาวิธีการแปลงฟูเรียร์แบบเร็วบนเครื่องคอมพิวเตอร์แบบขนานโดยใช้ขั้นตอนทางคอมพิวเตอร์แบบ MIMD ภายใต้โครงสร้างพื้นฐานที่ชื่อว่า วีเอ็มเอลเอสซีเอส (VMLSCS : Virtual Machine Loosely Synchronous Communication System) ซึ่งจะใช้หน่อยความจำแบบ Hierarchical [19] เป็นต้น

งานวิจัยนี้ได้เสนอการประยุกต์ใช้ขั้นตอนวิธีแบบขนานบนระบบพีซีคลัสเตอร์ (A Cluster of Pcs) โดยเน้นเรื่องการแปลงฟูเรียร์แบบเร็ว 2 มิติ (2-D Fast Fourier Transform : 2-D FFT) ซึ่งจะทำการพัฒนาวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน (2-D Parallel Fast Fourier Transform : 2-D PFFT) แบบต่างๆ แล้วทำการเปรียบเทียบสมรรถนะ (Performance) ของแต่ละวิธีโดยประเมินผลจากเวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time) อัตราการเพิ่มขึ้นของความเร็ว (Speedup) และประสิทธิภาพ (Efficiency)

1.2 วัตถุประสงค์ของการศึกษา

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน (Parallel Fast Fourier Transform : PFFT) พร้อมนำเสนอวิธีการเพิ่มสมรรถนะ (Performance) ของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์ ดังนี้

1) ลดระยะเวลาในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบอนุกรมที่ใช้หนึ่งหน่วยประมวลผล โดยการใช้หน่วยประมวลผลเพิ่มขึ้นและประมวลผลแบบขนาน ในกรณีที่ข้อมูลมีขนาดใหญ่ๆ

2) จัดสรรงานที่เหมาะสมให้แต่ละหน่วยประมวลผลของระบบพีซีคลัสเตอร์

3) ทำการเปรียบเทียบสมรรถนะของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธีต่างๆ เพื่อหาข้อดีและข้อจำกัดของวิธีต่างๆ ที่เสนอ

1.3 สมมติฐานการศึกษา

1) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยใช้จำนวนหน่วยประมวลผลเพิ่มขึ้นสำหรับข้อมูลขนาดใหญ่เป็นการช่วยให้เวลาในการประมวลผลในการแปลงฟูเรียร์ลดลง

2) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานมีสมรรถนะมากกว่าการแปลงฟูเรียร์ อย่างเร็วแบบอนุกรม ในกรณีที่ข้อมูลมีขนาดใหญ่ ๆ

3) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธีใหม่ที่พัฒนาขึ้นมีสมรรถนะสูงกว่าการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานในด้านลดการติดต่อสื่อสารระหว่างหน่วยประมวลผล

1.4 ขอบเขตการวิจัย

ขอบเขตการวิจัยของวิทยานิพนธ์นี้มี ดังต่อไปนี้

1) การแปลงฟูเรียร์แบบไม่ต่อเนื่องที่ศึกษาเป็นการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบอนุกรมและแบบขนาน

2) ทำการพัฒนาวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบที่มีการประมวลผลแบบขนาน โดยระบบที่ใช้พัฒนา คือ ระบบพีซีคลัสเตอร์ (A Cluster of PCs)

3) การตรวจสอบข้อมูลที่ใช้ในการทดลองจะทำการเปรียบเทียบโดยใช้ไลบรารี่มาตรฐานที่ใช้ในการแปลงฟูเรียร์แบบขนาน คือ FFTW เวอร์ชั่น 2.1.5 (ที่มา http://www.fftw.org)

4) การวัดสมรรถนะของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธีต่างๆ ทำการวัดจากเวลาทั้งหมดที่ใช้ในการประมวลผล อัตราการเพิ่มขึ้นของความเร็วและประสิทธิภาพ

5) งานวิจัยนี้จะทำการเปรียบเทียบสมรรถนะของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานระหว่างขั้นตอนวิธีแบบใหม่ กับขั้นตอนวิธีแบบเก่า ซึ่งทำการพัฒนาบนระบบพีซีคลัสเตอร์

1.5 ขั้นตอนการศึกษาและการดำเนินงานวิจัย

วิทยานิพนธ์นี้มีขั้นตอนการศึกษาและการดำเนินงานวิจัย ดังนี้

1) ศึกษาขั้นตอนวิธีการทางคอมพิวเตอร์ (Computer Algorithm) ของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบอนุกรมและแบบขนาน

2) ศึกษางานวิจัยที่เกี่ยวข้องกับการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน

3) ทำการตั้งสมมติฐานดังนี้ 1) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่ได้นำเสนอเมื่อใช้จำนวนหน่วยประมวลผลเพิ่มขึ้น จะเป็นการช่วยให้เวลาในการประมวลผล และมีสมรรถนะมากขึ้นโดยเฉพาะกรณีที่ข้อมูลมีขนาดใหญ่ๆ และ 2) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่ได้จากการนำข้อดีของวิธีต่างๆมารวมกันจะเป็นการช่วยให้เวลาในการติดต่อสื่อสารระหว่างหน่วยประมวลผลลดลงและมีสมรรถนะเพิ่มขึ้น

4) นำเสนอวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่มีสมรรถนะดังที่ได้ตั้งสมมติฐานไว้ในข้อ (3)

5) พัฒนาโปรแกรมการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่นำเสนอโดยใช้โปรแกรมภาษาซี (C Language) บนระบบพีซีคลัสเตอร์ที่ใช้ระบบปฏิบัติการลีนุกซ์ (Linux) และตรงตามมาตรฐานภาษาเอ็มพีไอ (MPI Standard) ที่สนับสนุนการโปรแกรมแบบขนาน

6) วิเคราะห์ผลการทดลองโดยการเปรียบเทียบสมรรถนะของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยประเมินผลจากเวลาทั้งหมดที่ใช้ในการประมวลผล อัตราการเพิ่มขึ้นของความเร็วและประสิทธิภาพ

7) สรุปผลการทดลอง พร้อมเสนอแนวทางการพัฒนางานวิจัย

8) เขียนวิทยานิพนธ์

1.6 ประโยชน์ที่คาดว่าจะได้รับ

งานวิจัยนี้มีประโยชน์ที่คาดว่าจะได้รับจากการทดลองบนระบบพีซีคลัสเตอร์ ดังนี้

1) สามารถหาผลการแปลงฟูเรียร์แบบเร็ว 2 มิติ กรณีที่ข้อมูลมีขนาดใหญ่ๆ โดยใช้เวลาน้อยลงใช้หน่วยความจำน้อยลงและสามารถนำไปพัฒนาใช้ในงานวิจัยที่เกี่ยวข้องได้

2) นำมาใช้เป็นแนวทางในการพัฒนาโปรแกรมคอมพิวเตอร์แบบขนานเพื่อให้สามารถใช้งานได้สะดวกและเข้าใจได้ง่าย

3) นำไปประยุกต์พัฒนากับวิธีการแปลงข้อมูลทางคณิตศาสตร์แบบอื่นๆ เช่น ดีสครีตไซน์ทรานสฟอร์ม (Discrete Sine Transform) ดีสครีตโคไซน์ทรานสฟอร์ม (Discrete Cosine Transform) หรือ ฮาร์ทเลย์ทรานสฟอร์ม (Hartley Transform) เป็นต้น

4) นำไปประยุกต์ใช้กับการพัฒนาโปรแกรมด้านต่างๆ เช่น งานทางด้านการประมวลผลสัญญาณดิจิทัล (Digital Signal Processing) หรือ งานทางด้านการประมวลผลรูปภาพดิจิทัล (Digital Image Processing) เป็นต้น

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีบทที่เกี่ยวกับจำนวนเชิงซ้อน

จำนวนเชิงซ้อนเป็นคู่ลำดับของจำนวนจริงที่อยู่ในรูป (x , y) โดยเราจะเรียก x ว่าส่วนจริง (Real Part) และ y ว่าส่วนจินตภาพ (Imaginary Part) [17] [18]

จำนวนเชิงซ้อนสองจำนวนคือ (x_1, y_1) และ (x_2, y_2) จะเท่ากัน ก็ต่อเมื่อ $x_1 = x_2$ และ $y_1 = y_2$

ถ้าให้ $z_1 = (x_1, y_1)$ และ $z_2 = (x_2, y_2)$ เป็นจำนวนเชิงซ้อนสองจำนวนแล้ว ผลบวกของจำนวนเชิงซ้อนทั้งสองจำนวนจะมีค่าเท่ากับ

$$z_1 + z_2 = (x_1 + x_2, y_1 + y_2)$$

และผลคูณของจำนวนเชิงซ้อนทั้งสองจำนวนจะมีค่าเท่ากับ

$$z_1 \cdot z_2 = (x_1 x_2 - y_1 y_2, x_1 y_2 + y_1 x_2)$$

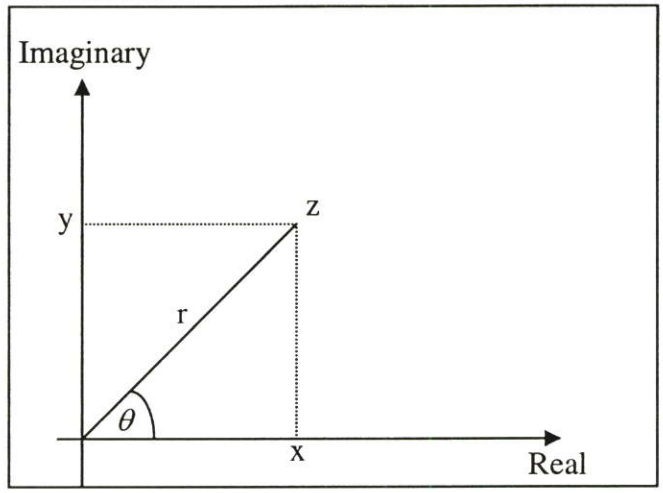
โดยแต่ละจำนวนจริง x สามารถเขียนอยู่ในรูปจำนวนเชิงซ้อนคือ (x , 0)

<u>ทฤษฎีบทที่ 1</u> ทุกจำนวนเชิงซ้อน z = (x , y) สามารถเขียนให้อยู่ในรูป $x + iy$ ได้ เมื่อ i คือองค์ประกอบของส่วนจินตภาพที่มีค่าเท่ากับ รากที่สองของ -1 ซึ่งจะมีค่าในรูปของจำนวนเชิงซ้อนเป็น (0 ,1)

พิสูจน์ จำนวนจริง x = (x , 0) และผลคูณของจำนวนจริง y กับ ค่า i คือ

$$iy = (0,1)(y,0) = (0 \times y - 1 \times 0, 0 \times 0 + y \times 1) = (0, y)$$

ดังนั้น $x + iy = (x, 0) + (0, y) = (x, y)$



รูปที่ 2.1 แสดงค่าของจำนวนเชิงซ้อนในรูปคู่อันดับของจำนวนจริงเมื่อ x แทนส่วนจริง(Real Part) และ y แทนส่วนจินตภาพ (Image Part)

เราจะได้ว่า z เป็นเวกเตอร์ที่มีความยาว r หน่วย และมีมุมขนาด θ เมื่อ θ คือมุมที่เวกเตอร์ z ทำกับแกนของส่วนจริง (Real Axis) ดังนั้น เราจะได้ว่า

$$x = r\cos\theta$$

$$y = r\sin\theta$$

ทำให้เราสามารถเขียนสมการได้เป็น $z = x + iy = r(\cos\theta + i\sin\theta)$

ซึ่งเราสามารถเขียน z ให้อยู่ในรูปของสมการเอ็กซ์โพเนนเชี่ยลได้โดยใช้อนุกรมเทย์เลอร์ได้ดังนี้

$$\sin\theta = \theta - \frac{\theta^3}{3} + \frac{\theta^5}{5} - \frac{\theta^7}{7} + ...$$

$$\cos\theta = 1 - \frac{\theta^2}{2} + \frac{\theta^4}{4} - \frac{\theta^6}{6} + ...$$

$$e^{i\theta} = 1 + i\theta - \frac{\theta^2}{2} - \frac{i\theta^3}{3} + \frac{\theta^4}{4} + \frac{i\theta^5}{5} + ...$$

$$= \left(1 - \frac{\theta^2}{2} + \frac{\theta^4}{4} + ...\right) + i\left(\theta - \frac{\theta^3}{3} + \frac{\theta^5}{5} + ...\right)$$

ดังนั้นจะได้ว่า $e^{i\theta} = \cos\theta + i\sin\theta$

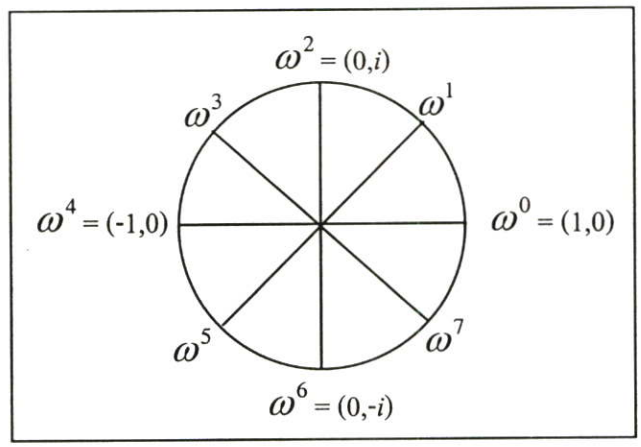
และ $e^{-i\theta} = \cos\theta - i\sin\theta$

จากที่เราได้ว่า $z = x + iy = r(\cos\theta + i\sin\theta)$ ดังนั้นจะได้ว่า

$$z = r\,e^{i\theta}$$

2.2 ทฤษฎีที่เกี่ยวข้องกับ ω_N

เราเรียก ω ว่า รากที่ N ของ 1 ของจำนวนเชิงซ้อน (Complex n^{th} of root of unity) ซึ่ง $\omega^N = 1$ [21]



รูปที่ 2.2 แสดงค่ารากที่หนึ่งของ 8 ของจำนวนเชิงซ้อน

<u>บทนิยาม</u>

$$\omega_N = e^{i(2\pi/N)} \tag{1}$$
$$= \cos(2\pi/N) + i\sin(2\pi/N)$$

จากนิยามถ้าเรายกกำลัง ω_N ถึงกำลังที่ N นั่นคือ ω_N^N ซึ่งเราจะเห็นได้ว่า ω_N^N มีค่าเท่ากับ

$$\omega_N^N = e^{i(2\pi/N)\cdot N}$$
$$= e^{i(2\pi)}$$
$$= \cos(2\pi) + i\sin(2\pi)$$
$$= 1 \tag{2}$$

จากสมการที่ 2 จะได้ว่า ω_N คือรากที่ N ของ 1

<u>ทฤษฎีย่อยที่ 1.1</u> ถ้า ω_N เป็นรากที่ N ของ 1 แล้ว เราจะได้ว่า ω_N^k เป็นรากที่ N ของ 1 ด้วยนั่นคือ

$$(\omega_N^k)^N = 1$$

พิสูจน์ จากเงื่อนไขข้างต้นในสมการที่ 2 เราจะได้ว่า

ถ้า ω_N เป็นรากที่ N ของ 1 แล้ว เราจะได้ว่า $\omega_N^N = 1$

$$(\omega_N^k)^N = \omega_N^{kN}$$
$$= (\omega_N^N)^k$$
$$= 1^k$$
$$= 1$$

ดังนั้นจึงสรุปได้ว่า ω_N^k เป็นรากที่ N ของ 1

ซึ่งจากทฤษฎีย่อยที่ 1.1 ข้างต้น ทำให้เราสรุปได้ว่า $\omega_N^0, \omega_N^1, ..., \omega_N^{N-1}$ ต่างก็เป็นรากที่ N ของ 1 ได้ทั้งหมด ทำได้ว่าค่าของ ω_N^k เมื่อ k > N จะต้องเท่ากับค่าใดค่าหนึ่งของ $\omega_N^0, \omega_N^1, ..., \omega_N^N$ ซึ่งสามารถพิสูจน์ได้โดย

พิสูจน์ ให้ k = N+C (k > N) เราจะได้ว่า

$$\omega_N^{N+C} = \omega_N^N \cdot \omega_N^C$$
$$= \omega_N^C \qquad (\because \omega_N^N = 1)$$

<u>ทฤษฎีย่อยที่ 1.2</u> ถ้าให้ d และ k เป็นจำนวนเต็มบวก ที่มากกว่า 0 เราจะได้ว่า $\omega_{dN}^{dk} = \omega_N^k$

พิสูจน์ เนื่องจากนิยามของ $\omega_N = e^{i(2\pi / N)}$ ถ้าเราแทนค่า N ด้วย dN จะได้ว่า

$$\omega_{dN} = e^{i(2\pi / dN)}$$

และ
$$\omega_{dN}^{dk} = e^{i(2\pi / dN)\cdot dk}$$
$$= e^{i(2\pi / N)k}$$
$$= \omega_N^k$$

<u>ทฤษฎีย่อยที่ 1.3</u> $\omega_N^{-1} = \omega_N^{N-1}$

พิสูจน์
$$\omega_N^{-1} = 1 \cdot \omega_N^{-1}$$
$$= \omega_N^N \cdot \omega_N^{-1}$$
$$= \omega_N^{N-1}$$

<u>ทฤษฎีย่อยที่ 1.4</u> $\omega_N^{N/2} = \omega_2 = -1$ เมื่อ N เป็นเลขคู่

พิสูจน์ จากบทนิยาม $\omega_N = e^{i(2\pi/N)}$

ถ้ายกกำลัง N/2 ทั้งสองข้างของสมการจะได้ว่า

$$\begin{aligned}\omega_N^{N/2} &= e^{i(2\pi/N)\cdot N/2} \\ &= e^{i(\pi)} \\ &= \cos(\pi) + i\sin(\pi) \\ &= -1\end{aligned}$$

หรือเขียนใหม่ได้ว่า

$$\begin{aligned}\omega_N^{N/2} &= \cos(2\pi/2) + i\sin(2\pi/2) \\ &= e^{i(2\pi/2)} \\ &= \omega_2\end{aligned}$$

ดังนั้นจึงสรุปได้ว่า $\omega_N^{N/2} = \omega_2 = -1$

<u>ทฤษฎีย่อยที่ 1.5</u> $(\omega_N^k)^2 = \omega_{N/2}^k$

พิสูจน์ จากบทนิยาม $\omega_N = e^{i(2\pi/N)}$

ถ้าเอา 2/2 คูณกับค่า N เราจะได้ว่า

$$\begin{aligned}\omega_N &= e^{i(2\pi/N)\cdot 2/2} \\ &= e^{i(2\pi/N/2)\cdot 1/2} \\ &= \omega_{N/2}^{1/2}\end{aligned}$$

ดังนั้น $$\begin{aligned}(\omega_N^k)^2 &= (\omega_{N/2}^{1/2})^{2k} \\ &= \omega_{N/2}^k\end{aligned}$$

<u>ทฤษฎีย่อยที่ 1.6</u> $(\omega_N^{k+N/2})^2 = (\omega_N^k)^2$ เมื่อ N เป็นเลขคู่

พิสูจน์ $$\begin{aligned}(\omega_N^{k+N/2})^2 &= \omega_N^{2k+N} \\ &= \omega_N^{2k}\cdot\omega_N^N \\ &= \omega_N^{2k}\cdot 1 \\ &= (\omega_N^k)^2\end{aligned}$$

ทฤษฎีย่อยที่ 1.7 ถ้า k หารด้วย N ไม่ลงตัวเราจะได้ว่า $\sum_{i=0}^{N-1}(\omega_N^k)^i = 0$

พิสูจน์ $$\sum_{i=0}^{N-1}(\omega_N^k)^i = \frac{(\omega_N^k)^N - 1}{\omega_N^k - 1}$$
$$= \frac{(\omega_N^N)^k - 1}{\omega_N^k - 1}$$
$$= \frac{1^k - 1}{\omega_N^k - 1}$$
$$= 0$$

บทแทรกที่ 1.1 $\omega_N^{(N/2)+b} = -\omega_N^b$ เมื่อ $0 \le b \le \frac{N}{2} - 1$

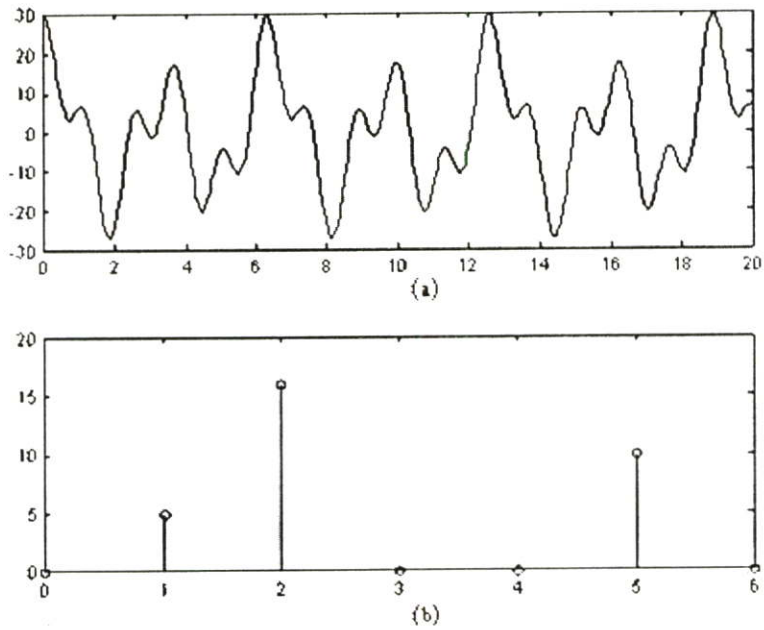
พิสูจน์ $\omega_N^{(N/2)+b} = \omega_N^{N/2} \cdot \omega_N^b$

จากทฤษฎีย่อยที่ 1.4 เราจะได้ว่า $\omega_N^{N/2} = -1$

ดังนั้น $$\omega_N^{(N/2)+b} = \omega_N^{N/2} \cdot \omega_N^b$$
$$= (-1) \cdot \omega_N^b$$
$$= -\omega_N^b$$

2.3 การแปลงฟูเรียร์ (Fourier Transform)

ในการวิเคราะห์ข้อมูลบางอย่างของฟังก์ชันในโดเมนเชิงเวลา (Time Domain) นั้นจะพิจารณาการเปลี่ยนแปลงของฟังก์ชันเทียบกับเวลา ในกรณีที่เป็นฟังก์ชันซ้ำตัวเองนั้น ข้อมูลที่ได้หลังจากหนึ่งคาบของฟังก์ชันจะมีค่าเท่าเดิมไม่มีการเปลี่ยนแปลง แต่มีข้อมูลบางอย่างที่ฟังก์ชันในโดเมนเชิงเวลาที่ไม่สามารถทำการวิเคราะห์ได้ เช่น การวิเคราะห์การสัญญาณบางอย่าง ทำให้ต้องทำการแปลงฟังก์ชันที่อยู่ในโดเมนเชิงเวลาให้มาอยู่ในโดเมนเชิงความถี่ (Frequency Domain) แทน โดยใช้การแปลงฟูเรียร์ (Fourier Transform) ซึ่งเป็นขั้นตอนในการเปลี่ยนฟังก์ชันที่อยู่ในโดเมนเชิงเวลาให้อยู่ในโดเมนเชิงความถี่ เช่น ในการวัดการสั่นของเครื่องจักรหนึ่ง จากกราฟดังรูปที่ 2.3(a) ที่จะแสดงเฉพาะขนาดของการสั่น ณ เวลาที่เท่าใด แต่จะไม่ทราบว่าการสั่นมาจากชิ้นส่วนใดและแต่ละชิ้นมีการสั่นมากน้อยเท่าใด ซึ่งถ้าทำการแปลงข้อมูลของฟังก์ชันให้อยู่ในรูปข้อมูลเชิงความถี่ดังกราฟรูปที่ 2.3(b) ก็จะทำให้สามารถวิเคราะห์ข้อมูลดังกล่าวได้



รูปที่ 2.3 (a) และ (b) แสดงข้อมูลของของฟังก์ชันในโดเมนเชิงเวลา (Time Domain)และโดเมนเชิงความถี่ (Frequency Domain) ตามลำดับ

การแปลงฟูเรียร์ (Fourier Transform) จะมีชื่อเรียกต่างกันออกไปตามลักษณะของฟังก์ชัน กล่าวคือ

1. ฟังก์ชันมีลักษณะเป็นคาบแบบต่อเนื่อง (Periodic and Continous)
   จะมีชื่อเรียกว่า อนุกรมฟูเรียร์ (Fourier Series : FS)
2. ฟังก์ชันไม่มีลักษณะเป็นคาบแบบต่อเนื่อง (Non Periodic and Continous)
   จะมีชื่อเรียกว่า การแปลงฟูเรียร์ (Fourier Transform : FT)
3. ฟังก์ชันมีลักษณะเป็นคาบแบบไม่ต่อเนื่อง (Periodic and Discrete)
   จะมีชื่อเรียกว่า อนุกรมฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Series : DFS)
4. ฟังก์ชันไม่มีลักษณะเป็นคาบแบบไม่ต่อเนื่อง (Non Periodic and Discrete)
   จะมีชื่อเรียกว่า การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform : DFT)

โดยในงานวิจัยนี้จะทำการศึกษาการแปลงฟูเรียร์ในลักษณะที่เป็นฟังก์ชันไม่มีคาบแบบไม่ต่อเนื่อง (Non Periodic and Discrete) หรือ การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform : DFT) เนื่องจากในการคำนวณเพื่อหาค่าการแปลงฟูเรียร์โดยใช้เครื่องมือในการคำนวณจำเป็นต้องมีค่าที่จำกัด

2.3.1 การแปลงฟูเรียร์แบบต่อเนื่อง (Continuous Fourier Transform)

การแปลงฟูเรียร์แบบต่อเนื่อง (Continuous Fourier Transform) มีรูปแบบการแปลงดังนี้

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{2\pi i f} df$$

เมื่อ $X(f)$ อยู่ในรูปฟังก์ชันของความถี่ สามารถเขียน $X(f)$ ได้ดังนี้

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi i f t} dt$$

ซึ่ง $X(f)$ คือ สเปกตรัม (Spectrum) ของ $x(t)$ หรือ คือ การแปลงฟูเรียร์ของ $x(t)$

2.3.2 การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform)

การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform :DFT) เป็นการแปลงฟูเรียร์แบบหนึ่งที่เกิดจากการที่เราต้องการคำนวณหาค่าการแปลงฟูเรียร์ โดยใช้อุปกรณ์การคำนวณ เช่น คอมพิวเตอร์ มาช่วยคำนวณการแปลงฟูเรียร์ ซึ่งข้อมูลนำเข้า (Input) จะเป็นสัญญาณเชิงเวลา (Time Domain) ที่ผ่านการดึงตัวอย่าง (Sampling) แล้ว ส่วนข้อมูลขาออกจะเป็นข้อมูลเชิงความถี่ (Frequency Domain) แต่เนื่องจากการแปลงฟูเรียร์เป็นการคำนวณที่มีลำดับข้อมูลไม่จำกัด ดังนั้น หากต้องการใช้คอมพิวเตอร์มาช่วยในการคำนวณการแปลงฟูเรียร์นั้นเราต้องการลำดับข้อมูลที่เป็นจำนวนจำกัด ซึ่งเราหาได้จากการที่เราจะเลือกช่วงของอนุกรมฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Series : DFS) มา 1 ช่วงโดยกำหนดให้เป็น 1 คาบ แล้วทำการวิเคราะห์ โดยเราจะเรียกการแปลงอนุกรมฟูเรียร์ แบบไม่ต่อเนื่องของสัญญาณ 1 คาบนั้นว่า การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform :DFT) ซึ่งมีลักษณะเป็นสัญญาณเชิงความถี่แบบไม่ต่อเนื่องและมีลักษณะเป็นคาบ เพื่อนำไปใช้ในการวิเคราะห์ หรือจัดการกับสัญญาณนั้นในเชิงความถี่ เช่น กรองสัญญาณเฉพาะช่วงความถี่ที่ต้องการเป็นต้น ดังนั้นจึงสามารถนำไปใช้ในการแยกความถี่ของเสียงคนได้ [12] [15] [17] [18] [20] [24]

การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete fourier transfrom: DFT) ในรูปของเมตริกซ์ [13] [17] [18] แสดงได้ดังนี้

ให้ $\omega_n = \exp(\frac{-2\pi i}{n})$ และ y = DFT(x) จะได้ว่า

$$y_k = \sum_{j=0}^{n-1} \omega_n^{kj} x_j \quad , \text{k=0,1, ... , } n\text{-1} \qquad (2.1)$$

หรือ

$$y \equiv F_n x \qquad (2.2)$$

โดยที่ F_n คือ เมตริกซ์ขนาด $n \times n$ ซึ่งเขียนได้ดังนี้

$$[F_n]_{kj} = \omega_n^{kj} = e^{(\frac{-2\pi i}{n} \cdot kj)} \quad \text{k , j = 0,1, ... , } n\text{-1}$$

เช่น $F_1 = [1]$

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

<u>ตัวอย่างที่ 2.1</u> แสดงการคำนวณการแปลงฟูเรียร์แบบไม่ต่อเนื่องแบบ 1 มิติ

(1) คำนวณค่า DFT ของเวกเตอร์ (2,3) จะได้ค่าดังนี้

$$\begin{pmatrix} \omega_2^{0\times0} & \omega_2^{0\times1} \\ \omega_2^{1\times0} & \omega_2^{1\times1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 5 \\ -1 \end{pmatrix}$$

หรือ (2) คำนวณค่า DFT ของเวกเตอร์ (1,2,4,3) จะได้ค่าดังนี้

$$\begin{pmatrix} \omega_4^0 & \omega_4^0 & \omega_4^0 & \omega_4^0 \\ \omega_4^0 & \omega_4^1 & \omega_4^2 & \omega_4^3 \\ \omega_4^0 & \omega_4^2 & \omega_4^4 & \omega_4^6 \\ \omega_4^0 & \omega_4^3 & \omega_4^6 & \omega_4^9 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \omega_4^0 & \omega_4^0 & \omega_4^0 & \omega_4^0 \\ \omega_4^0 & \omega_4^1 & \omega_4^2 & \omega_4^3 \\ \omega_4^0 & \omega_4^2 & \omega_4^0 & \omega_4^2 \\ \omega_4^0 & \omega_4^3 & \omega_4^2 & \omega_4^1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 10 \\ -3-i \\ 0 \\ -3+i \end{pmatrix}$$

2.3.3 การแปลงฟูเรียร์แบบเร็ว (Fast Fourier Transform : FFT)

จากการที่ได้นำคอมพิวเตอร์มาช่วยในการคำนวณการแปลงฟูเรียร์แบบไม่ต่อเนื่อง (DFT) นั้น ก็ทำให้เกิดปัญหาที่สำคัญประการหนึ่ง คือ เรื่องของหน่วยความจำที่ใช้ในการเก็บสำรองค่าของ ω_N^{kn} สำหรับทุกๆค่าของ k และ n ซึ่งมีจำนวนมากตามค่าของ N รวมทั้งยังต้องทำการเผื่อเนื้อที่หน่วยความจำสำหรับลำดับข้อมูลนำเข้า และลำดับข้อมูลผลลัพธ์อีกจำนวนหนึ่งด้วย ทำให้จำเป็นต้องใช้หน่วยความจำที่มีขนาดใหญ่มากสำหรับกรณีที่ค่า N มีขนาดใหญ่ๆ ซึ่งในบางครั้งเกินความสามารถที่เครื่องคอมพิวเตอร์จะสามารถทำงานได้ และอีกปัญหาหนึ่ง คือ เวลาที่สูญเสียไปในการรับส่งข้อมูลระหว่างหน่วยประมวลผล และหน่วยความจำในการคำนวณการแปลงฟูเรียร์ที่จะ

เพิ่มขึ้นตามจำนวนครั้งของการคูณและการบวก จึงทำให้มีการศึกษาค้นคว้าเพื่อที่จะหาทางในการแก้ไขปัญหาเหล่านี้ [12] [15] [17] [18] [20]

โดยในปี ค.ศ. 1965 Cooley และ Tukey ได้นำเสนอวิธีการในการแปลงฟูเรียร์แบบไม่ต่อเนื่องแบบใหม่ที่มีประสิทธิภาพในด้านความเร็ว ที่มีชื่อเฉพาะว่า การแปลงฟูเรียร์แบบเร็ว (Fast Fourier Transform) ซึ่งถูกนำไปเป็นเทคนิคใช้ในการคำนวณงานด้านต่างๆ เช่น การวิเคราะห์สเปกตรัมเชิงเลข หรือการทำคอนโวลูชันของเครื่องกรอง [24] โดยวัตถุประสงค์ของขั้นตอนวิธีนี้ก็คือต้องการที่จะลดปริมาณครั้งของการคูณของการแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform)ให้น้อยกว่า $O(n^2)$ ครั้ง โดยพยายามให้ปริมาณครั้งของการคูณอยู่ในรูป $O(n \log n)$ ซึ่งจากสมการของการแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform) จะพบว่าจำนวนครั้งของการคูณจะขึ้นอยู่กับการยกกำลังของ $e^{-i(2\pi / N / 2)\cdot kn}$ เมื่อ n มีค่าตั้งแต่ 0 ไปจนถึง N-1 [21]

โดยวิธีการที่ใช้ในการลดปริมาณการคูณนั้น ทำได้โดยการแบ่งพหุนาม (Polynomial) ทางขวามือของสมการออกเป็นส่วนย่อยๆ และใช้ความรู้เกี่ยวกับทฤษฎีของ ω_N โดยใช้ทฤษฎีย่อยที่ 1.5 ที่ว่า $(\omega_N^k)^2 = \omega_{N/2}^k$ มาช่วยลดจำนวนการคูณสำหรับการยกกำลังของ $e^{-i(2\pi / N / 2)\cdot kn}$ ซึ่งการแปลงฟูเรียร์แบบเร็วมีขั้นตอนการคำนวณ ดังนี้

ให้ลำดับของจำนวนเป็น x = [x[0] , x[1] ,..., x[n-1]] และลำดับของการแปลงฟูเรียร์แบบไม่ต่อเนื่อง เป็น Y = [Y[0] , Y[1] ,..., Y[n-1]] จะได้รูปสมการการแปลงฟูเรียร์แบบไม่ต่อเนื่อง ดังนี้

$$Y[i] = \sum_{k=0}^{n-1} X[k] \times \omega_n^k \quad \text{, เมื่อ} \quad 0 \le i < n$$

โดยที่ $\omega_n = e^{2\pi i / n}$

$$\begin{aligned} Y[i] &= \sum_{k=0}^{(n/2)-1} X[2k] \times \omega_n^{2ki} + \sum_{k=0}^{(n/2)-1} X[2k+1] \times \omega_n^{(2i+1)i} \\ &= \sum_{k=0}^{(n/2)-1} X[2k] \times e^{2(2\pi\sqrt{-1}/n)ki} + \sum_{k=0}^{(n/2)-1} X[2k+1] \times \omega_n^i e^{2(2\pi\sqrt{-1}/n)ki} \\ &= \sum_{k=0}^{(n/2)-1} X[2k] \times e^{2\pi\sqrt{-1}ki/(n/2)} + \omega_n^i \sum_{k=0}^{(n/2)-1} X[2k+1] \times e^{2\pi\sqrt{-1}ki/(n/2)} \end{aligned}$$

กำหนดให้ $\tilde{\omega} = e^{2\pi\sqrt{-1}/(n/2)} = \omega_n^2$

$$Y[i] = \sum_{k=0}^{(n/2)-1} X[2k]\times\tilde{\omega}^{ki} + \omega_n^i \sum_{k=0}^{(n/2)-1} X[2k+1]\times\tilde{\omega}^{ki}$$

$$= X_{even} + \omega_n^i X_{ood}$$

สำหรับ i = 0,1,2, ... N-1 , เมื่อ X_{even} คือ $N/2$ จุด DFT ของจำนวนคู่ เช่น $x_0, x_2, x_4, \ldots$ และ X_{ood} คือ $N/2$ จุด DFT ของจำนวนคี่ เช่น $x_1, x_3, x_5, \ldots$

ถ้าให้ i มีช่วงเป็นจาก 0,1,2, ... $N/2$-1 จะได้ลำดับเป็น 2 ส่วนดังนี้

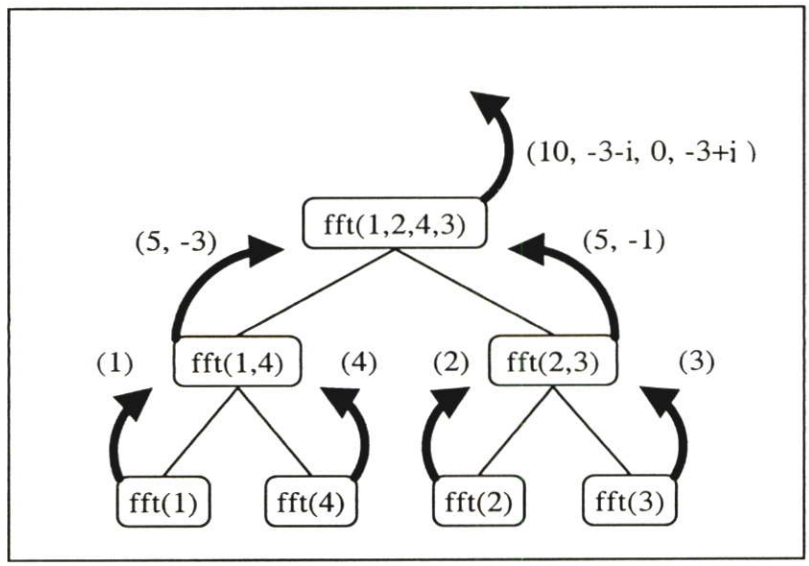
$$Y[i] = X_{even} + \omega_n^i X_{ood}$$

และ $$Y[i+N/2] = X_{even} + \omega_n^{i+N/2} X_{ood} = X_{even} - \omega_n^i X_{ood}$$

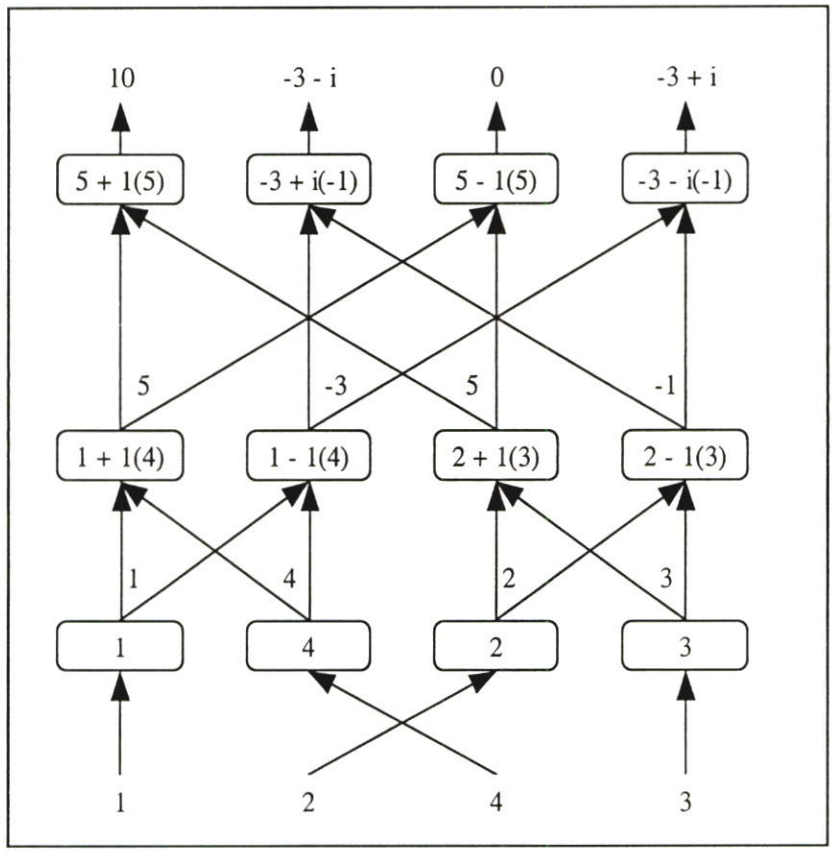
จากทฤษฎีบทย่อยที่ 1.4 $\omega_N^{i+N/2} = -\omega_N^i$ เมื่อ $0 \le i < N/2$
ซึ่งเราจะคำนวณค่า X_i และ $X_{i+N/2}$ โดยใช้ 2 N/2 จุดของ DFT

โดยแต่ละจุด N/2 ของ DFT นั้นสามารถที่จะแบ่งย่อยไปเป็น 2 N/4 จุด ของ DFT ได้อีก และเมื่อทำการแบ่งไปเรื่อยๆจนเหลือเพียงจุดเดียว ก็จะทำการแปลงข้อมูลเสร็จ [10] [20]

เช่น จากตัวอย่างที่ (2.1) ถ้าใช้วิธี Recursive จะมีขั้นตอนดังรูปที่ 2.4 (a) และ 2.4 (b)



(a)



(b)

รูปที่ 2.4 (a) และ 2.4 (b) แสดงการขั้นตอนคำนวณการแปลงฟูเรียร์โดยวิธี Recursive

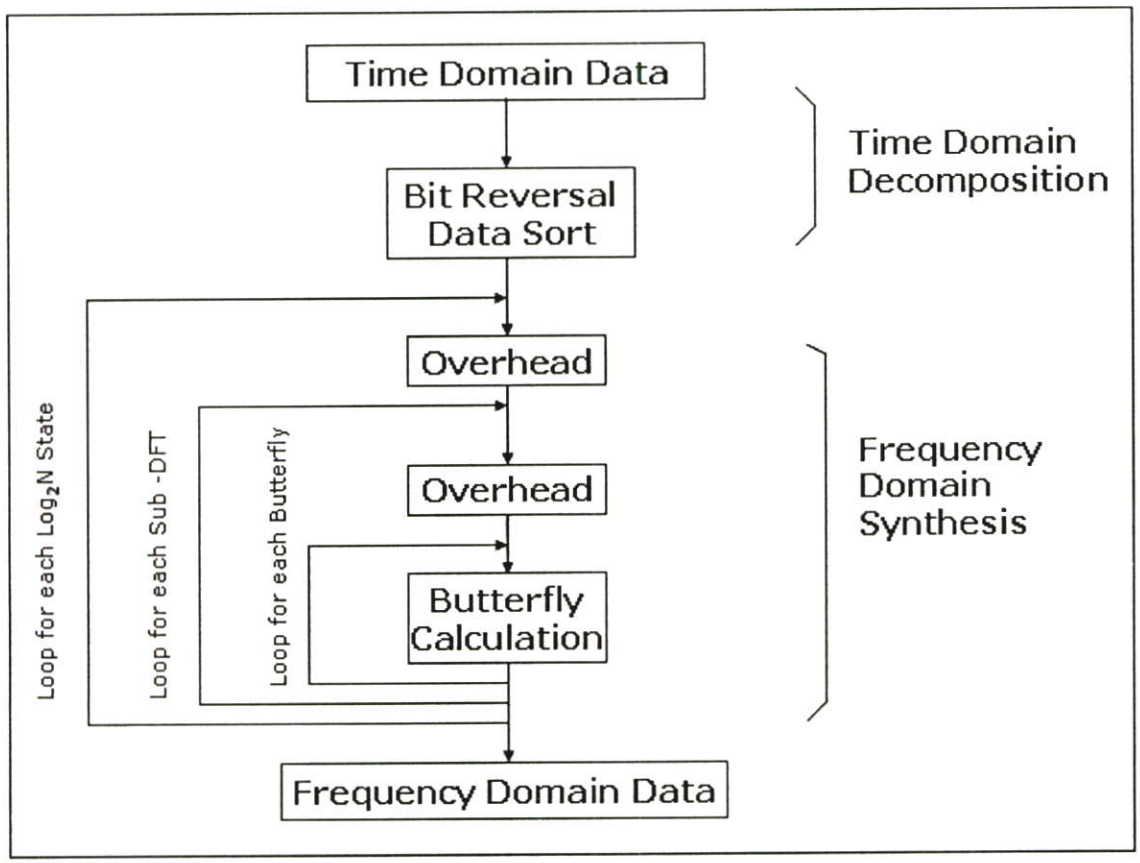
```
RECURSIVE.FFT (a,n)

Parameter   n                  {Number of element in array a}
            a [0,...,(n-1)]    {Coefficients of polynomial to be evaluate}
Local       ω_n                {Complex n th root of unity}
            ω                  {Evaluate polynomial here}
            a^[0]              {Even-numbered coefficients}
            a^[1]              {Odd-numbered coefficients}
            y                  {Result Tranform}
            y^[0]              {Result of FFT of a^[0]}
            y^[1]              {Result of FFT of a^[1]}

Begin
    if n = 1 then return a
    else
      ω_n ← e^(2πi/n)
      ω   ← 1
      a^[0] ← (a[0],a[2],...,a[n-2])
      a^[1] ← (a[1],a[3],...,a[n-1])
      y^[0] ← RECURSIVE.FFT( a^[0] )
                  y^[1] ← RECURSIVE.FFT( a^[1] )
      for k ← 0 to n/2 – 1 do
        y[ k ]     ← y^[0][k] + ω y^[1][k]
        y[ k+n/2 ] ← y^[0][k] - ω y^[1][k]
        ω ← ω × ω_n
      endfor k
      return y
    endif
end
```

รูปที่ 2.5 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบ Recursive

การแปลงฟูเรียร์แบบเร็วแบบลำดับนั้นสามารถจัดการได้อีกแบบโดยจะอยู่ในรูปการทำซ้ำ (Iterative Form) ซึ่งวิธีนี้เป็นวิธีที่เหมาะในการนำไปพัฒนาทางด้านการประมวลผลแบบขนาน ซึ่งวิธีนี้มีการทำงานดังขั้นตอนดังต่อไปนี้ [17] [18] [20]

ให้ a แทนลำดับข้อมูลนำเข้า ที่มีขนาด n ตัวและ A แทนลำดับข้อมูลนำเข้าที่ถูกทำการสลับบิต (Reverse Bit) แล้วโดยขั้นตอนแรกจะทำการแปลงข้อมูลนำเข้า a ให้อยู่ในรูปอันดับบิตผกผัน (Bit Reversed order) ซึ่งแทนด้วย A แล้วจะทำการคำนวณการแปลงฟูเรียร์ ซึ่งจะทำงานทั้งหมด 3 รอบด้วยกัน คือ รอบนอกสุดทำการคำนวณค่าของ ω โดยจะทำทั้งหมด log N รอบ

ส่วนรอบในทั้ง 2 รอบจะเป็นการคำนวณค่า DFT ย่อยของแต่ละส่วน ซึ่งจะมีการติดต่อกันแบบ Butterfly และทำการเก็บค่าของการแปลงฟูเรียร์ ดังรูป 2.6



รูปที่ 2.6 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็วแบบ Iterative

```
ITERATIVE.FFT (a,n)

Parameter   n                 {Number of element in array a}
            a [0,...,(n-1)]   {Coefficients of polynomial to be evaluate}
Local       A [0,...,(n-1)]   {Coefficients in bitreverse order}
            j                 {Iterate through lower half of butterfly}
            k                 {Iterate through indices with same relation
                               position}
            m                 {Current butterfly width}
            s                 {Iterate number}
            t,u               {Temporaries}
            ω_m               {Complex m th root of unity}
            ω                 {Evaluate polynomial here}

Begin
      REVERSE.INDEX.BITS.AND.COPY( a,A )
for s ← 1 to log(n) do
   m ← 2^s
   ω_m ← e^(2πi/m)
   ω ← 1
     for j ← 0 to m/2 -1 do
       for k ← j to n -1 step m do
         t ← ω × A[k + m/2]
         u ← A[ k ]
         A[ k ] ← u + t
         A[ k + m/2 ] ← u - t
       endfor k
       ω ← ω × ω_m
     endfor j
 endfor s
end
```

รูปที่ 2.7 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบ Iterative

2.4 การแปลงฟูเรียร์แบบไม่ต่อเนื่อง 2 มิติ (2-D Discrete Fourier Transform)

การแปลงฟูเรียร์แบบไม่ต่อเนื่อง 1 มิติส่วนมากจะใช้ในงานทางด้านไฟฟ้าและในสาขาที่เกี่ยวข้อง โดยส่วนมากจะเป็นเรื่องราวของความสัมพันธ์ระหว่างตัวแปรทางเวลา (t) และตัวแปรทางความถี่ (f หรือ ω) แต่นอกจากการแปลงฟูเรียร์แบบไม่ต่อเนื่อง 1 มิติ แล้วการแปลงฟูเรียร์แบบไม่ต่อเนื่อง 2 มิติ ก็สามารถนำไปใช้ประโยชน์ได้อย่างมากมาย โดยเฉพาะในงานทางด้านการประมวลผลสัญญาณดิจิทัล (Digital Signal Processing) หรืองานทางด้านการประมวลผลรูปภาพดิจิทัล (Digital Image Processing) ซึ่งวิธีการแปลงฟูเรียร์แบบไม่ต่อเนื่อง 2 มิติจะมีนิยามที่คล้ายคลึงกับการแปลงฟูเรียร์แบบไม่ต่อเนื่อง 1 มิติ [17] [20] [24] ดังนี้

จากนิยามเราจะทำการแปลงฟูเรียร์โดยใช้วิธีการของเมตริกซ์สี่เหลี่ยม

ให้ $W_N = \{\omega_n(i,j); 0 \le i, j < N\}$ เป็นเมตริกซ์สมมาตรที่มีขนาด N × N เมตริกซ์ โดยให้

$$\omega_n(i,j) \equiv \omega_N^{ij} \tag{2.3}$$

จากสมการ (2.3) สามารถแปลงสมการ (2.1) ได้ใหม่ดังนี้ :

$$y(p,q) = \sum_{s=0}^{N-1}\left(\sum_{r=0}^{N-1}\omega_N(p,r)\cdot x(r,s)\right)\cdot\omega_N(s,q) \tag{2.4 a}$$

หรือ

$$y(p,q) = \sum_{s=0}^{N-1}\omega_N(p,r)\left(\sum_{r=0}^{N-1}x(r,s)\cdot\omega_N(s,q)\right) \tag{2.4 b}$$

จากสมการ (2.4 a และ 2.4 b) สามารถแปลงให้อยู่ในรูปเมตริกซ์ดังนี้ :

$$\begin{bmatrix} y(p,0) \\ \dots \\ y(p,N-1) \end{bmatrix}^T =$$

$$\left(\begin{bmatrix} \omega_N(p,0) \\ \dots \\ \omega_N(p,N-1) \end{bmatrix}^T \times \begin{bmatrix} x(0,0) & \dots & x(0,N-1) \\ \dots & \dots & \dots \\ x(N-1,0) & \dots & x(N-1,N-1) \end{bmatrix}\right) \times \begin{bmatrix} \omega_N(0,0) & \dots & \omega_N(0,N-1) \\ \dots & \dots & \dots \\ \omega_N(N-1,0) & \dots & \omega_N(N-1,N-1) \end{bmatrix}$$

สำหรับค่า p = 0,1,2, ... , *N*-1 (2.5 a)

$$\begin{bmatrix} y(0,q) \\ \dots \\ y(N-1,q) \end{bmatrix} =$$

$$\begin{bmatrix} \omega_N(0,0) & \dots & \omega_N(0,N-1) \\ \dots & \dots & \dots \\ \omega_N(N-1,0) & \dots & \omega_N(N-1,N-1) \end{bmatrix} \times \left(\begin{bmatrix} x(0,0) & \dots & x(0,N-1) \\ \dots & \dots & \dots \\ x(N-1,0) & \dots & x(N-1,N-1) \end{bmatrix} \times \begin{bmatrix} \omega_N(0,q) \\ \dots \\ \omega_N(N-1,q) \end{bmatrix}\right)$$

สำหรับค่า q = 0,1,2, ... , *N*-1 (2.5 b)

ดังนั้น ถ้าให้

y_p^R และ $w_{N_p}^R$ แทน แถวที่ p ของ Y และ W_N ตามลำดับ

y_q^C และ $w_{N_q}^C$ แทน แถวที่ q ของ Y และ W_N ตามลำดับ

และสามารถเขียนสมการได้ใหม่เป็น

$$y_p^R = \left(w_{N_p}^R \times X\right) \times W_N \qquad (2.5'a)$$

$$y_q^R = W_N \times \left(X \times w_{N_q}^C\right) \qquad (2.5'b)$$

วิธีการในการแปลงฟูเรียร์แบบเร็ว 2 มิตินั้น สามารถนำเอาวิธีการแปลงฟูเรียร์แบบเร็ว (Fast Fourier Transform) ของข้อมูลแบบ 1มิติมาใช้เพื่อเพิ่มประสิทธิภาพในการคำนวณการแปลงฟูเรียร์แบบไม่ต่อเนื่อง 2 มิติได้เช่นเดียวกัน ซึ่งมีผู้ศึกษาไว้แล้ว 2 ขั้นตอนวิธี คือ ขั้นวิธีแบบแถวและหลัก (Row-Column Fast Fourier Transform) และขั้นตอนวิธีเวกเตอร์เรดิกซ์ (Vector Radix Fast Fourier Transform)

2.4.1 การแปลงฟูเรียร์แบบเร็ว 2 มิติโดยวิธีแถวและหลัก

(Row-Coloumn Two-Dimensional Fast Fourier Transform)

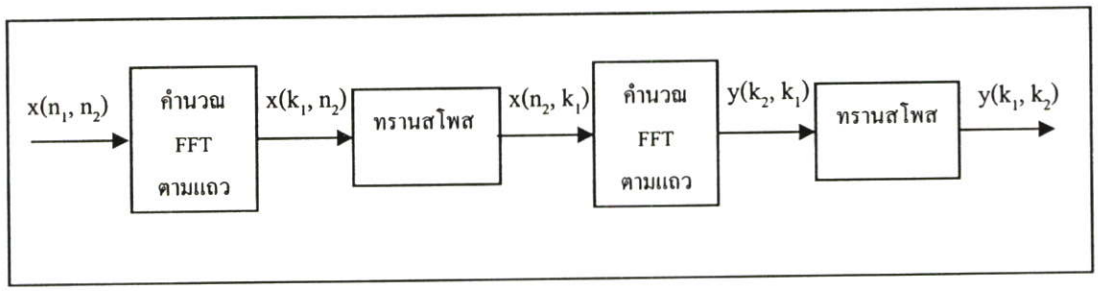
ขั้นตอนวิธีแถวและหลัก (Row - Coloumn Algorithm) เป็นขั้นตอนวิธีพื้นฐานที่ใช้ในการคำนวณการแปลงฟูเรียร์แบบไม่ต่อเนื่อง 2 มิติโดยจะใช้วิธีการแปลงฟูเรียร์แบบเร็ว 1 มิติเข้ามาช่วยในการคำนวณซึ่งจะทำให้สามารถคำนวณได้เร็วกว่าการคำนวณโดยตรง ดังนั้น ขั้นตอนวิธีแถวและหลัก จึงเป็นวิธีการที่ใช้มากที่สุดในการคำนวณการแปลงฟูเรียร์แบบเร็ว 2 มิติ [2] [8] [12] [15] [20]

ให้ $X = \{x(n_1, n_2); 0 \le n_1, n_2 \le N\}$ อยู่ในรูป $N \times N$ เมตริกซ์ และ 2D DFT ของ $N \times N$ เมตริกซ์ $Y = \{y(k_1, k_2); 0 \le k_1, k_2 \le N\}$

$$y(k_1, k_2) = \sum_{r=0}^{N-1} \sum_{s=0}^{N-1} x(n_1, n_2) \cdot \omega_N^{k_1 n_1} \cdot \omega_N^{k_2 n_2} \qquad (2.6)$$

เมื่อ ω_N คือ

$$\omega_N = e^{\frac{-2j\pi}{N}} \qquad (2.7)$$



รูปที่ 2.8 แสดงขั้นตอนวิธีแถวและหลัก (Row-Coloumn Algorithm)

โดยมีขั้นตอนในการคำนวณขั้นตอนวิธีพื้นฐานดังนี้

1) ทำการคำนวณการแปลงฟูเรียร์ 1 มิติของแต่ละแถว(Row)
2) ทำการทรานสโพสข้อมูล
3) ทำการคำนวณการแปลงฟูเรียร์ 1 มิติของแต่ละแถว หรือ หลัก (Column) ของข้อมูลที่ถูกทรานสโพสแล้ว
4) ทำการทรานสโพสข้อมูล

<u>ตัวอย่างที่ 2.2</u> แสดงการแปลงฟูเรียร์แบบเร็ว 2 มิติด้วยขั้นตอนวิธีแถวและหลัก (Row – Column Algorithm)

ให้ข้อมูลนำเข้าอยู่ในรูปเมตริกซ์ขนาด 4×4 โดยมีข้อมูลดังนี้

$$A_{rc} = \begin{bmatrix} 1 & -1 & 2 & 3 \\ 4 & 2 & 1 & -2 \\ 2 & 1 & -1 & 2 \\ 4 & 1 & 3 & -1 \end{bmatrix}$$

ขั้นตอนที่ 1 จะทำการแปลงฟูเรียร์แบบเร็วของข้อมูลนำเข้าแต่ละแถวด้วยวิธีการแปลงฟูเรียร์แบบเร็ว 1 มิติ ดังนี้

รอบที่ 1 ทำการแปลงข้อมูลแถวที่ 1 ซึ่งมีสมาชิกดังนี้

a1 = [1 -1 2 3]

เมื่อทำการแปลงข้อมูลตามวิธีการแปลงฟูเรียร์แบบเร็ว 1 มิติ แล้วจะได้ผลลัพธ์เป็น

A1 = [5 -1 + 4i 5 3 + 4i]

รอบที่ 2 ทำการแปลงข้อมูลแถวที่ 2 เหมือนกับแถวที่ 1 แล้วจะได้ผลลัพธ์เป็น

A2 = [4 3 - 4i 5 3 + 4i]

รอบที่ 3 ทำการแปลงข้อมูลแถวที่ 2 เหมือนกับแถวที่ 1 แล้วจะได้ผลลัพธ์เป็น

A3 = [1 3 + i -2 3 - i]

รอบที่ 4 ทำการแปลงข้อมูลแถวที่ 2 เหมือนกับแถวที่ 1 แล้วจะได้ผลลัพธ์เป็น

A4 = [7 1 – 2i 7 1 + 2i]

เมื่อทำการแปลงฟูเรียร์แบบเร็วของข้อมูลนำเข้าครบทุกแถวแล้วจะได้ข้อมูลเป็นดังนี้

$$A_{Rc} = \begin{bmatrix} 5 & -1+4i & 5 & 3+4i \\ 4 & 3-4i & 5 & 3+4i \\ 1 & 3+i & -2 & 3-i \\ 7 & 1-2i & 7 & 1+2i \end{bmatrix}$$

ขั้นตอนที่ 2 จะทำการทรานสโพสของข้อมูลที่ได้ทำการแปลงฟูเรียร์แบบเร็วตามแถวแล้ว ซึ่งจะทำให้ได้ข้อมูลดังนี้

$$A_{cR} = \begin{bmatrix} 5 & 4 & 1 & 7 \\ -1+4i & 3-4i & 3+i & 1-2i \\ 5 & 5 & -2 & 7 \\ 3+4i & 3+4i & 3-i & 1+2i \end{bmatrix}$$

ขั้นตอนที่ 3 จะทำการแปลงฟูเรียร์แบบเร็วของข้อมูลนำเข้าที่ทำการทรานสโพสแล้วแต่ละแถวด้วยวิธีการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติ เหมือนกับขั้นตอนที่ 1

โดยจะจะได้ข้อมูลเป็น

$$A_{CR} = \begin{bmatrix} 21 & 1+2i & -3 & 1-2i \\ 6-i & -6+i & -2+11i & -2+5i \\ 11 & 3+2i & -13 & 3-2i \\ 6+i & -2-5i & -2-11i & -6-i \end{bmatrix}$$

ขั้นตอนที่ 4 จะทำการทรานสโพสของข้อมูลที่ได้ทำการแปลงฟูเรียร์แบบเร็วตามแถวแล้ว ซึ่งจะทำให้ได้ข้อมูลดังนี้

$$A_{RC} = \begin{bmatrix} 21 & 6-i & 11 & 6+i \\ 1+2i & -6+i & 3+2i & -2-5i \\ -3 & -2+11i & -13 & -2-11i \\ 1-2i & -2+5i & 3-2i & -6-i \end{bmatrix}$$

2.4.2 การแปลงฟูเรียร์แบบเร็ว 2 มิติโดยวิธีเวกเตอร์เรดิกซ์ (Vector Radix Fast Fourier Transform)

นอกจากขั้นตอนวิธีแถวและหลักที่ใช้ในการคำนวณการแปลงฟูเรียร์แบบ 2 มิติแล้วยังมีผู้นำเสนอขั้นตอนวิธีในการคำนวณการแปลงฟูเรียร์แบบ 2 มิติอีกวิธี นั่นคือ ขั้นตอนวิธีเวกเตอร์เรดิกซ์ (Vector Radix Algorithm) ซึ่งขั้นตอนวิธีเวกเตอร์เรดิกซ์จะมีวิธีการคล้ายกับขั้นตอนวิธีพื้นฐานของการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติ ที่ทำการแบ่งผลรวมออกเป็น 2 ส่วน คือ ส่วนที่เป็นกลุ่มคู่ และกลุ่มคี่ แต่ขั้นตอนวิธีเวกเตอร์เรดิกซ์นั้นจะใช้กับข้อมูลที่มีขนาดหลายมิติ เช่น ข้อมูลที่มี 2 มิติก็จะต้องทำการแบ่งผลรวมออกเป็น 4 ส่วนด้วยกัน ซึ่งก็จะได้ดังนี้ [2] [7] [12]

ให้ $X = \{x(n_1, n_2); 0 \le n_1, n_2 \le N\}$ อยู่ในรูป $N \times N$ เมตริกซ์ และ 2-D DFT ของ $N \times N$ เมตริกซ์ $Y = \{y(k_1, k_2); 0 \le k_1, k_2 \le N\}$

$$y(k_1,k_2) = \sum_{r=0}^{N-1}\sum_{s=0}^{N-1} x(n_1,n_2)\cdot \omega_N^{k_1 n_1}\cdot \omega_N^{k_2 n_2} \tag{2.8}$$

เมื่อ ω_N คือ

$$\omega_N = e^{\frac{-2j\pi}{N}} \tag{2.9}$$

เมื่อแบ่งเป็นผลรวมย่อย 4 ส่วนจะได้ดังนี้

$$\begin{aligned} y(k_1,k_2) &= \sum_{r=0}^{N-1}\sum_{s=0}^{N-1} x(n_1,n_2)\cdot \omega_N^{k_1 n_1}\cdot \omega_N^{k_2 n_2} \\ &= \sum_{n_1:even}\sum_{n_2:even} x(n_1,n_2)\cdot \omega_N^{k_1 n_1}\cdot \omega_N^{k_2 n_2} + \sum_{n_1:odd}\sum_{n_2:even} x(n_1,n_2)\cdot \omega_N^{k_1 n_1}\cdot \omega_N^{k_2 n_2} + \\ &= \sum_{n_1:even}\sum_{n_2:odd} x(n_1,n_2)\cdot \omega_N^{k_1 n_1}\cdot \omega_N^{k_2 n_2} + \sum_{n_1:odd}\sum_{n_2:odd} x(n_1,n_2)\cdot \omega_N^{k_1 n_1}\cdot \omega_N^{k_2 n_2} \\ &= G_{ee}(k_1,k_2) + G_{oe}(k_1,k_2)\cdot \omega_N^{k_1} + G_{eo}(k_1,k_2)\cdot \omega_N^{k_2} + G_{oo}(k_1,k_2)\cdot \omega_N^{k_1+2} \end{aligned}$$

เมื่อ $G_{ee}(k_1,k_2), G_{oe}(k_1,k_2), G_{eo}(k_1,k_2), G_{oo}(k_1,k_2)$ คือ $N/2 \times N/2$ – point DFT ของ $g_{ee}(n_1,n_2), g_{oe}(n_1,n_2), g_{eo}(n_1,n_2), g_{oo}(n_1,n_2)$ ตามลำดับ

โดยมีขั้นตอนการทำงานดังนี้

1. ในการแบ่งชุดข้อมูลออกเป็น 4 กลุ่ม คือ กลุ่มข้อมูลแถวคู่หลักคู่ กลุ่มข้อมูลแถวคู่หลักคี่ กลุ่มข้อมูลแถวคี่หลักคู่ กลุ่มข้อมูลแถวคี่หลักคี่ จะใช้วิธีการสลับบิต (Reverse bit Order) ทั้งทางแถวและหลักของข้อมูล เพื่อทำให้ได้ข้อมูลในขั้นสุดท้าย
2. ทำการแบ่งข้อมูลออกเป็น 4 ส่วนโดยแต่ละส่วนจะได้ข้อมูลขนาด N/2 × N/2 หลังจากนั้นทำการแบ่งข้อมูลขนาด N/2 × N/2 แต่ละส่วนออกเป็น 4 ส่วนอีกโดยแต่ละส่วนของข้อมูลขนาด N/2 × N/2 จะได้ข้อมูลขนาด N/4 × N/4 อีก 4 ส่วนแล้วทำการแบ่งแบบเดิมไปเรื่อยๆ จนกระทั่งได้ข้อมูลขนาด 2 × 2 จึงเริ่มทำการคำนวณการแปลงฟูเรียร์ขั้นตอนวิธีเวกเตอร์เรดิกซ์
3. ทำการคำนวณด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ โดยมีขั้นตอนดังนี้
   3.1 เมื่อแบ่งจนได้ข้อมูลขนาด 2 × 2 แล้ว โดยแทนข้อมูลทั้ง 4 ส่วนด้วยโหนด a , b , c และ d เมื่อแต่ละโหนดแทนข้อมูล

a	b
c	d

   3.2 ทำการคูณแต่ละโหนดด้วยค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อนของข้อมูล 2 มิติดังนี้

   ที่โหนด b ทำการคูณด้วยค่า $\omega_N^{k_2}$

   ที่โหนด c ทำการคูณด้วยค่า $\omega_N^{k_1}$

   ที่โหนด d ทำการคูณด้วยค่า $\omega_N^{k_1+k_2}$

   เมื่อ k_1 และ k_2 คือ ตำแหน่งแถวและหลักของข้อมูลแต่ละโหนด

a	$b\times\omega_N^{k_2}$
$c\times\omega_N^{k_1}$	$d\times\omega_N^{k_1+k_2}$

   3.3 เมื่อทำการคูณแต่ละโหนดด้วยค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อนของข้อมูล 2 มิติแล้วให้แทนข้อมูลแต่ละโหนดใหม่ด้วย A , B , C และ D ดังนี้

A	B
C	D

3.4 ทำการคำนวณขั้นสุดท้ายโดยมีรูปแบบดังนี้

โหนด A จะมีค่าเป็น $A = A + B + C + D$

โหนด B จะมีค่าเป็น $B = A + B - C - D$

โหนด C จะมีค่าเป็น $C = A - B + C - D$

โหนด D จะมีค่าเป็น $D = A - B - C + D$

A + B + C + D	A + B - C - D
A - B + C - D	A - B - C + D

4. ทำการคำนวณข้อมูลชุดอื่นๆด้วยวิธีเดียวกันกับวิธีในข้อ 3 โดยขนาดข้อมูลจะเพิ่มขึ้นเรื่อยจนถึงข้อมูลขนาด N/2 × N/2 เป็นอันเสร็จขั้นตอนการคำนวณด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ โดยในแต่ละโหนด a , b , c และ d ในข้อ 3.1 เราสามารถเขียนค่าของข้อมูลเป็น $y(k_1,k_2)$, $y(k_1+N/2,k_2)$, $y(k_1,k_2+N/2)$, $y(k_1+N/2,k_2+N/2)$ ตามลำดับได้ดังนี้

เมื่อให้

$$A = a + b$$

$$B = a - b$$

$$C = c + d$$

$$D = c - d$$

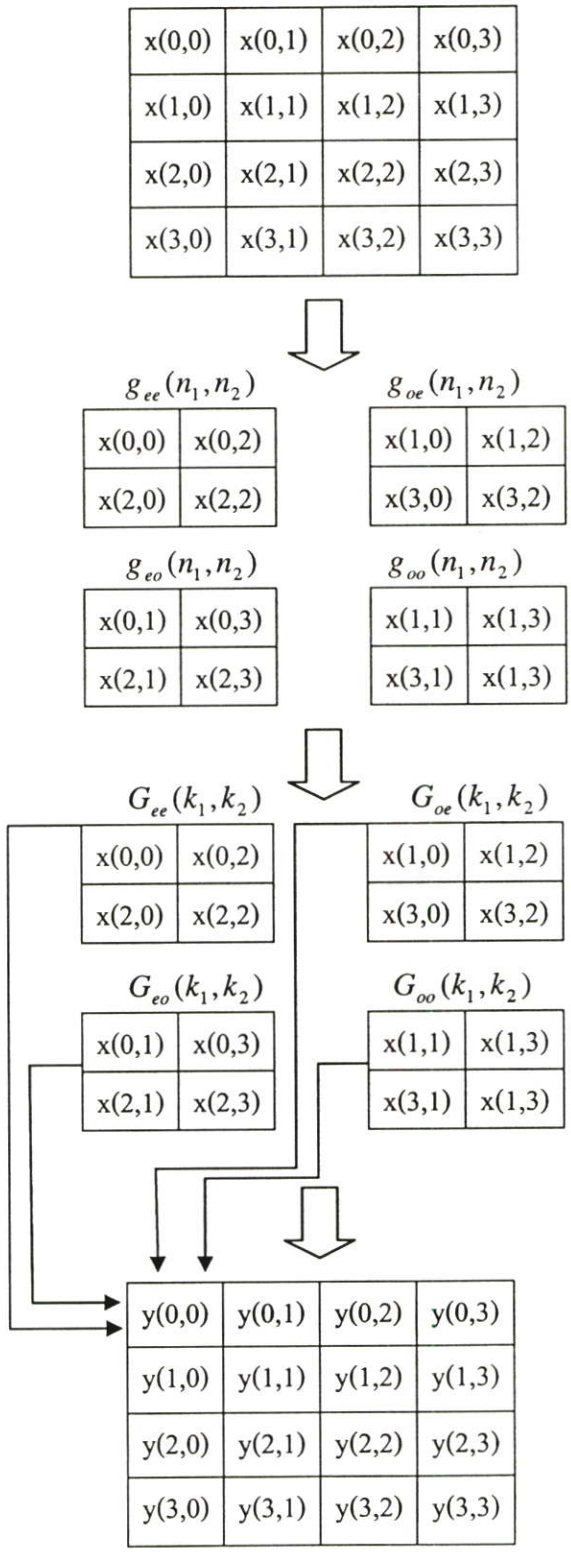
จะได้ว่า

$$y(k_1,k_2) = A + C$$

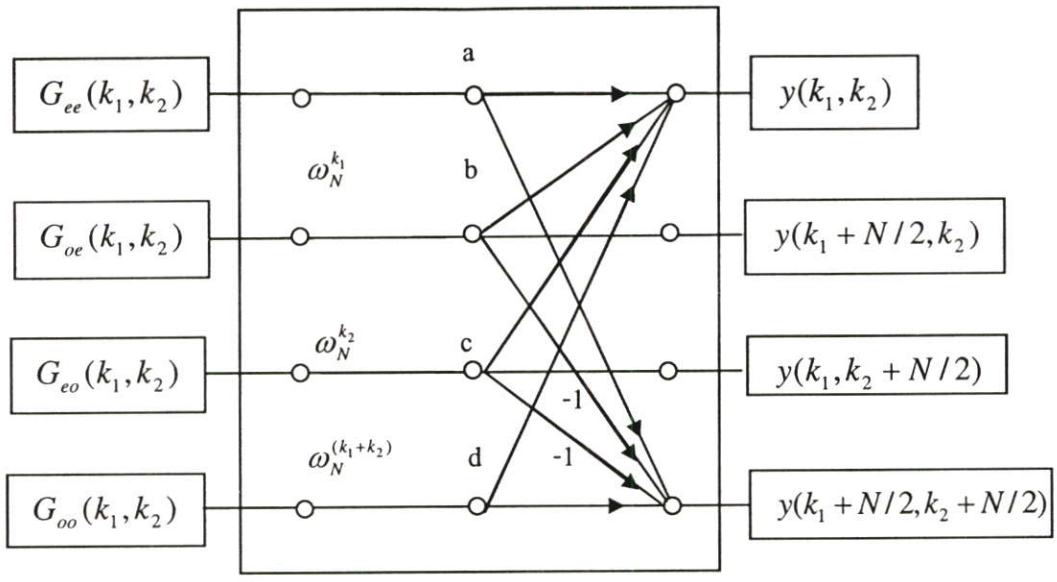
$$y(k_1+N/2,k_2) = B + D$$

$$y(k_1,k_2+N/2) = A - C$$

$$y(k_1+N/2,k_2+N/2) = B - D$$



รูปที่ 2.9 แสดงขั้นตอนการทำงาน 1 ขั้นของขั้นตอนวิธีเวกเตอร์เรดิกซ์ของข้อมูลขนาด 2× 2



รูปที่ 2.10 แสดงขั้นตอนการคำนวณแบบ Butterfly ของขั้นตอนวิธีเวกเตอร์เรดิกซ์

<u>ตัวอย่างที่ 2.3</u> แสดงการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ (Vector- Radix)

ให้ข้อมูลนำเข้าอยู่ในรูปเมตริกซ์ขนาด 4 ×4 โดยมีข้อมูลดังนี้

$$A = \begin{bmatrix} 1 & -1 & 2 & 3 \\ 4 & 2 & 1 & -2 \\ 2 & 1 & -1 & 2 \\ 4 & 1 & 3 & -1 \end{bmatrix}$$

ขั้นตอนที่ 1 เป็นขั้นตอนในการจัดกลุ่มข้อมูลเพื่อทำการการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ โดยมีขั้นตอนในการจัดกลุ่มข้อมูลดังนี้

เมื่อให้แถวแรก คือ แถวที่ 0 และหลักแรก คือ หลักที่ 0 จะได้ว่าข้อมูลจะมี 4 แถว และ4 หลัก โดยเริ่มตำแหน่งแถวและหลักที่ 0 , 1 , 2 และ 3 ตามลำดับ จะทำการแบ่งข้อมูลออกเป็น 4 กลุ่มตามตำแหน่งของข้อมูลของแถวและหลักได้ดังนี้

กลุ่มที่ 1 ข้อมูลที่อยู่ในตำแหน่งแถวที่เป็นเลขคู่ และอยู่ในตำแหน่งหลักที่เป็นเลขคู่ ซึ่งจะได้ข้อมูลดังนี้

$$a_{EE} = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$$

กลุ่มที่ 2 ข้อมูลที่อยู่ในตำแหน่งแถวที่เป็นเลขคู่ และอยู่ในตำแหน่งหลักที่เป็นเลขคี่ ซึ่งจะได้ข้อมูลดังนี้

$$a_{EO} = \begin{bmatrix} -1 & 3 \\ 1 & 2 \end{bmatrix}$$

กลุ่มที่ 3 ข้อมูลที่อยู่ในตำแหน่งแถวที่เป็นเลขคี่ และอยู่ในตำแหน่งหลักที่เป็นเลขคู่ ซึ่งจะได้ข้อมูลดังนี้

$$a_{OE} = \begin{bmatrix} 4 & 1 \\ 4 & 3 \end{bmatrix}$$

กลุ่มที่ 4 ข้อมูลที่อยู่ในตำแหน่งแถวที่เป็นเลขคี่ และอยู่ในตำแหน่งหลักที่เป็นเลขคู่ ซึ่งจะได้ข้อมูลดังนี้

$$a_{OO} = \begin{bmatrix} 2 & -2 \\ 1 & -1 \end{bmatrix}$$

ขั้นตอนที่ 2 เป็นขั้นตอนการคำนวณข้อมูลการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ ซึ่งมีขั้นตอนดังนี้

ให้ $A = \begin{bmatrix} a_{EE} & a_{EO} \\ a_{OE} & a_{OO} \end{bmatrix}$ เมื่อ N คือ ขนาดของมิติของเมตริกซ์

ขั้นตอนที่ 2.1

ทำการคูณค่าของ a ด้วยค่า ω_N ซึ่งจะมีลำดับในการคูณเป็นดังนี้

a_{EE} จะคูณกับค่า ω_N^0

a_{EO} จะคูณกับค่า ω_N^C

a_{OE} จะคูณกับค่า ω_N^R

a_{OO} จะคูณกับค่า ω_N^{R+C}

เมื่อแทน R ด้วยตำแหน่งของแถวของข้อมูล และ C ด้วยตำแหน่งของหลักของข้อมูล

ขั้นตอนที่ 2.2

ทำการรวมข้อมูลที่ได้ทำการคูณด้วยค่า ω_N แล้วโดยมีรูปแบบดังนี้

$$a_{EE} = a_{EE} + a_{EO} + a_{OE} + a_{OO}$$
$$a_{EO} = a_{EE} + a_{EO} - a_{OE} - a_{OO}$$
$$a_{OE} = a_{EE} - a_{EO} + a_{OE} - a_{OO}$$
$$a_{OO} = a_{EE} - a_{EO} - a_{OE} + a_{OO}$$

เมื่อนำข้อมูลที่ได้ทำการแบ่งเป็น 4 กลุ่มแล้วจากขั้นตอนที่ 1 มาทำการคำนวณข้อมูลในแต่ละกลุ่มด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์จะได้ดังนี้

จากข้อมูลเริ่มต้นจะได้ว่าค่า N มีค่าเท่ากับ 4 ดังนั้น ค่า ω_N จะเป็น ω_4 โดยจะทำการคำนวณข้อมูลทีละกลุ่มดังนี้

กลุ่มที่ 1

มีข้อมูลเป็น $a_{EE} = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$ ขั้นแรกจะทำการคูณด้วยค่า ω_N ตามวิธีการข้างต้นโดยจะได้เป็น

$$a_{EE} = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2\times\omega_4^0 \\ 2\times\omega_4^0 & -1\times\omega_4^0 \end{bmatrix} = \begin{bmatrix} 1 & 2\times1 \\ 2\times1 & -1\times1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$$

ขั้นต่อมาจะทำการทำการรวมข้อมูลตามขั้นตอนวิธีเวกเตอร์เรดิกซ์

$$a_{EE} = \begin{bmatrix} 1+2+2+(-1) & 1+2-2-(-1) \\ 1-2+2-(-1) & 1-2-2+(-1) \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & -4 \end{bmatrix}$$

กลุ่มที่ 2

มีข้อมูลเป็น $a_{EO}=\begin{bmatrix}-1 & 3\\ 1 & 2\end{bmatrix}$ ขั้นแรกจะทำการคูณด้วยค่า ω_N ตามวิธีการข้างต้นโดยจะได้เป็น

$$a_{EO}=\begin{bmatrix}-1 & 3\\ 1 & 2\end{bmatrix}=\begin{bmatrix}-1 & 3\times\omega_4^0\\ 1\times\omega_4^0 & 2\times\omega_4^0\end{bmatrix}=\begin{bmatrix}-1 & 3\times1\\ 1\times1 & 2\times1\end{bmatrix}=\begin{bmatrix}-1 & 3\\ 1 & 2\end{bmatrix}$$

ขั้นต่อมาจะทำการทำการรวมข้อมูลตามขั้นตอนวิธีเวกเตอร์เรดิกซ์

$$a_{EO}=\begin{bmatrix}-1+3+1+2 & -1+3-1-2\\ -1-3+1-2 & -1-3-1+2\end{bmatrix}=\begin{bmatrix}5 & -1\\ -5 & -3\end{bmatrix}$$

กลุ่มที่ 3

มีข้อมูลเป็น $a_{OE}=\begin{bmatrix}4 & 1\\ 4 & 3\end{bmatrix}$ ขั้นแรกจะทำการคูณด้วยค่า ω_N ตามวิธีการข้างต้นโดยจะได้เป็น

$$a_{OE}=\begin{bmatrix}4 & 1\\ 4 & 3\end{bmatrix}=\begin{bmatrix}4 & 1\times\omega_4^0\\ 4\times\omega_4^0 & 3\times\omega_4^0\end{bmatrix}=\begin{bmatrix}4 & 1\times1\\ 4\times1 & 3\times1\end{bmatrix}=\begin{bmatrix}4 & 1\\ 4 & 3\end{bmatrix}$$

ขั้นต่อมาจะทำการทำการรวมข้อมูลตามขั้นตอนวิธีเวกเตอร์เรดิกซ์

$$a_{OE}=\begin{bmatrix}4+1+4+3 & 4+1-4-3\\ 4-1+4-3 & 4-1-4+3\end{bmatrix}=\begin{bmatrix}12 & -2\\ 4 & 2\end{bmatrix}$$

กลุ่มที่ 4

มีข้อมูลเป็น $a_{OO}=\begin{bmatrix}2 & -2\\ 1 & -1\end{bmatrix}$ ขั้นแรกจะทำการคูณด้วยค่า ω_N ตามวิธีการข้างต้นโดยจะได้เป็น

$$a_{OO}=\begin{bmatrix}2 & -2\\ 1 & -1\end{bmatrix}=\begin{bmatrix}2 & -2\times\omega_4^0\\ 1\times\omega_4^0 & -1\times\omega_4^0\end{bmatrix}=\begin{bmatrix}2 & -2\times1\\ 1\times1 & -1\times1\end{bmatrix}=\begin{bmatrix}2 & -2\\ 1 & -1\end{bmatrix}$$

ขั้นต่อมาจะทำการทำการรวมข้อมูลตามขั้นตอนวิธีเวกเตอร์เรดิกซ์

$$a_{OO}=\begin{bmatrix}2+(-2)+1+(-1) & 2+(-2)-1-(-1)\\ 2-(-2)+1-(-1) & 2-(-2)-1+(-1)\end{bmatrix}=\begin{bmatrix}0 & 0\\ 6 & 2\end{bmatrix}$$

เมื่อทำการคำนวณข้อมูลของกลุ่มย่อยเสร็จแล้วก็จะทำการคำนวณข้อมูลของกลุ่มใหญ่ซึ่งมีขนาดเป็น 4 ×4 ตามวิธีการเดิม คือ

$$A=\begin{bmatrix} a_{EE} & a_{EO} \\ a_{OE} & a_{OO} \end{bmatrix}$$

เมื่อ a_{EE} , a_{EO} , a_{OE} และ a_{OO} คือ ข้อมูลขนาด 2 ×2 ที่ได้คำนวณด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์

เมื่อทำการคูณด้วยค่า ω_N จะได้เป็น

$$A=\begin{bmatrix} a_{EE}\times\omega_4^0 & a_{EO}\times\omega_4^C \\ a_{OE}\times\omega_4^R & a_{OO}\times\omega_4^{R+C} \end{bmatrix}$$

ซึ่งจะได้ค่าเป็นดังนี้

$$A=\begin{bmatrix} 4 & 2 & 5 & 5i \\ 2 & 4 & -1 & 3i \\ 12 & 4 & 0 & -6i \\ 2i & -2i & 0 & -2 \end{bmatrix}$$

แล้วทำการรวมข้อมูลที่ได้ทำการคูณด้วยค่า ω_N แล้วโดยมีวิธีเดิม จะได้ผลลัพธ์เป็น

$$A=\begin{bmatrix} 21 & 6-i & 11 & 6+i \\ 1+2i & -6+i & 3+2i & -2-5i \\ -3 & -2+11i & -13 & -2-11i \\ 1-2i & -2+5i & 3-2i & -6-i \end{bmatrix}$$

2.5 การแปลงฟูเรียร์แบบเร็วแบบขนาน (Parallel Fast Fourier Transform)

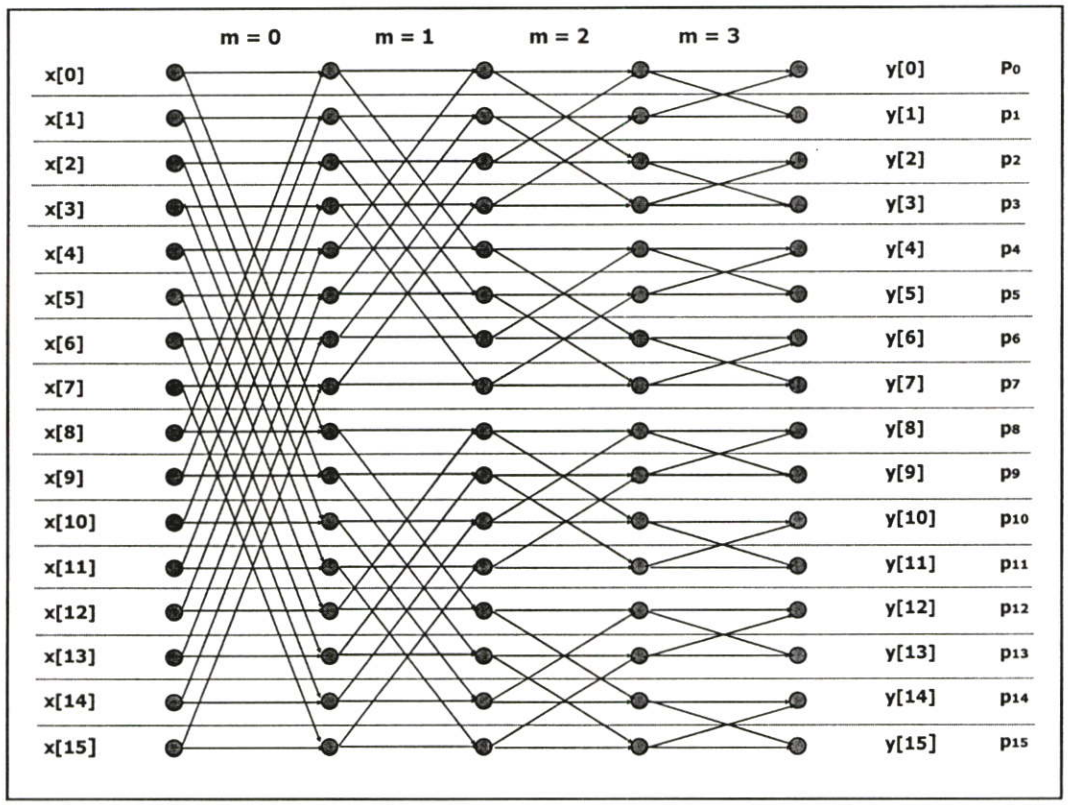
จากการแปลงฟูเรียร์เป็นความรู้ที่เป็นพื้นฐานสำคัญยิ่งต่องานวิจัยและพัฒนาทางเทคโนโลยีหลายสาขา โดยเฉพาะงานทางด้านวิทยาศาสตร์ และทางด้านวิศวกรรมศาสตร์ จึงทำให้มีการพัฒนางานวิจัยที่เกี่ยวข้องกับการแปลงฟูเรียร์อย่างมากมาย แต่ถึงแม้ว่าจะมีการนำอุปกรณ์อิเล็กทรอนิกส์ที่มีสมรรถนะสูงขึ้นอย่างเครื่องคอมพิวเตอร์มาช่วยคำนวณ หรือรวมถึงการพัฒนาขั้นตอนวิธีที่มีอยู่ให้มีประสิทธิภาพมากขึ้น แต่การทำงานในส่วนของการแปลงฟูเรียร์ก็ยังคงมีปัญหาในด้านเวลาอยู่

โดยเฉพาะกับข้อมูลที่มีขนาดใหญ่มาก ทำให้ได้มีแนวความคิดที่จะนำการแปลงฟูเรียร์มาจัดการบนเครื่องคอมพิวเตอร์ที่มีการประมวลผลแบบขนานซึ่งมีความสามารถในการประมวลผลมากกว่าเครื่องคอมพิวเตอร์ธรรมดา เพื่อแก้ปัญหาด้านเวลาที่มากของการคำนวณการแปลงฟูเรียร์ [2] [5] [11] [15] [17] [18] [19] [8] [20]

โดยการแปลงฟูเรียร์แบบเร็วแบบขนานสามารถแยกตามขนาดและโครงสร้างการติดต่อสื่อสารระหว่างหน่วยประมวลผลมีรายละเอียดดังนี้

2.5.1 การแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบไฮเปอร์คิว (Hypercube)

1.) กรณีที่หนึ่งข้อมูลต่อหนึ่งหน่วยประมวลผล



รูปที่ 2.11 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็วแบบขนานมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบไฮเปอร์คิว กรณีที่มีหนึ่งข้อมูลต่อหนึ่งหน่วยประมวลผล

การแปลงฟูเรียร์แบบเร็วแบบขนานโดยมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบไฮเปอร์คิว มี 3 ขั้นตอนดังนี้

ขั้นตอนที่ 1

ทำการจัดเรียงข้อมูลใหม่โดยข้อมูลจะอยู่ในรูป Bit Reverse เช่น

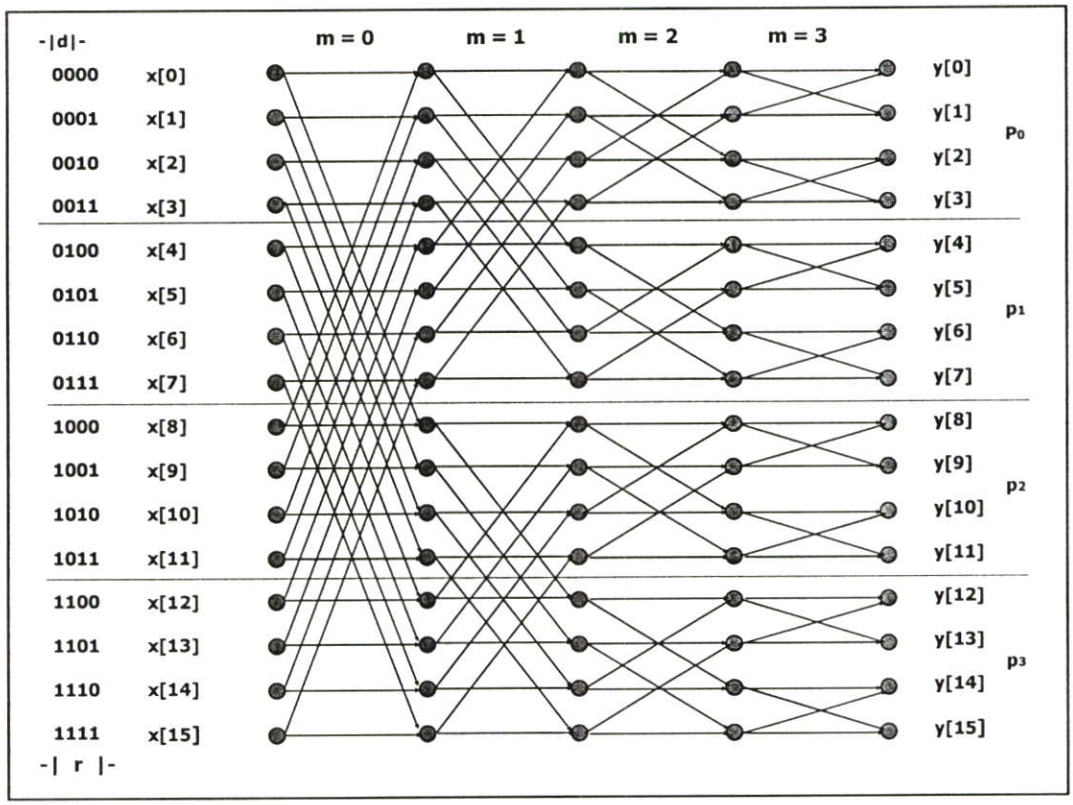
ขั้นตอนที่ 2

ทำการคำนวณการแปลงฟูเรียร์แบบเร็วโดยจะไม่มีการติดต่อกับหน่วยประมวลผลอื่น ซึ่งจะมีการคำนวณทั้งหมด log N – log P รอบ

ขั้นตอนที่ 3

ทำการคำนวณการแปลงฟูเรียร์แบบเร็วโดยจะมีการติดต่อกับหน่วยประมวลผล อื่นซึ่งจะมีการคำนวณทั้งหมด log P รอบ

2.) กรณีที่หลายข้อมูลต่อหนึ่งหน่วยประมวลผล



รูปที่ 2.12 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็วแบบขนานมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบไฮเปอร์คิว กรณีที่มีหลายข้อมูลต่อหนึ่งหน่วยประมวลผล

จากการทำงานของขั้นตอนวิธีของการแปลงฟูเรียร์ด้วยขั้นตอนวิธีทำซ้ำ (Iterative Algorithm) นั้น เราสามารถนำมาพัฒนาบนเครื่องคอมพิวเตอร์ที่มีการติดต่อระหว่างหน่วยประมวลผลแบบไฮเปอร์คิว [17] [18] [20] ได้ดังนี้

ถ้าให้ จำนวนลำดับข้อมูลนำเข้ามีจำนวน n = 16 และ มีจำนวนหน่วยประมวลผลเป็น p = 4 โดยหน่วยประมวลผลจะมีการการแปลงฟูเรียร์แบบเร็วแบบขนานโดยมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบไฮเปอร์คิว มี 3 ขั้นตอนดังนี้

ขั้นตอนที่ 1

ทำการจัดเรียงข้อมูลใหม่โดยข้อมูลจะอยู่ในรูปการสลับบิตข้อมูล (Bit Reverse Order) เช่น

ขั้นตอนที่ 2

ทำการคำนวณการแปลงฟูเรียร์แบบเร็วโดยจะไม่มีการติดต่อกับหน่วยประมวลผลอื่น ซึ่งจะมีการคำนวณทั้งหมด log N – log P รอบ

ขั้นตอนที่ 3

ทำการคำนวณการแปลงฟูเรียร์แบบเร็วโดยจะมีการติดต่อกับหน่วยประมวลผลอื่น ซึ่งจะมีการคำนวณทั้งหมด log P รอบ

ขั้นตอนทางคอมพิวเตอร์การพัฒนาโปรแกรมของการแปลงฟูเรียร์แบบเร็วบนการติดต่อแบบไฮเปอร์คิว ดังแสดงในรูปที่ 2.13

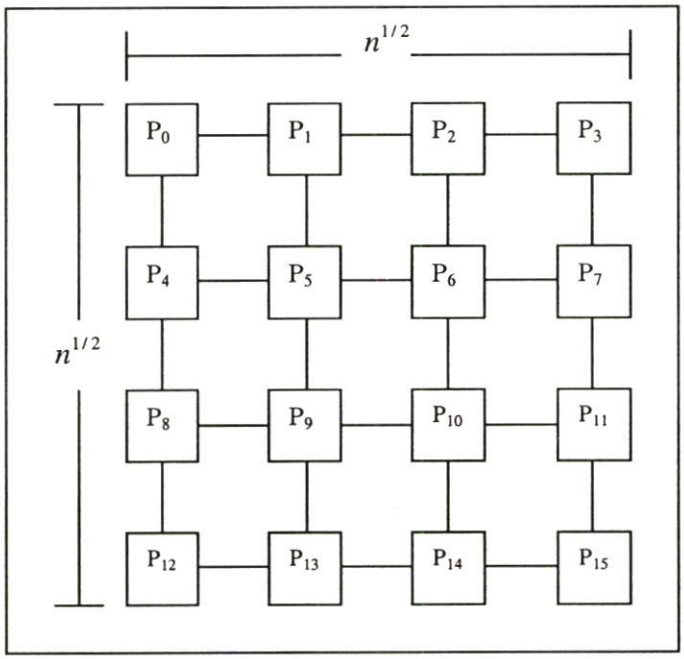
```
FAST FOURIER TRANSFORM (a,y) (HYPERCUBE MULTICOMPUTER)

Global  n                       {Number of sample point}
        P                       {Number of processor}
Local   a [0,...,(n/p-1)]       {Input element}
        pos                     {Power}
        t [0,...,(n/p-1)]       {Temporary values}
        u [0,...,(n/p-1)]       {Temporary values}
        y [0,...,(n/p-1)]       {Output element}
        ω_m                     {Complex m th root of unity}
        ω                       { Power complex m th root of unity }

{Phase1 : Permute input element}

for all P_i, where 0 ≤ i < p do
   for k ← 0 to n/p -1 do
      id ← i × (n/p) + k
      dest.proceor ← REVERSE(id)/(n/p)
      dest.offset ← REVERSE(id) modulo (n/p)
      [dest.proceor] y [dest.offset] ⇐ a[k]
   endfor
endfor.

{Phase2 : Perform iteration not requiring interprocessor communication}

for s ← 1 to log(n/p) do
    m ← 2^s
    ω_m ← e^(2πi/m)
    for all P_i, where 0 ≤ i < p do
       ω ← 1
       for j ← 0 to m/2 -1 do
          for k ← j to n/p -1 step m do
             q ← ω × y[k + m/2]
             r ← y[ k ]
             y[ k ] ← r + q
             y[ k + m/2 ] ← r - q
          endfor k
          ω ← ω × ω_m
       endfor j
    endfor s
endfor
```

รูปที่ 2.13 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบไฮเปอร์คิว

```
{Phase3 : Perform iteration  requiring interprocessor communication}

for s ← 1 to log(p) do
    m ← 2^(s+log(n/p))
    ω_m ← e^(2π/m)
    for all P_i, where 0 ≤ i < p do
      if i / 2^(s-1) is odd then
        pos ← (i × (n/p)) modulo m/2
        ω_m ← e^(2(pos)π/m)
        for j ← 0 to n/p -1 do
          t[j] ← ω × ω_m
        endfor j
      endif
      shift ← 2^(s-1)
      partner ← k ⊗ shift {Exclusive or}
      if i / 2^(s-1) is odd then
        [partner] u ⇐ t
      else
        [partner] u ⇐ y
      endif
endfor s
    for all P_i, where 0 ≤ i < p do
      if i / 2^(s-1) is odd then
        for j ← 0 to n/p -1 do
          y[j] ← u[j] - t[j]
        endfor j
      else
        for j ← 0 to n/p -1 do
          y[j] ← u[j] + t[j]
        endfor j
      endif
    endfor
endfor
```

รูปที่ 2.13 (ต่อ)

2.5.2 การแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบเมสช์ (Mesh)



รูปที่ 2.14 แสดงการติดต่อระหว่างหน่วยประมวลผลแบบเมสช์

การแปลงฟูเรียร์แบบเร็วแบบขนานโดยมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบเมสช์ มี 3 ขั้นตอน [17] [18] [20] ดังนี้

ขั้นตอนที่ 1 แต่ละหน่วยประมวลผล (P_i) ทำการเก็บค่าลำดับนำเข้า(a_k)มาเก็บไว้ที่ตัวแปร c_k

ขั้นตอนที่ 2 ทำการคำนวณการแปลงฟูเรียร์แบบเร็ว

ขั้นตอนที่ 3 ทำการเก็บค่าที่ได้จากการคำนวณการแปลงฟูเรียร์แบบเร็วมาไว้ที่ตัวแปร b_k

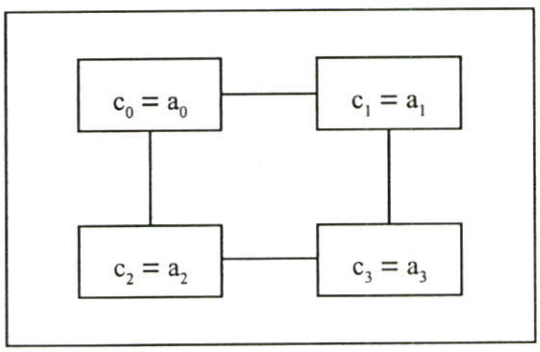
ขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วแบบขนานโดยมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบเมสช์ ดังรูปที่ 2.15

```
Procedure MASH FFT (A,B)

Step 1 :
 for k = 0 to n-1 do in parallel
       c_k ← a_k
  end for k

Step 2 :
 for h = (log n) - 1 downto 0 do
   for k = 0 to n-1 do in parallel
      (2.1) p ← 2^h
      (2.2) q ← n/p
      (2.3) z ← w^p
      (2.4) if ( k mod p ) = ( k mod 2p )
            then (i) c_k ← c_k + c_{k+p} × z^{r(k) mod p}
                 (ii) c_{k+p} ← c_k - c_{k+p} × z^{r(k) mod p}
            end if
    end for k
  end for h

Step 3 :
 for k = 0 to n-1 do in parallel
       b_k ← c_{r(k)}
  end for k
```

รูปที่ 2.15 แสดงขั้นตอนทางคอมพิวเตอร์ของการแปลงฟูเรียร์แบบเร็วบนการติดต่อแบบเมสช์

<u>**ตัวอย่างที่ 2.4**</u> การแปลงฟูเรียร์แบบเร็วแบบขนานขนาด 2x2 โดยมีการติดต่อระหว่างหน่วยประมวลผลเป็นแบบเมสช์จำนวน 2 หน่วยประมวลผล (P=2)

ขั้นตอนที่ 1 แต่ละหน่วยประมวลผล (P_i) ทำการเก็บค่าลำดับนำเข้า (a_k)มาเก็บไว้ที่ตัวแปร c_k ดังรูปที่ 2.1.6



รูปที่ 2.16 แสดงขั้นตอนที่ 1 ในการคำนวณการแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบเมสช์

ขั้นตอนที่ 2 จะทำการคำนวณการแปลงฟูเรียร์แบบเร็วทั้งหมด 2 รอบ (h = 2)
โดยในรอบที่ 1(h = 1) { p=2 , q=2 , z = ω^2 }

ตามเงื่อนไข k mod p = k mod 2p จะคำนวณเฉพาะ k = 0 และ 1 ดังรูปที่ (2.17)
หน่วยประมวลผล P_0 จะคำนวณ

$$c_0 = c_0 + (\omega^2)^0 c_2$$
$$= a_0 + a_2$$

และ

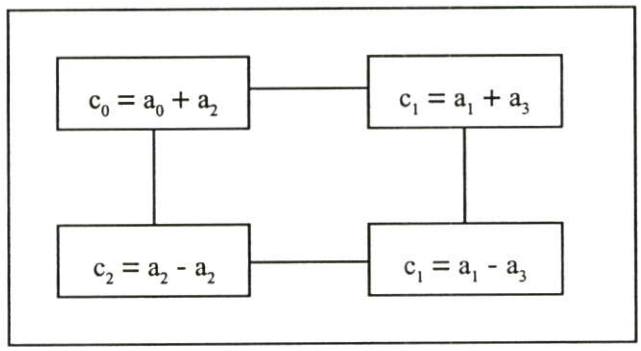
$$c_2 = c_0 - (\omega^2)^0 c_2$$
$$= a_0 - a_2$$

ส่วนหน่วยประมวลผล P_1 จะคำนวณ

$$c_1 = c_1 + (\omega^2)^0 c_3$$
$$= a_1 + a_3$$

และ

$$c_3 = c_1 - (\omega^2)^0 c_3$$
$$= a_1 - a_3$$



รูปที่ 2.17 แสดงขั้นตอนที่ 2 ในการคำนวณการแปลงฟูเรียร์แบบเร็วแบบขนานบนการติดต่อแบบเมสช์

โดยในรอบที่ 2 (h = 2) { p=1 , q=4 , z = ω }

ตามเงื่อนไข k mod p = k mod 2p จะคำนวณเฉพาะ k = 0 และ 2 หน่วยประมวลผล

โดย P_0 จะคำนวณ

$$c_0 = c_0 + \omega^0 c_1$$
$$= a_0 + a_2 + a_1 + a_3$$

และ

$$c_1 = c_0 - \omega^0 c_1$$
$$= a_0 - a_2 - (a_1 + a_3)$$

ส่วนหน่วยประมวลผล P_2 จะคำนวณ

$$c_2 = c_2 + \omega^1 c_3$$
$$= a_0 + a_2 + \omega (a_1 - a_3)$$

และ

$$c_3 = c_2 - \omega^1 c_3$$
$$= a_0 - a_2 - \omega (a_1 - a_3)$$

ขั้นตอนที่ 3 ทำการเก็บค่าที่ได้จากการคำนวณการแปลงฟูเรียร์แบบเร็วมาไว้ที่ตัวแปร b_k

$b_0 = c_0$, $b_1 = c_2$, $b_2 = c_1$ และ $b_3 = c_3$ ดังนี้

$$b_0 = a_0 + a_2 + a_1 + a_3$$

$$b_1 = a_0 + a_2 + \omega a_1 - \omega a_3$$
$$= a_0 + a_2 + \omega^2 a_1 - \omega^3 a_3$$

$$b_2 = a_0 - a_2 - a_1 - a_3$$
$$= a_0 + \omega^2 a_2 + \omega^4 a_1 + \omega 6 a_3$$

$$b_3 = a_0 - a_2 - \omega a_1 + \omega a_3$$
$$= a_0 + \omega^3 a_2 + \omega^6 a_1 + \omega^9 a_3$$

2.6 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน

(Two-Dimensional Parallel Fast Fourier Transform : 2-D PFFT)

การแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ จะใช้วิธีการในการแปลงเช่นเดียวกับแบบ 1 มิติ แต่จะต่างกันที่การแบ่งงานให้แต่ละหน่วยประมวลผลทำงาน ซึ่งจะทำให้เวลาที่ใช้ในการประมวลผลลดลง จึงมีการใช้วิธีการแปลงฟูเรียร์แบบขนาน 2 มิติอย่างกว้างขวางในงานวิจัยต่างๆที่เกี่ยวข้อง [2] [8] [12] [15] [16] [17] [18] [20]

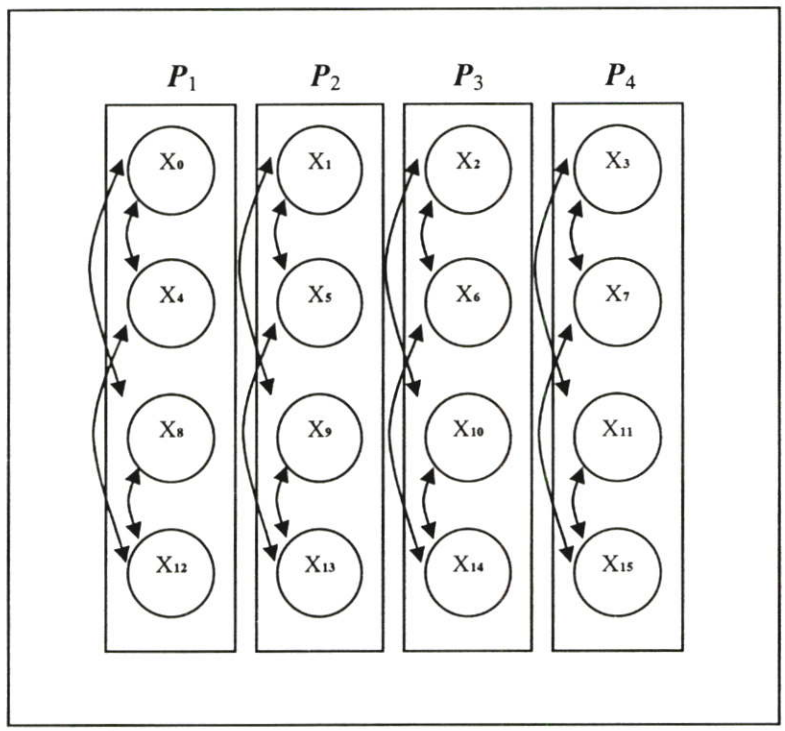
ในงานวิจัยที่เกี่ยวข้องกับการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ ส่วนใหญ่จะใช้ขั้นตอนวิธีแถวและหลักในการคำนวณการแปลงข้อมูลเป็นหลัก หรือที่เรียกว่า การแปลงฟูเรียร์แบบเร็วแบบขนานด้วยขั้นตอนวิธีแถวและหลัก (Row – Column Parallel Fast Fourier Transform : RC PFFT) เนื่องจากเป็นวิธีที่ง่ายต่อการพัฒนา และชุดข้อมูล (Library) ที่ใช้ช่วยในการคำนวณ เช่น FFTW หรือ FFTPACK เป็นต้น [4] [5] [6] ก็จะใช้ขั้นตอนวิธีนี้ทั้งสิ้น

รูปแบบการคำนวณด้วยขั้นตอนวิธีแถวและหลักแบบขนานจะเหมือนกับแบบอนุกรม แต่จะต่างกันที่การแบ่งงานให้แต่ละหน่วยประมวลผลทำงาน ซึ่งในแบบอนุกรมนั้นจะมีเพียงหนึ่งหน่วยประมวลผลทำให้ขั้นตอนในการคำนวณต้องรอข้อมูลในแต่ละส่วนให้คำนวณเสร็จก่อนจึงจะเริ่มทำการคำนวณขั้นต่อไปได้ แต่ในแบบขนานนั้นจะแบ่งงานให้แต่ละหน่วยประมวลผลช่วยกันคำนวณทำให้ลดปัญหาเรื่องเวลาในการรอข้อมูลแต่ละส่วน จึงมีผลให้เวลาในการคำนวณลดลง ซึ่งรูปแบบการแบ่งงานสามารถอธิบายได้ดังนี้

ถ้าให้ N อยู่ในรูปกำลัง 2 และลำดับมีขนาด N จะได้กลุ่มข้อมูลอยู่ในรูปอาร์เรย์ 2 มิติที่มีขนาดเป็น $N \times N$ เราจะสามารถทำการคำนวณการแปลงฟูเรียร์แบบเร็วด้วยขั้นตอนวิธีแถวและหลัก 3 ขั้นตอนดังนี้ [17] [18] [20]

ขั้นตอนที่ 1

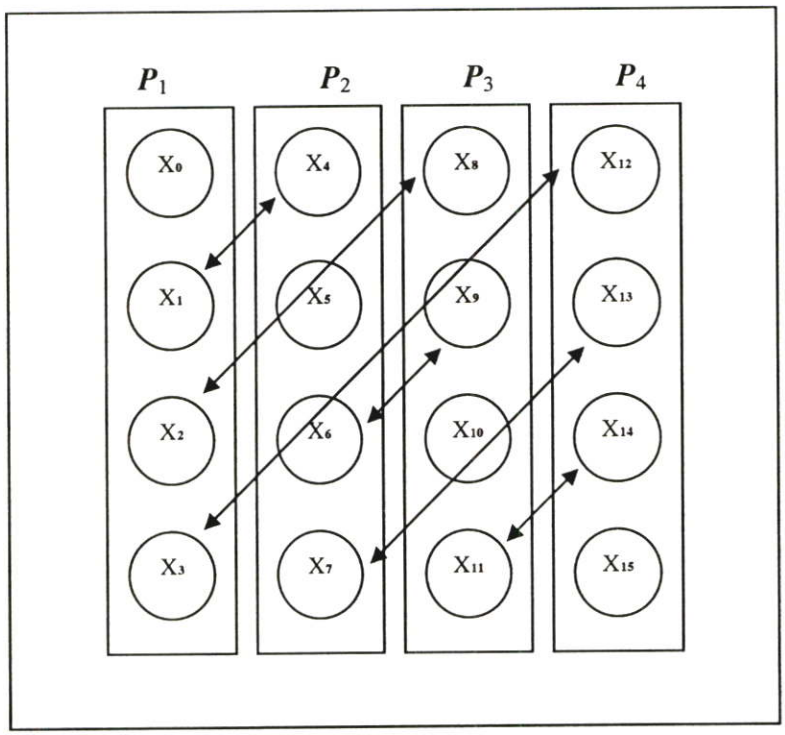
แต่ละหน่วยประมวลผลทำการคำนวณการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติของข้อมูลภายในหน่วยประมวลผลของตัวเองเท่านั้น อย่างเช่น ข้อมูลมีขนาด 4 ×4 แล้วมีหน่วยประมวลผลทั้งหมด 4 หน่วยประมวลผลก็จะทำการแบ่งให้หน่วยประมวลผลละหนึ่งหลัก ดังรูป 2.18 ซึ่งจะทำการคำนวณเฉพาะภายในหน่วยประมวลผลของตัวเองเท่านั้น ไม่มีการติดต่อสื่อสารกับหน่วยประมวลผลอื่น



รูปที่ 2.18 แสดงขั้นตอนการคำนวณการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติของข้อมูลภายในหน่วยประมวลผลของตัวเอง

ขั้นตอนที่ 2

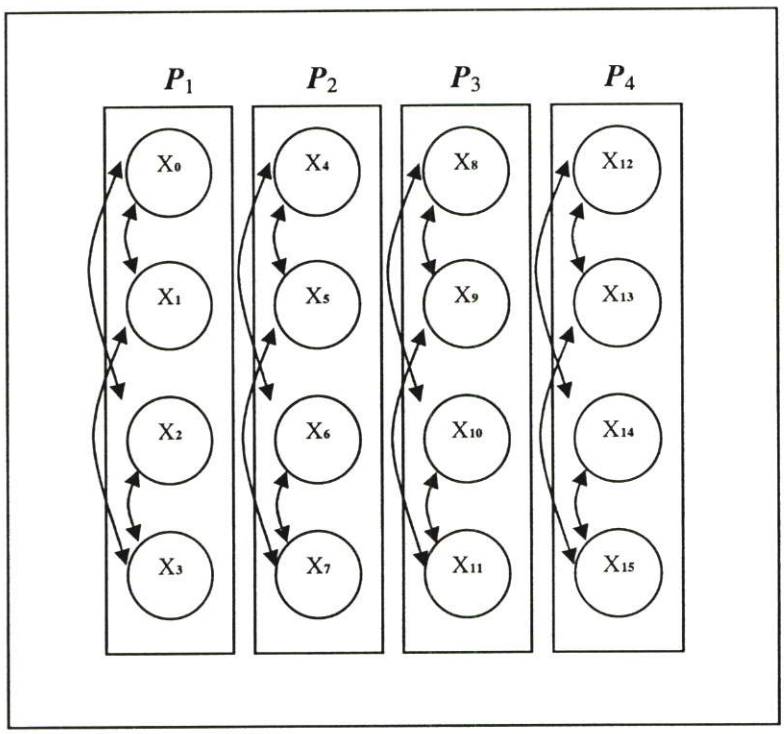
ทำการทรานสโพสข้อมูล โดยจะต้องทำการทรานสโพสทั้งภายในหน่วยประมวลผล และภายนอกหน่วยประมวลผล ซึ่งการทำงานในส่วนนี้จะเป็นส่วนที่มีการติดต่อสื่อสารกับหน่วยประมวลผลอื่น ทำให้เป็นส่วนที่ใช้เวลามากที่สุด โดยเฉลี่ยคิดเป็น 30 – 40% ของเวลาที่ใช้ในการประมวลผลทั้งหมด



รูปที่ 2.19 แสดงขั้นตอนการทรานสโพสของข้อมูลระหว่างหน่วยประมวลผล

ขั้นตอนที่ 3

แต่ละหน่วยประมวลผลทำการคำนวณการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติของข้อมูลที่ทำการทรานสโพสแล้ว ซึ่งในขั้นตอนการคำนวณจะคำนวณเฉพาะข้อมูลภายในหน่วยประมวลผลของตัวเองเท่านั้น ไม่มีการติดต่อสื่อสารกับหน่วยประมวลผลอื่นเหมือนอย่างในขั้นตอนที่ 1



รูปที่ 2.20 แสดงขั้นตอนการคำนวณการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติของข้อมูลภายในหน่วยประมวลผล

2.7 การวัดสมรรถนะของการประมวลผลแบบขนาน (Performance Measurement)

วิธีการวัดสมรรถนะ (Performance Measurement) ของขั้นตอนวิธีการทางคอมพิวเตอร์แบบขนาน (Parallel algorithm) จะมีความซับซ้อนกว่าขั้นตอนวิธีการทางคอมพิวเตอร์แบบอนุกรม (Sequential algorithm) โดยสามารถประเมินผลจากเวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time) อัตราการเพิ่มขึ้นของความเร็ว (Speedup) และประสิทธิภาพ (Efficiency) เมื่อใช้หน่วยประมวลผลเพิ่มขึ้น ในการประมวลผลจริงบนระบบคลัสเตอร์สามารถวัดเวลาที่ใช้ในการประมวลผลงานหนึ่งได้โดยจับเวลาที่ใช้ในการประมวลผลตั้งแต่เริ่มต้นจนสิ้นสุดการประมวลผล

ซึ่งวัดได้ทั้งแบบอนุกรม (Sequential Programming) และแบบขนาน (Parallel Programming) โดยจะทำการวัดเวลาที่ใช้ในการประมวลผล ซึ่งไม่รวมเวลาที่ใช้ในการสร้างข้อมูลมาเก็บไว้ในหน่วยความจำสำรอง (Hard disk) และการตั้งค่าระบบ (Initialization) ของ MPI ซึ่งกระบวนการทั้งสองจะถูกทำครั้งเดียว

2.7.1 เวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time)

การวัดเวลาที่ใช้ในการประมวลผลในทางทฤษฎี (Theoretical Approach) เวลาที่ใช้ในการประมวลผลแบบขนาน (Parallel Computation) จะสามารถคำนวณได้ดังสมการที่ 2.10 [17] [18]

$$T_p = \frac{T_s}{P} \tag{2.10}$$

เมื่อ T_p คือ เวลาที่ใช้ในการประมวลผลด้วย P หน่วยประมวลผล

T_s คือ เวลาที่ใช้ในการประมวลผลด้วยหนึ่งหน่วยประมวลผล

P คือ จำนวนหน่วยประมวลผล (Processor) ที่ใช้ในระบบ

ซึ่งกรณีนี้จะเรียกว่าเป็นกรณีอุดมคติ (Ideal Case) เพราะว่ามีการสมมติให้เวลาที่ใช้ในการติดต่อสื่อสาร (Communication Time) ระหว่างหน่วยประมวลผลมีค่าเป็นศูนย์ แต่ในทางปฏิบัติ (Practical Approach) บนระบบคลัสเตอร์ การวัดเวลาที่ใช้ในการประมวลผลแบบขนานจะวัดโดยการจับเวลา ตั้งแต่เริ่มประมวลผล จนกระทั่งสิ้นสุดการประมวลผล หรือได้คำตอบที่ต้องการ ซึ่งเวลาที่วัดได้ดังกล่าวจะรวมเวลาที่ใช้ในการติดต่อสื่อสารด้วย ดังนั้นถึงแม้ว่าจำนวนของหน่วยประมวลผล (P) ในระบบจะมีเพิ่มขึ้นมากๆ แต่เวลาที่ได้ของระบบจะไม่ลดลงจนเข้าใกล้ศูนย์ เพราะสาเหตุมาจากเวลาที่ใช้ในการประมวลผลทั้งหมดประกอบด้วยเวลาที่ใช้ในการประมวลผล (Computation Time) และเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล (Communication Time) ซึ่งปกติจะไม่เป็นศูนย์ ดังกรณีอุดมคติ โดยจะมีลักษณะตามสมการที่ 2.11

$$T_p = t_{computation} + t_{communication} \tag{2.11}$$

เมื่อ T_p คือ เวลาที่ใช้ในการประมวลผลแบบขนานด้วย P หน่วยประมวลผล

$t_{computation}$ คือ เวลาที่ใช้ในการประมวลผลซึ่งไม่รวม เวลาที่ใช้ในการติดต่อสื่อสาร

ระหว่างหน่วยประมวลผล

$t_{communication}$ คือ เวลาที่ใช้ในการสื่อสารระหว่างหน่วยประมวลผล

ดังนั้นในทางปฏิบัติเราสามารถเพิ่มหน่วยประมวลผลเข้าไปในระบบมากขึ้น และเวลาจะลดลงช่วงหนึ่งเท่านั้น เพราะเวลาที่ใช้ในการประมวลผล (Computation Time) มากกว่าเวลาที่ใช้ในการติดต่อสื่อสาร (Communication Time) ระหว่างหน่วยประมวลผล แต่ ณ จุดหนึ่งเมื่อเพิ่มหน่วยประมวลผลเข้าไปอีก เวลาจะไม่ลดลงแต่อาจจะเพิ่มขึ้นเนื่องจากเวลาที่ใช้ในการสื่อสารระหว่างหน่วยประมวลผล $t_{communication}$ มากขึ้นและมีค่ามากกว่า เวลาที่ใช้ในการประมวลผล

2.7.2 อัตราการเพิ่มของความเร็ว (Speedup)

การวัดอัตราการเพิ่มของความเร็ว (Speedup) แสดงได้ดังสมการที่ 2.12 [17] [18]

$$S_p = \frac{T_s}{T_p} \leq p \tag{2.12}$$

เมื่อ S_p คือ อัตราการเพิ่มของความเร็ว โดยใช้ P หน่วยประมวลผล ซึ่งมีค่าสูงสุดคือ P

T_s คือ เวลาที่ใช้ในการประมวลผลโดยใช้หนึ่งหน่วยประมวลผล

T_p คือ เวลาที่ใช้ในการประมวลผลแบบขนานด้วย P หน่วยประมวลผล

2.7.3 ประสิทธิภาพ (Efficiency)

การวัดประสิทธิภาพ แสดงดังสมการที่ 2.13 [17] [18]

$$E_p = \frac{S_p}{P} \leq 1 \tag{2.13}$$

เมื่อ E_p คือ ประสิทธิภาพของระบบมีค่าสูงสุดคือ 1

S_p คือ อัตราการเพิ่มของความเร็วที่คำนวณได้จากสมการที่ 2.3

P คือ จำนวนหน่วยประมวลผลที่ใช้บนระบบ

2.8 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการแปลงฟูเรียร์นั้น สามารถแบ่งออกเป็น 2 กลุ่มใหญ่ๆดังนี้ คือ

2.8.1 กลุ่มที่เกี่ยวกับการพัฒนาในด้านขั้นตอนวิธี (Algorithm)

โดยงานวิจัยในกลุ่มนี้จะมุ่งเน้นที่การพัฒนาและคิดค้นขั้นตอนวิธีการแปลงฟูเรียร์แบบต่างๆให้มีประสิทธิภาพเพิ่มขึ้นจากขั้นตอนวิธีเดิม ซึ่งมีการพัฒนาทั้งบนระบบที่มีหน่วยประมวลผลเดียว และบนระบบแบบขนานที่มีลายหน่วยประมวลผล ตัวอย่างงานวิจัยในกลุ่มนี้ทั้งบนระบบหน่วยประมวลผลเดียว และบนระบบแบบขนาน [2] [7] [8] [11] [14] [18] [19] ตัวอย่างเช่น

ในปี 1977 David B. Harris, James H. McClellan, David S. K. Chan และ Han W. Schuessler [7] ทำการนำเสนอวิธีการแปลงฟูเรียร์แบบเร็วแบบใหม่ที่ชื่อว่า Vector Radix FFT ที่สามารถคำนวณการแปลงฟูเรียร์ได้ โดยทำการแบ่งข้อมูลออกเป็นชุดข้อมูลย่อยจำนวน 4 ชุดไปเรื่อยๆ แล้วทำการคำนวณการแปลงฟูเรียร์ ซึ่งสามารถลดปริมาณการคูณได้ 25%

ในปี 1988 Clare Y. Chu [2] ทำการทดลองวิธีการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติบนเครื่องคอมพิวเตอร์แบบขนานที่มีโครงสร้างของหน่อยประมวลผลติดต่อกันแบบไฮเปอร์คิว และทำการเปรียบเทียบประสิทธิภาพระหว่างวิธีการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติ 4 วิธี คือ

1. Transpose-Spilt (TS) 2D-FFT
2. Block(B) 2D-FFT
3. Local-Distribute(LD) 2D-FFT
4. Partial Vector Radix(PVR) 2D-FFT

ซึ่งทั้ง 4 วิธีจะพัฒนามาจากขั้นตอนวิธีพื้นฐานทั้ง 2 ขั้นตอนวิธีคือ วิธีที่ 1-3 จะพัฒนามาจากขั้นตอนวิธีRow-Column ส่วนวิธีที่ 4 พัฒนามาจากขั้นตอนวิธีVector-Radixโดยจากการทดลองสามารถสรุปได้ดังนี้

1. วิธี Transpose-Split จะมีประสิทธิภาพมากที่สุดเมื่อใช้กับหน่วยประมวลผลที่มีจำนวนน้อย เนื่องจากจะใช้เวลาในการติดต่อระหว่างหน่วยประมวลมาก
2. วิธี Vector Radix จะใช้ปริมาณการคูณจำนวนเชิงซ้อนน้อยกว่าวิธี Row-Column อยู่ 25% และเป็นวิธีที่เหมาะกับการติดต่อสื่อสารระหว่างหน่วยประมวลผลมากที่สุด

2.8.2 กลุ่มที่เกี่ยวกับการนำไปใช้พัฒนาในงานด้านการประยุกต์ (Application)

โดยงานวิจัยในกลุ่มนี้จะมุ่งเน้นที่การนำเอาวิธีการแปลงฟูเรียร์ไปใช้พัฒนาในงานด้านต่างๆที่เกี่ยวข้อง เช่น งานในด้านการประมวลผลสัญญาณดิจิทัล (Digital Signal Processing) หรือในด้านการประมวลผลรูปภาพดิจิทัล (Digital Image Processing) [1] [12]

ซึ่งงานวิจัยที่นำเสนอในวิทยานิพนธ์นี้เป็นงานวิจัยในด้านการพัฒนาขั้นตอนวิธี โดยมุ่งเน้นไปที่การพัฒนาเพื่อเพิ่มสมรรถนะของวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์ ด้วยการลดเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล ซึ่งพัฒนาบนระบบพีซีคลัสเตอร์ซึ่งมีการประมวลผลแบบขนาน

บทที่ 3
การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน บนระบบพีซีคลัสเตอร์

วิทยานิพนธ์นี้เป็นการศึกษาวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์ โดยผู้วิจัยจะศึกษาขั้นตอนวิธีการการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน โดยงานวิจัยนี้จะมุ่งเน้นที่การเพิ่มประสิทธิภาพของวิธีการติดต่อสื่อสารระหว่างหน่วยประมวลผลเพื่อให้สามารถนำไปประยุกต์ใช้จริงบนระบบคอมพิวเตอร์แบบขนานที่มีอยู่ได้อย่างมีประสิทธิภาพ

วิทยานิพนธ์นี้มีขั้นตอนการศึกษาและการดำเนินงานวิจัย ดังนี้

1) ศึกษาขั้นตอนวิธีการทางคอมพิวเตอร์ (Computer Algorithm) ของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบอนุกรมและแบบขนาน

2) ศึกษางานวิจัยที่เกี่ยวข้องกับการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน

3) ทำการตั้งสมมติฐานโดยคาดว่าวิธีการหาการแปลงฟูเรียร์แบบเร็ว 2 มิติที่ได้นำเสนอเมื่อใช้จำนวนหน่วยประมวลผลเพิ่มขึ้น จะเป็นการช่วยให้เวลาในการประมวลผลในการหาการแปลงฟูเรียร์แบบเร็ว 2 มิติลดลงโดยเฉพาะกรณีที่ข้อมูลมีขนาดใหญ่ๆ ซึ่งเป็นการทำให้การแปลงฟูเรียร์แบบเร็ว 2 มิติมีสมรรถนะ (Performance) มากขึ้น และสรุปได้ว่าการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานมีสมรรถนะมากกว่าการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบอนุกรม ในกรณีที่ข้อมูลมีขนาดใหญ่ ๆ

4) นำเสนอวิธีการการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่มีสมรรถนะดังที่ได้ตั้งสมมติฐานไว้ในข้อ (3)

5) พัฒนาโปรแกรมการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่นำเสนอโดยใช้โปรแกรมภาษาซี (C Language) บนระบบพีซีคลัสเตอร์ที่ใช้ระบบปฏิบัติการลีนุกต์ (Linux) และตรงตามมาตรฐานภาษาเอ็มพีไอ (MPI Standard) ที่สนับสนุนการโปรแกรมแบบขนาน

6) วิเคราะห์ผลการทดลองโดยการเปรียบเทียบสมรรถนะ (Performance) ของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยประเมินผลจากเวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time) อัตราการเพิ่มขึ้นของความเร็ว (Speedup) และประสิทธิภาพ (Efficiency)

7) สรุปผลการทดลอง พร้อมเสนอแนวทางการพัฒนางานวิจัย

8) เขียนวิทยานิพนธ์

ในบทนี้ผู้วิจัยเสนอวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน 3 วิธี ซึ่งเป็นวิธีที่เหมาะสมกับระบบคลัสเตอร์ คือ

1) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยวิธี RC 2-D PFFT (Row-Column Two-Dimensional Parallel Fast Fourier Transform : RC 2-D PFFT)
2) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยวิธี VR 2-D PFFT (Vector Radix Two-Dimensional Parallel Fast Fourier Transform : VR 2-D PFFT)
3) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยวิธี PBRC 2-D PFFT (Partition Block Row-Column Two-Dimensional Parallel Fast Fourier Transform : PBRC 2-D PFFT)

โดยวิธีแรก (วิธี RC 2-D PFFT) เป็นวิธีที่ส่วนใหญ่ใช้ในงานวิจัยที่เกี่ยวข้องกับการแปลงฟูเรียร์เนื่องจากเป็นวิธีที่มีประสิทธิภาพในการคำนวณมากที่สุดและเป็นวิธีที่ง่ายต่อการพัฒนา แต่เมื่อนำมาใช้กับการคำนวณแบบขนาน จะพบว่าประสิทธิภาพจะลดลงเมื่อเพิ่มจำนวนหน่วยประมวลผล เนื่องจากมีปัญหาในด้านการติดต่อสื่อสารระหว่างหน่วยประมวลผลที่มีความซับซ้อนมากขึ้น ส่วนวิธีที่สอง (วิธี VR 2-D PFFT) เป็นวิธีที่เดิมมีผู้พัฒนาขึ้นมาเพื่อใช้ในการคำนวณการแปลงฟูเรียร์แบบเร็ว 2 มิติอีกวิธีหนึ่งนอกเหนือจากวิธีแถวและหลัก (Row – Column Algorithm) [3] โดยในวิทยานิพนธ์นี้ได้นำเอาขั้นตอนวิธีดังกล่าวมาพัฒนาเพื่อใช้บนระบบที่มีการประมวลผลแบบขนาน โดยใช้ระบบพีซีคลัสเตอร์ในการพัฒนา ซึ่งพบว่าวิธีนี้มีความเหมาะสมกับรูปแบบของการประมวลผลแบบขนานในด้านการติดต่อสื่อสารระหว่างหน่วยประมวลผล เนื่องจากมีรูปแบบการแบ่งและรวมข้อมูลของแต่ละหน่วยประมวลผลที่ไม่ซับซ้อนและง่ายต่อการพัฒนา แต่วิธีนี้มีปัญหาในด้านขั้นตอนการคำนวณที่มีความซับซ้อนกว่าวิธีแรก ทำให้ใช้เวลาในการคำนวณที่มากกว่า และมีข้อจำกัดเกี่ยวกับหน่วยความจำที่ใช้และจำนวนหน่วยประมวลผลที่นำมาใช้ในการคำนวณที่จะมีจำนวนไม่เกิน 4 หน่วยประมวลผลจึงจะมีประสิทธิภาพมากที่สุด ส่วนวิธีสุดท้าย (วิธี PBRC 2-D PFFT) เป็นวิธีใหม่ที่เสนอขึ้นในวิทยานิพนธ์นี้เพื่อแก้ปัญหาของทั้ง 2 วิธีข้างต้น โดยจะเป็นการเพิ่มประสิทธิภาพในด้านการคำนวณและการติดต่อสื่อสารระหว่างหน่วยประมวลผล ซึ่งเป็นวิธีที่นำเอาข้อดีของขั้นตอนของทั้ง 2 วิธีแรกมาประยุกต์ใช้ร่วมกันในการพัฒนาเพื่อใช้กับการประมวลผลแบบขนานบนระบบพีซีคลัสเตอร์ โดยแบ่งขั้นตอนเป็น 2 ส่วนใหญ่ๆ คือ ส่วนแรกในขั้นตอนของการแบ่งและรวมงานที่มีการติดต่อสื่อสารระหว่างหน่วยประมวลผล จะใช้ขั้นตอนของวิธี VR 2-D PFFT ซึ่งเป็นวิธีที่มีประสิทธิภาพในด้านการติดต่อสื่อสารระหว่างหน่วยประมวลผล และส่วนที่สองในขั้นตอนการคำนวณจะใช้ขั้นตอนของวิธี RC 2-D PFFT ซึ่งเป็นวิธีที่มีประสิทธิภาพในด้านการคำนวณ ทำให้ได้เป็นวิธีที่ประสิทธิภาพในด้านการคำนวณและการติดต่อสื่อสารระหว่างหน่วยประมวลผลมากกว่า 2 วิธีแรก

3.1 การแปลงฟูเรียร์แบบเร็วแบบ 2 มิติ ด้วยขั้นตอนวิธี RC 2-D PFFT

วิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยขั้นตอนวิธี RC 2-D PFFT ได้พัฒนาขึ้นจากวิธีการพื้นฐานของการแปลงฟูเรียร์แบบเร็ว โดยใช้วิธีการแปลงฟูเรียร์แบบเร็วแบบ 1 มิติมาทำการคำนวณทั้งทางแถวและหลักของข้อมูล ซึ่งในการประมวลผลจะประกอบด้วยหน่วยประมวลผลซึ่งแบ่งเป็น 2 ชนิด คือ หน่วยประมวลผลหลัก (Master Computer) จำนวน 1 หน่วย และหน่วยประมวลผลทำงาน (Worker Computer) จำนวน P หน่วย มีขั้นตอนในการทำงานดังต่อไปนี้

ในส่วนของหน่วยประมวลผลหลัก (Master Computer)

ขั้นตอนที่ 1 สร้างข้อมูล คือ เมตริกซ์ A ขนาด N x N และ ค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (Complex n^{th} of root of unity : ω_N) เก็บลงในหน่วยความจำ

ขั้นตอนที่ *2* ทำการแบ่งข้อมูลตามแถวของเมตริกซ์ A ขนาด k = $\lceil N/p \rceil$ เพื่อให้ภาระงาน (Work load) มีขนาดเท่ากัน สำหรับหน่วยประมวลผลทำงาน P หน่วย เพื่อคำนวณหาผลคูณเมตริกซ์ C จำนวน $\lceil N/p \rceil$ แถวแบบขนาน

ขั้นตอนที่ 3 ส่งขนาดของช่วงและขนาดของเมตริกซ์ A ที่แต่ละหน่วยประมวลผลทำงานต้องทำการอ่านจากหน่วยความจำของหน่วยประมวลผลหลัก และส่งค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (Complex n^{th} of root of unity : ω_N)ให้ทุกหน่วยประมวลผลทำงาน

ขั้นตอนที่ 4 รอรับผลการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติ จากหน่วยประมวลผลทำงาน

ในส่วนหน่วยประมวลผลทำงาน จำนวน P หน่วย

ขั้นตอนที่ 1 รอรับการติดต่อจากหน่วยประมวลผลหลักและหลังจากนั้น อ่านข้อมูลบางส่วนของเมตริกซ์ A ขนาด k = $\lceil N/p \rceil$แถว และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (Complex n^{th} of root of unity : ω_N) ทั้งหมดจากหน่วยความจำของหน่วยประมวลผลหลัก

ขั้นตอนที่ 2 ทำการคำนวณแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ ตามแถวของข้อมูล

ขั้นตอนที่ 3 ทำการทรานสโพสข้อมูล โดยส่วนนี้จะมีการติดต่อกับหน่วยประมวลผลอื่น

ขั้นตอนที่ 4 ทำการคำนวณแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ ตามแถวของข้อมูลที่ทำการทรานสโพสแล้ว

ขั้นตอนที่ 5 ส่งผลแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ กลับไปยังหน่วยประมวลผลหลัก

ตัวอย่างที่3.1 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยขั้นตอนวิธี RC โดยใช้จำนวนหน่วยประมวลผล 4 หน่วยประมวลผล เมื่อข้อมูลเป็นเมตริกซ์ขนาด 8 × 8 ดังรูป 3.2

เมตริกซ์ A

a0	a1	a2	a3	a4	a5	a6	a7
a8	a9	a10	a11	a12	a13	a14	a15
a16	a17	a18	a19	a20	a21	a22	a23
a24	a25	a26	a27	a28	a29	a30	a31
a32	a33	a34	a35	a36	a37	a38	a39
a40	a41	a42	a43	a44	a45	a46	a47
a48	a49	a50	a51	a52	a53	a54	a55
a56	a57	a58	a59	a60	a61	a62	a63

Complex n^{th} of root of unity (ω_N)

ω_N^0
ω_N^1
ω_N^2
ω_N^3
ω_N^4
ω_N^5
ω_N^6
ω_N^7

รูปที่ 3.1 แสดงเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N)

วิธีทำ

ขั้นตอนที่ 1 หน่วยประมวลผลหลักแบ่งข้อมูลบางส่วนตามแถวของเมตริกซ์ A และข้อมูลทั้งหมดของค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) ให้แต่ละหน่วยประมวลผล(P_i) เมื่อ i= 1,2,...,p และ k = $\lceil N/p \rceil$ โดยกรณีนี้ p=4 N=8 และ k=2 ดังรูปที่ 3.3

เมตริกซ์ A

a0	a1	a2	a3	a4	a5	a6	a7	} หน่วยประมวลผลที่ 1 จำนวน 2 แถว (แถวที่ 1-2)
a8	a9	a10	a11	a12	a13	a14	a15	
a16	a17	a18	a19	a20	a21	a22	a23	} หน่วยประมวลผลที่ 2 จำนวน 2 แถว (แถวที่ 3-4)
a24	a25	a26	a27	a28	a29	a30	a31	
a32	a33	a34	a35	a36	a37	a38	a39	} หน่วยประมวลผลที่ 3 จำนวน 2 แถว (แถวที่ 5-6)
a40	a41	a42	a43	a44	a45	a46	a47	
a48	a49	a50	a51	a52	a53	a54	a55	} หน่วยประมวลผลที่ 4 จำนวน 2 แถว (แถวที่ 7-8)
a56	a57	a58	a59	a60	a61	a62	a63	

รูปที่ 3.2 แสดงการแบ่งข้อมูลเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน

Complex n[th] of root of unity : ω_N

ω_N^0
ω_N^1
ω_N^2
ω_N^3
ω_N^4
ω_N^5
ω_N^6
ω_N^7

ทุกหน่วยประมวลผล

รูปที่ 3.2 (ต่อ)

ขั้นตอนที่ 2 หน่วยประมวลผลแต่ละหน่วยทำการอ่านข้อมูลบางส่วนตามแถวของเมตริกซ์ A และข้อมูลทั้งหมดของค่า Complex n[th] of root of unity (ω_N) จากนั้นคำนวณการแปลงฟูเรียร์ไปพร้อมๆ กันทุกหน่วยประมวลผล

หน่วยประมวลผลที่ 1 (P_1) ทำการแปลงฟูเรียร์ของเมตริกซ์จำนวน 2 แถว คือ แถวที่ 1 และแถวที่ 2

a0	a1	a2	a3	a4	a5	a6	a7
a8	a9	a10	a11	a12	a13	a14	a15

FFT

A0	A1	A2	A3	A4	A5	A6	A7
A8	A9	A10	A11	A12	A13	A14	A15

หน่วยประมวลผลที่ 2 (P_2) ทำการแปลงฟูเรียร์ของเมตริกซ์จำนวน 2 แถว คือ แถวที่ 3 และแถวที่ 4

a16	a17	a18	a19	a20	a21	a22	a23
a24	a25	a26	a27	a28	a29	a30	a31

FFT

A16	A17	A18	A19	A20	A21	A22	A23
A24	A25	A26	A27	A28	A29	A30	A31

หน่วยประมวลผลที่ 3 (P_3) ทำการแปลงฟูเรียร์ของเมตริกซ์จำนวน 2 แถว คือ แถวที่ 5 และแถวที่ 6

a32	a33	a34	a35	a36	a37	a38	a39
a40	a41	a42	a43	a44	a45	a46	a47

FFT

A32	A33	A34	A35	A36	A37	A38	A39
A40	A41	A42	A43	A44	A45	A46	A47

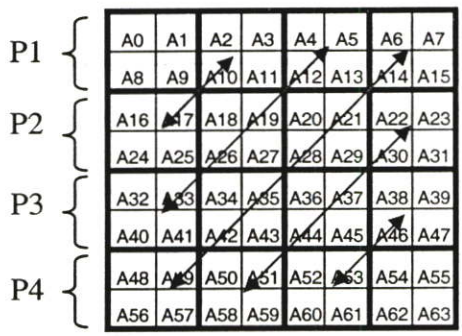
หน่วยประมวลผลที่ 4 (P_4) ทำการแปลงฟูเรียร์ของเมตริกซ์จำนวน 2 แถว คือ แถวที่ 7 และแถวที่ 8

a48	a49	a50	a51	a52	a53	a54	a55
a56	a57	a58	a59	a60	a61	a62	a63

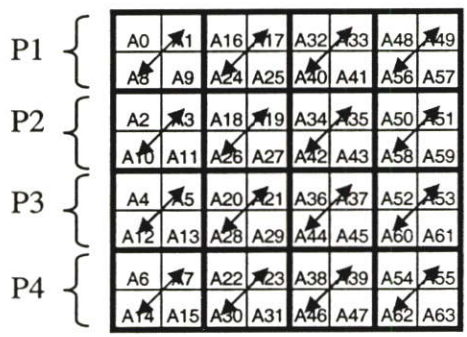
FFT

A48	A49	A50	A51	A52	A53	A54	A55
A56	A57	A58	A59	A60	A61	A62	A63

ขั้นตอนที่ 3 หน่วยประมวลผลแต่ละหน่วยทำการทรานสโพสข้อมูล (Transpose Data) ระหว่างหน่วยประมวลผล และภายในหน่วยประมวลผล ดังรูปที่ 3.3 และ 3.4 ตามลำดับ



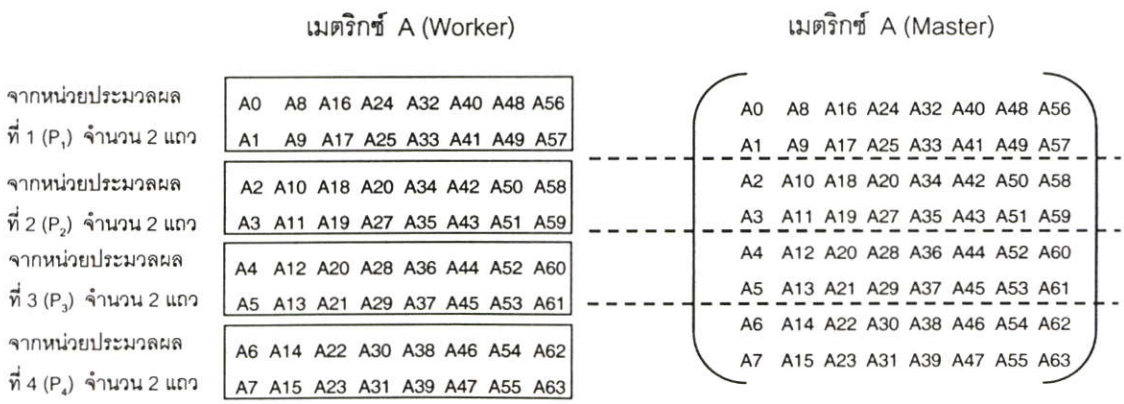
รูปที่ 3. 3 แสดงขั้นตอนการทรานสโพสระหว่างหน่วยประมวลผล



รูปที่ 3.4 แสดงขั้นตอนการทรานสโพสภายในหน่วยประมวลผล

ขั้นตอนที่ 3 หน่วยประมวลผลแต่ละหน่วยทำการแปลงฟูเรียร์ของข้อมูลที่ได้ทำการทรานสโพสแล้วเหมือนขั้นตอนที่ 2

ขั้นตอนที่ 4 แต่ละหน่วยประมวลผลทำการส่งผลการแปลงฟูเรียร์คืนให้หน่วยประมวลผลหลัก



```
Start MPI

MPI_Init(&argc, &argv);
MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
MPI_Comm_size( MPI_COMM_WORLD, &numprocs );

Create Data

Send Data For Each Node

MPI_Scatter( &Input, IMAGE_SLICE * IMAGE_SIZE , MPI_FLOAT,
             &Input_slice, IMAGE_SLICE * IMAGE_SIZE , MPI_FLOAT,
             SOURCE_PROCESSOR, MPI_COMM_WORLD);

Perform on Local Node

Perform FFT Computation on Colomn of Data

  for (i=0;i<IMAGE_SLICE;i++)
      fft(&Input_slice[i][0], w_common, IMAGE_SIZE, logn);

Communication between all pairs of nodes

Transpose Data
   MPI_Alltoall(&sendbuf,sendcount,sendtype,&recvbuf,recvcnt,recvtype,comm)

Perform on Local Node

Perform FFT Computation on Row of Data

   for (i=0;i<IMAGE_SLICE;i++)
       fft(&Input _slice[i][0], w_common, IMAGE_SIZE, logn);

Gather Data For Each Node

MPI_Gather( Input_slice, IMAGE_SLICE * IMAGE_SIZE, MPI_FLOAT,Input, IMAGE_SLICE *
            IMAGE_SIZE, MPI_FLOAT,DEST_PROCESSOR, MPI_COMM_WORLD);
```

รูปที่ 3.5 แสดงตัวอย่างรหัสเหมือนเอ็มพีไอ-ซี (MPI-C like) ขั้นตอนวิธี RC 2-D PFFT

3.2 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยขั้นตอนวิธี VR 2-D PFFT

วิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยขั้นตอนวิธี VR นั้น จะเป็นการแบ่งข้อมูลในรูปเมตริกซ์ขนาด N ×N ออกเป็นข้อมูลขนาด N/2 ×N/2 จำนวน 4 ส่วน โดยข้อมูลขนาด N/2 ×N/2 แต่ละส่วนก็จะทำการแบ่งออกได้เป็นข้อมูลขนาด N/4 ×N/4 อีกจำนวน 4 ส่วนเช่นเดียวกัน หลังจากนั้นจะทำการแบ่งแบบเดิมไปเรื่อยๆ จนกระทั่งข้อมูลมีขนาดเป็น 2 × 2 ก็จะทำการคำนวณ ซึ่งวิธี VR นั้นจะใช้เวลาในขั้นตอนการแบ่งข้อมูลมาก แต่ในส่วนของขั้นตอนการคำนวณจะทำการคำนวณเฉพาะภายในหน่วยประมวลผลของตนเองเท่านั้น ดังนั้นจะมีการติดต่อระหว่างหน่วยประมวลผลเพียงแค่ขั้นตอนที่หน่วยประมวลผลหลักส่งข้อมูลให้แต่ละหน่วยประมวลผลคำนวณ และตอนที่แต่ละหน่วยประมวลผลส่งข้อมูลที่คำนวณเสร็จแล้วให้กับหน่วยประมวลผลหลักเท่านั้น โดยในการคำนวณจะประกอบด้วยหน่วยประมวลผลซึ่งแบ่งเป็น 2 ชนิด คือ หน่วยประมวลผลหลัก (Master Computer) จำนวน 1 หน่วย และหน่วยประมวลผลทำงาน (Worker Computer) จำนวน P หน่วย (P = 2 หรือ 4)

ในส่วนของหน่วยประมวลผลหลัก (Master Computer)

ขั้นตอนที่ 1 สร้างข้อมูล คือ เมตริกซ์ A ขนาด N x N และ ค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) เก็บลงในหน่วยความจำ

ขั้นตอนที่ *2* ทำการแปลงข้อมูลให้อยู่ในรูปของอันดับบิตผกผัน (Bit Reversed Order) ทั้งทางแถวและทางหลักแล้วแบ่งข้อมูลของเมตริกซ์ A เป็น 4 ส่วนซึ่งแต่ละส่วนมีขนาด k = N/2 x N/2

ขั้นตอนที่ 3 ส่งขนาดของช่วงและขนาดของเมตริกซ์ A ที่แต่ละหน่วยประมวลผลทำงานต้องทำการอ่านจากหน่วยความจำของหน่วยประมวลผลหลัก และส่งค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) ให้ทุกหน่วยประมวลผลทำงาน

ขั้นตอนที่ 4 รอรับผลแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ จากหน่วยประมวลผลทำงาน P หน่วย

ในส่วนหน่วยประมวลผลทำงาน (Worker Computer) จำนวน P หน่วย (P = 2 หรือ 4)

ขั้นตอนที่ 1 รอรับการติดต่อจากหน่วยประมวลผลหลักและหลังจากนั้น อ่านข้อมูลบางส่วนของเมตริกซ์ A ขนาด k = N/2 x N/2 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) ทั้งหมดจากหน่วยความจำของหน่วยประมวลผลหลัก

ขั้นตอนที่ 2 ทำการคำนวณแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติด้วยวิธี VR

ขั้นตอนที่ 3 ส่งผลแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ กลับไปยังหน่วยประมวลผลหลัก

ตัวอย่างที่3.2 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยขั้นตอนวิธี VRโดยใช้จำนวนหน่วยประมวลผล 4 หน่วยประมวลผล เมื่อข้อมูลเป็นเมตริกซ์ขนาด 8 × 8 ดังรูปที่ 3.6

เมตริกซ์ A

a0	a1	a2	a3	a4	a5	a6	a7
a8	a9	a10	a11	a12	a13	a14	a15
a16	a17	a18	a19	a20	a21	a22	a23
a24	a25	a26	a27	a28	a29	a30	a31
a32	a33	a34	a35	a36	a37	a38	a39
a40	a41	a42	a43	a44	a45	a46	a47
a48	a49	a50	a51	a52	a53	a54	a55
a56	a57	a58	a59	a60	a61	a62	a63

Complex n^{th} of root of unity(ω_N)

ω_N^0
ω_N^1
ω_N^2
ω_N^3
ω_N^4
ω_N^5
ω_N^6
ω_N^7

รูปที่ 3.6 แสดงเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N)

วิธีทำ

ขั้นตอนที่ 1 หน่วยประมวลผลหลักทำการเตรียมข้อมูลของเมตริกซ์ A ให้อยู่ในรูปการสลับบิต (Bit Reverse Order) ทั้งแถวและหลักของข้อมูล ดังรูปที่ 3.7

เมตริกซ์ A

a0	a1	a2	a3	a4	a5	a6	a7
a8	a9	a10	a11	a12	a13	a14	a15
a16	a17	a18	a19	a20	a21	a22	a23
a24	a25	a26	a27	a28	a29	a30	a31
a32	a33	a34	a35	a36	a37	a38	a39
a40	a41	a42	a43	a44	a45	a46	a47
a48	a49	a50	a51	a52	a53	a54	a55
a56	a57	a58	a59	a60	a61	a62	a63

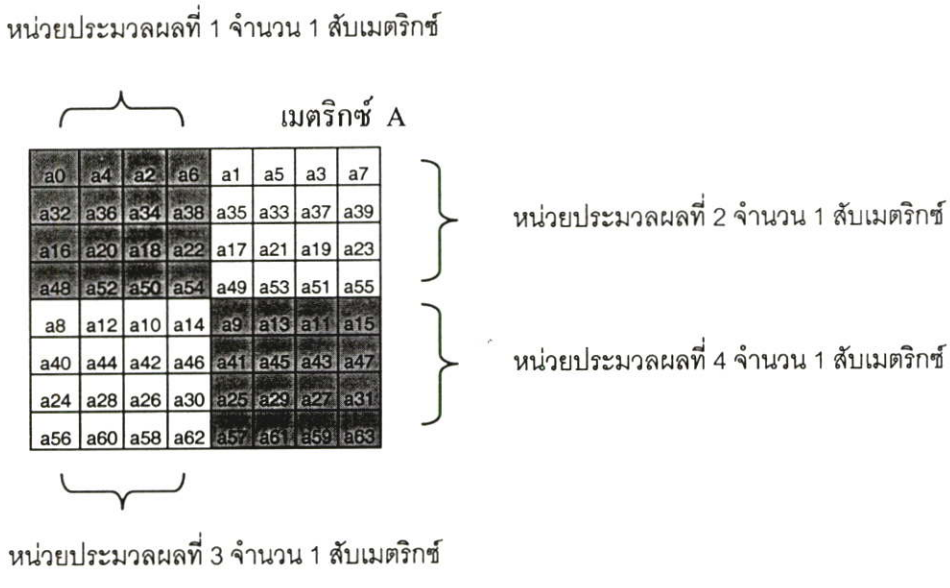
Bit Reverse

เมตริกซ์ A

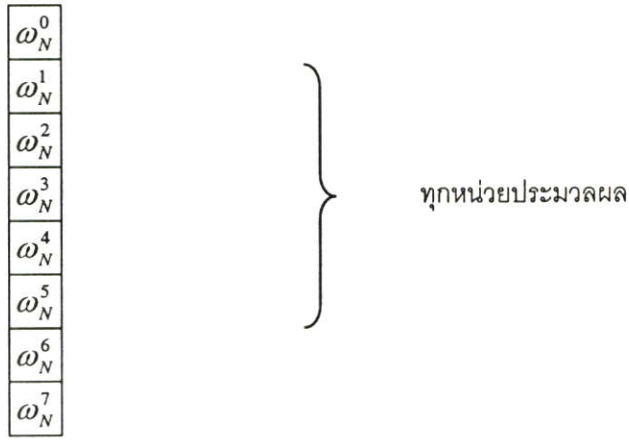
a0	a4	a2	a6	a1	a5	a3	a7
a32	a36	a34	a38	a35	a33	a37	a39
a16	a20	a18	a22	a17	a21	a19	a23
a48	a52	a50	a54	a49	a53	a51	a55
a8	a12	a10	a14	a9	a13	a11	a15
a40	a44	a42	a46	a41	a45	a43	a47
a24	a28	a26	a30	a25	a29	a27	a31
a56	a60	a58	a62	a57	a61	a59	a63

รูปที่ 3.7 แสดงการสลับบิต (Bit Reverse Order) เมตริกซ์ A

ขั้นตอนที่ 2 หน่วยประมวลผลหลักแบ่งข้อมูลของเมตริกซ์ A ออกเป็นกลุ่มย่อยจำนวน 4 กลุ่มเพื่อทำการส่งข้อมูลให้หน่วยประมวลผลทำงานหน่วยละ 1 กลุ่มและข้อมูลทั้งหมดของค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) ให้แต่ละหน่วยประมวลผล เมื่อ i= 1,2,...,p และ k = $\lceil n^2/p \rceil$ โดยกรณีนี้ p=4 n=8 และ k=16 ดังรูปที่ 3.8



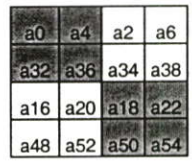
Complex n[th] of root of unity(ω_N)



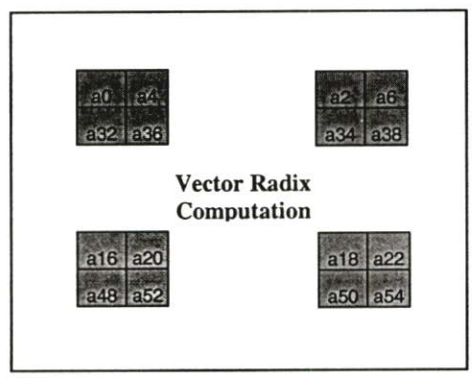
รูปที่ 3.8 แสดงการแบ่งข้อมูลเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน

ขั้นตอนที่ 3 หน่วยประมวลผลแต่ละหน่วยทำการอ่านข้อมูลสับเมตริกซ์ A และข้อค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) จากนั้นคำนวณหาผลการแปลงฟูเรียร์ด้วยขั้นตอนวิธี VR ของข้อมูลแต่ละส่วนไปพร้อมกัน

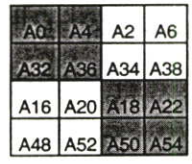
หน่วยประมวลผลที่ 1 (P_1) ทำการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 1 ด้วยวิธี VR



แบ่งข้อมูลเป็น 4 ส่วนแล้วทำการ



รวมข้อมูลทั้ง 4 ส่วน

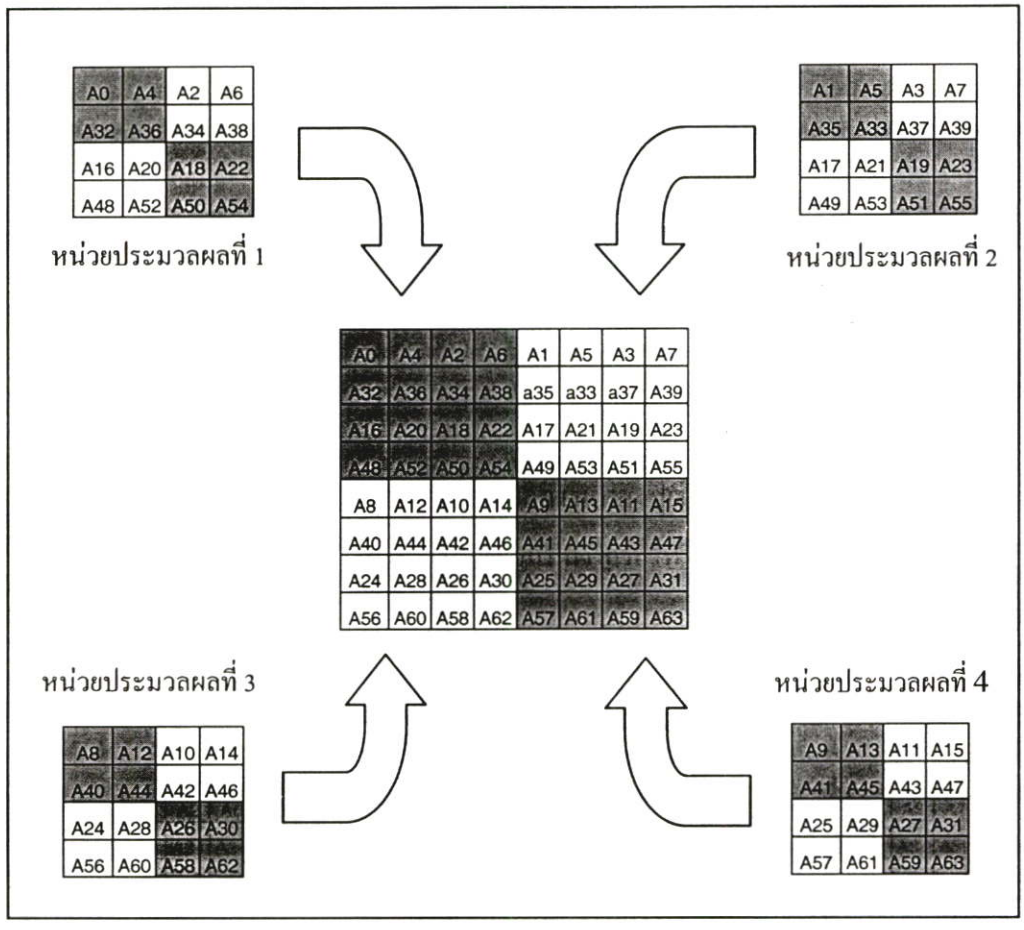


หน่วยประมวลผลที่ 2 (P_2) ทำการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 2 ด้วยวิธี VR เหมือนกับหน่วยประมวลผลที่ 1

หน่วยประมวลผลที่ 3 (P_3) ทำการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 3 ด้วยวิธี VR เหมือนกับหน่วยประมวลผลที่ 1

หน่วยประมวลผลที่ 4 (P_4) ทำการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 4 ด้วยวิธี VR เหมือนกับหน่วยประมวลผลที่ 1

ขั้นตอนที่ 4 แต่ละหน่วยประมวลผลทำการส่งผลการแปลงฟูเรียร์คืนให้หน่วยประมวลผลหลัก แล้วหน่วยประมวลผลหลักทำการคำนวณผลการแปลงฟูเรียร์ด้วยวิธี VR ในรอบสุดท้าย ดังรูปที่ 3.9



รูปที่ 3.9 แสดงการรวมข้อมูลจากแต่ละหน่วยประมวลผลทำงานเพื่อให้หน่วยประมวลผลหลักทำการคำนวณด้วยวิธี VR 2-D PFFT

Start MPI

```
MPI_Init(&argc, &argv);
MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
MPI_Comm_size( MPI_COMM_WORLD, &numprocs );
```

Master Node

Create Data and Twiddle Factor

Bit Reverse Data and VR Patition Data 4 part for each Node

Send Data For Each Node

```
MPI_Send(&a,N/2*N/2, MPI_DOUBLE, dest, mtype,MPI_COMM_WORLD);
```

Worker Node

VC Computation For each Node

```
void vr(mycomplex *a,int x1,int y1,int x2,int y2)
{
        vr(&a,x1,y1,x2-size,y2-size);
        vr(&a,x1,y2-size+1,x2-size,y2);
        vr(&a,x2-size+1,y1,x2,y2-size);
        vr(&a,x2-size+1,y2-size+1,x2,y2);

}
```

For each Node send data to Master Node

Master Node

Recieve Data from Worker Node and Computation

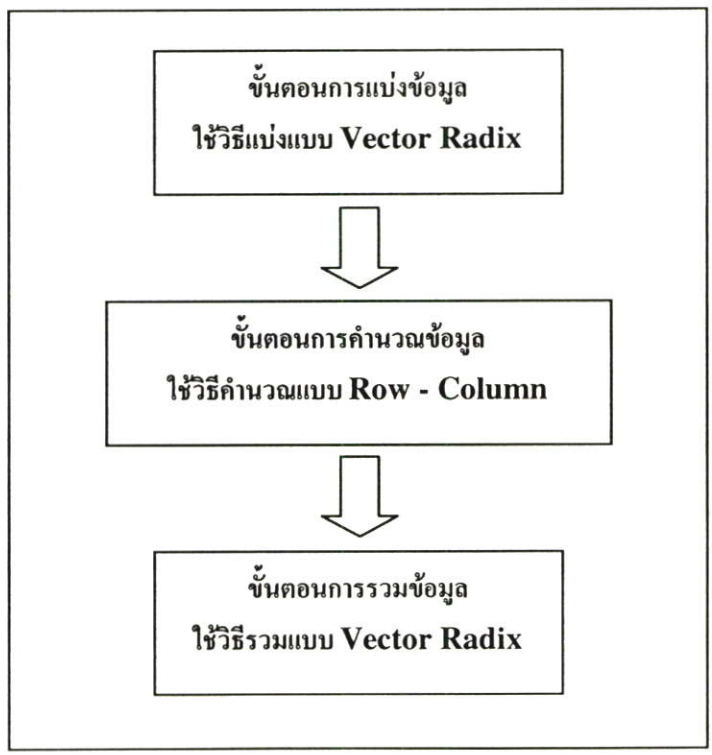
```
for(i=0;i<n/2;i++)
    for(j=0;j<n/2;j++)
        {
          ee[i][j].r=a[i][j];
          eo[i][j].r=a[i][j+n/2];
          oe[i][j].r=a[i+n/2][j];
          oo[i][j].r=a[i+n/2][j+n/2];

        }
```

รูปที่ 3.10 แสดงตัวอย่างรหัสเหมือนเอ็มพีไอ-ซี (MPI-C like) ขั้นตอนวิธี VR 2-D PFFT

3.3 การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน ด้วยขั้นตอนวิธี PBRC 2-D PFFT

จากวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธี RC ในหัวข้อ 3.1 นั้นจะพบว่าในส่วนของการทรานสโพสที่เป็นส่วนการติดต่อระหว่างหน่วยประมวลผลนั้น จะใช้เวลาในการประมวลผลในส่วนนี้มากที่สุดซึ่งมีผลต่อเวลาในการประมวลทั้งหมด ส่วนวิธี VR ในหัวข้อ 3.2 นั้นพบว่าแต่ละหน่วยประมวลผลจะติดต่อกันเฉพาะขั้นตอนในการแบ่งและรวมข้อมูลเพื่อคำนวณขั้นสุดท้ายเท่านั้น โดยส่วนที่ใช้เวลามากที่สุด คือ ขั้นตอนการคำนวณข้อมูล ดังนั้น วิธีนี้เสนอขึ้นเพื่อนำเอาส่วนที่ดีของทั้ง 2 วิธีมารวมกัน โดยให้ในขั้นตอนแรกทำการแบ่งกลุ่มข้อมูลด้วยวิธี VR และส่งข้อมูลให้แต่ละหน่วยประมวลผลทำการคำนวณ โดยในขั้นตอนการคำนวณก็ใช้วิธี RC ที่คำนวณเร็วกว่าทำการคำนวณ จากนั้นทำการส่งข้อมูลกลับให้หน่วยประมวลผลหลักทำการคำนวณค่าขั้นสุดท้ายด้วยวิธี VR ก็จะทำให้สามารถลดเวลาที่ใช้ในการประมวลผลลงได้ ดังรูปที่ 3.11

โดยในการคำนวณจะประกอบด้วยหน่วยประมวลผลซึ่งแบ่งเป็น 2 ชนิด คือ หน่วยประมวลผลหลัก (Master Computer) จำนวน 1 หน่วย และหน่วยประมวลผลทำงาน (Worker Computer) จำนวน P หน่วย (P = 2 หรือ 4)



รูปที่ 3.11 แสดงขั้นตอนการทำงานของขั้นตอนวิธี PBRC 2-D PFFT

ในส่วนของหน่วยประมวลผลหลัก (Master Computer)

ขั้นตอนที่ 1 สร้างข้อมูล คือ เมตริกซ์ A ขนาด N x N และ ค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) เก็บลงในหน่วยความจำ

ขั้นตอนที่ *2* ทำการแปลงข้อมูลให้อยู่ในรูปของอันดับบิตผกผัน (Bit Reversed order) ทั้งทางแถวและทางหลักแล้วแบ่งข้อมูลของเมตริกซ์ A เป็น 4 ส่วนซึ่งแต่ละส่วนมีขนาด k = N/2 x N/2

ขั้นตอนที่ 3 ส่งขนาดของช่วงและขนาดของเมตริกซ์ A ที่แต่ละหน่วยประมวลผลทำงานต้องทำการอ่านจากหน่วยความจำของหน่วยประมวลผลหลัก และส่งค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N)ให้ทุกหน่วยประมวลผลทำงาน

ขั้นตอนที่ 4 รอรับผลแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ จากหน่วยประมวลผลทำงาน P หน่วย เพื่อทำการคำนวณผลการแปลงฟูเรียร์ด้วยวิธี VR ในรอบสุดท้าย

ในส่วนหน่วยประมวลผลทำงาน จำนวน P หน่วย (P = 2 หรือ 4)

ขั้นตอนที่ 1 รอรับการติดต่อจากหน่วยประมวลผลหลักและหลังจากนั้น อ่านข้อมูลบางส่วนของเมตริกซ์ A ขนาด k = N/2 x N/2 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N) ทั้งหมดจากหน่วยความจำของหน่วยประมวลผลหลัก

ขั้นตอนที่ 2 ทำการคำนวณแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติด้วยวิธี RC

ขั้นตอนที่ 3 ส่งผลแปลงฟูเรียร์อย่างเร็วแบบ 2 มิติ กลับไปยังหน่วยประมวลผลหลัก

ซึ่งสามารถสรุปขั้นตอนการทำงานของขั้นตอนวิธี PBRC 2-D PFFT ได้ดังนี้ ในส่วนของหน่วยประมวลผลหลัก จะใช้ขั้นตอนเหมือนกับขั้นตอนวิธี RC 2-D PFFT และส่วนของหน่วยประมวลผลทำงานนั้นจะมีขั้นตอนเหมือนกับขั้นตอนวิธี VR 2-D PFFT

ตัวอย่างที่3.3 แสดงขั้นตอนการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธี PBRC โดยใช้จำนวนหน่วยประมวลผล 4 หน่วยประมวลผล เมื่อข้อมูลเป็นเมตริกซ์ขนาด 8 × 8 ดังรูป 3.12

เมตริกซ์ A

a0	a1	a2	a3	a4	a5	a6	a7
a8	a9	a10	a11	a12	a13	a14	a15
a16	a17	a18	a19	a20	a21	a22	a23
a24	a25	a26	a27	a28	a29	a30	a31
a32	a33	a34	a35	a36	a37	a38	a39
a40	a41	a42	a43	a44	a45	a46	a47
a48	a49	a50	a51	a52	a53	a54	a55
a56	a57	a58	a59	a60	a61	a62	a63

Complex n^{th} of root of unity(ω_N)

ω_N^0
ω_N^1
ω_N^2
ω_N^3
ω_N^4
ω_N^5
ω_N^6
ω_N^7

รูปที่ 3.12 แสดงเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N)

วิธีทำ

ขั้นตอนที่ 1 หน่วยประมวลผลหลักทำการเตรียมข้อมูลของเมตริกซ์ A ให้อยู่ในรูปการสลับบิต (Bit Reverse Order) ทั้งแถวและหลักของข้อมูล ดังรูปที่ 3.13

เมตริกซ์ A

a0	a1	a2	a3	a4	a5	a6	a7
a8	a9	a10	a11	a12	a13	a14	a15
a16	a17	a18	a19	a20	a21	a22	a23
a24	a25	a26	a27	a28	a29	a30	a31
a32	a33	a34	a35	a36	a37	a38	a39
a40	a41	a42	a43	a44	a45	a46	a47
a48	a49	a50	a51	a52	a53	a54	a55
a56	a57	a58	a59	a60	a61	a62	a63

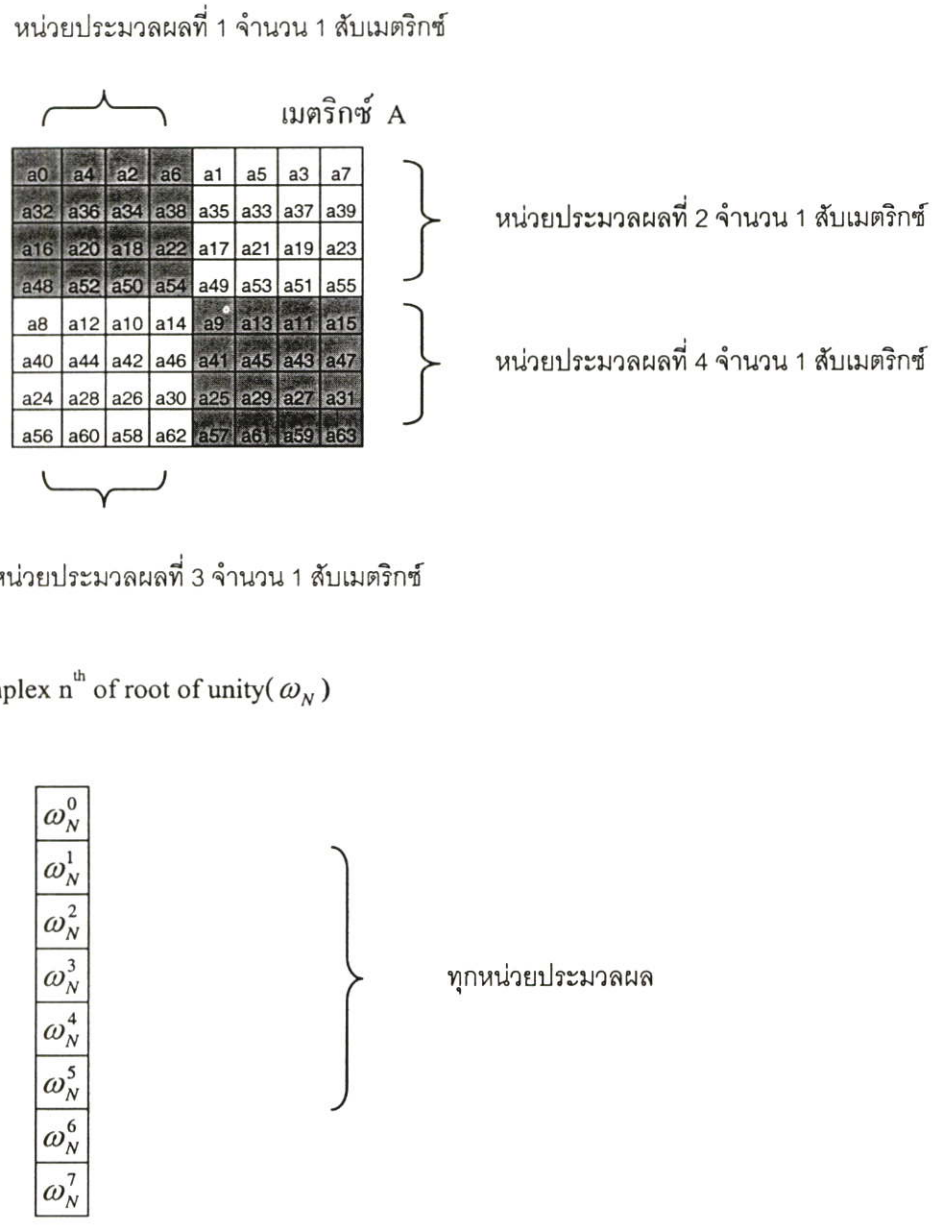
Bit Reverse

เมตริกซ์ A

a0	a4	a2	a6	a1	a5	a3	a7
a32	a36	a34	a38	a35	a33	a37	a39
a16	a20	a18	a22	a17	a21	a19	a23
a48	a52	a50	a54	a49	a53	a51	a55
a8	a12	a10	a14	a9	a13	a11	a15
a40	a44	a42	a46	a41	a45	a43	a47
a24	a28	a26	a30	a25	a29	a27	a31
a56	a60	a58	a62	a57	a61	a59	a63

รูปที่ 3.13 แสดงการสลับบิต (Bit Reverse Order) เมตริกซ์ A

ขั้นตอนที่ 2 หน่วยประมวลผลหลักแบ่งข้อมูลของเมตริกซ์ A ออกเป็นกลุ่มย่อยจำนวน 4 กลุ่มเพื่อทำการส่งข้อมูลให้หน่วยประมวลผลทำงานหน่วยละ 1 กลุ่มและข้อมูลทั้งหมดของค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N)ให้แต่ละหน่วยประมวลผล เมื่อ i= 1,2,...,p และ k = $\lceil n^2/p \rceil$ โดยกรณีนี้ p=4 n=8 และ k=16 ดังรูปที่ 3.14

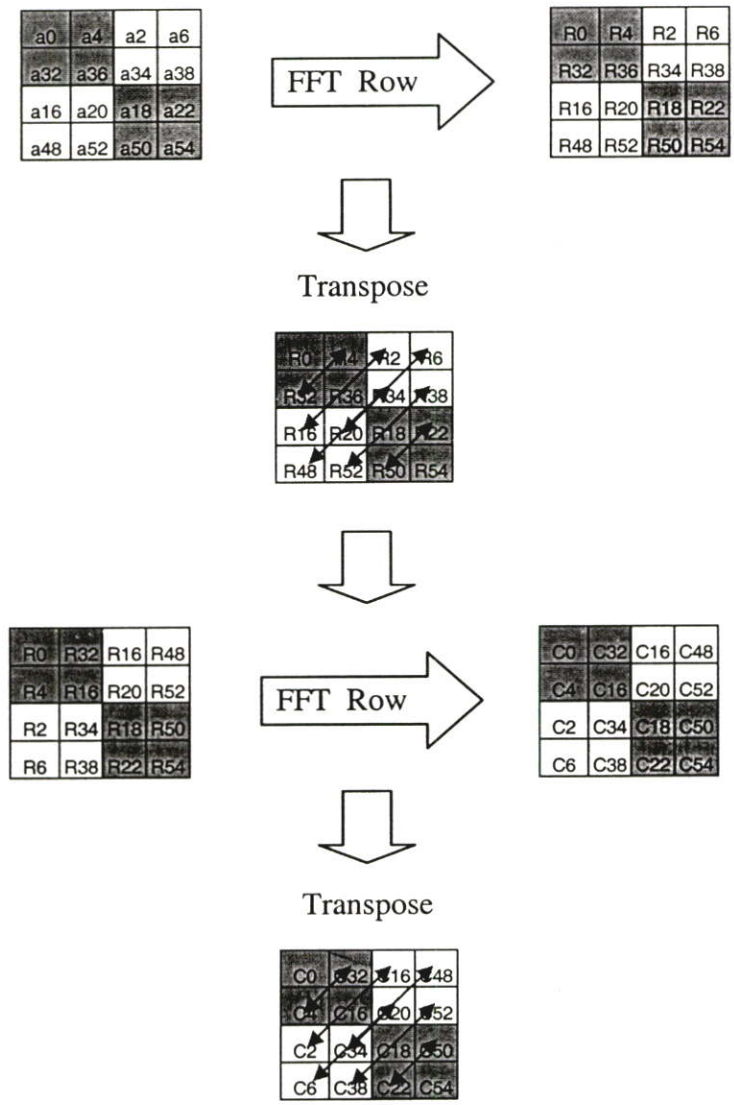


Complex n^{th} of root of unity(ω_N)

รูปที่ 3.14 แสดงการแบ่งข้อมูลเมตริกซ์ A ขนาด 8 × 8 และค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน

ขั้นตอนที่ 3 หน่วยประมวลผลแต่ละหน่วยทำการอ่านข้อมูลสับเมตริกซ์ A และข้อมูลค่ารากที่ N ของ 1 ของจำนวนเชิงซ้อน (ω_N)จากนั้นคำนวณหาผลการแปลงฟูเรียร์ด้วยวิธี RC ไปพร้อมๆ กัน ซึ่งในขั้นตอนการคำนวณนั้นจะไม่มีการติดต่อระหว่างหน่วยประมวลผล

หน่วยประมวลผลที่ 1 (P_1) ทำการคำนวณการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 1 ด้วยวิธี RC ดังรูป 3.15



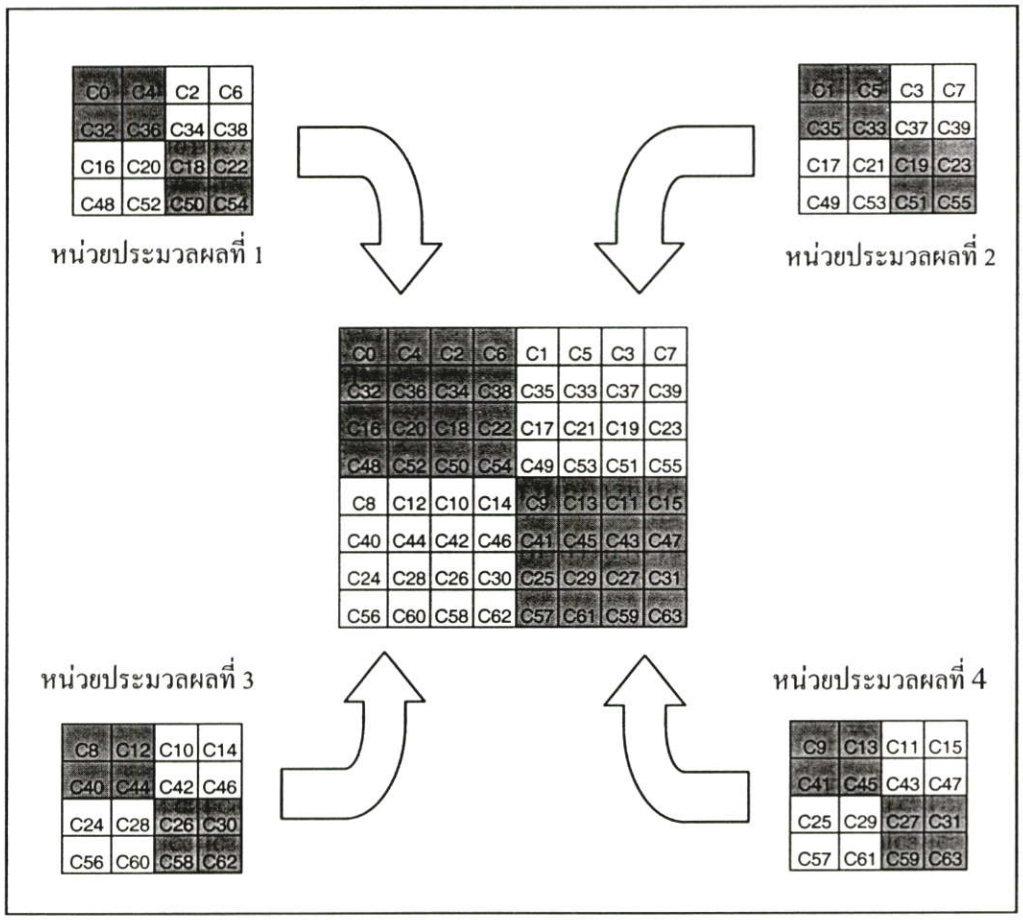
รูปที่ 3.15 แสดงการคำนวณการแปลงฟูเรียร์ด้วยวิธี RC ของหน่วยประมวลผลที่ 1

หน่วยประมวลผลที่ 2 (P_2) ทำการคำนวณการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 2 ด้วยวิธี RC เช่นเดียวกับหน่วยประมวลผลที่ 1

หน่วยประมวลผลที่ 3 (P_3) ทำการคำนวณการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 3 ด้วยวิธี RC เช่นเดียวกับหน่วยประมวลผลที่ 1

หน่วยประมวลผลที่ 4 (P_4) ทำการคำนวณการแปลงฟูเรียร์ของสับเมตริกซ์ที่ 4 ด้วยวิธี RC เช่นเดียวกับหน่วยประมวลผลที่ 1

ขั้นตอนที่ 4 แต่ละหน่วยประมวลผลทำการส่งผลการแปลงฟูเรียร์ด้วยวิธี RC คืนให้หน่วยประมวลผลหลัก เพื่อให้หน่วยประมวลผลหลักทำการคำนวณผลการแปลงฟูเรียร์ด้วยวิธี VR รอบสุดท้าย ดังรูปที่ 3.16



รูปที่ 3.16 แสดงการรวมข้อมูลจากแต่ละหน่วยประมวลผลทำงานเพื่อให้หน่วยประมวลผลหลักทำการคำนวณด้วยวิธีเวกเตอร์เรดิกซ์

```
Start MPI

 MPI_Init(&argc, &argv);
 MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
 MPI_Comm_size( MPI_COMM_WORLD, &numprocs );

Master Node

   Create Data

   VR Patition Data

   Send Data For Each Node

Slave Node

 RC Computation For each Node

     Perform FFT Computation on Colomn of Data
        for (i=0;i<IMAGE_SLICE;i++)
           fft(&Input[i][0], w_common, IMAGE_SIZE, logn);

     Transpose Data in local Node

     Perform FFT Computation on Row of Data
        for (i=0;i<IMAGE_SLICE;i++)
          fft(&Input [i][0], w_common, IMAGE_SIZE, logn);

 For each Node send data to Master Node

Master Node

   VR Computation
```

รูปที่ 3.17 แสดงตัวอย่างรหัสเหมือนเอ็มพีไอ-ซี (MPI-C like) ขั้นตอนวิธี PBRC 2-D PFFT

3.4 การวิเคราะห์ความซับซ้อนด้านเวลา (Time Complexity Analysis)

ในการวิเคราะห์ความซับซ้อนด้านเวลาของขั้นตอนวิธีแบบขนาน (Time Complexity of a Parallel Algorithm) บนระบบคลัสเตอร์นั้น เวลาทั้งหมดที่ใช้ในการประมวลผลแบบขนาน (Parallel Execution Time) จะแบ่งออกเป็นเวลา 2 ส่วน คือ เวลาที่ใช้ในการคำนวณ (Computation Time) และ เวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล (Communication Time) โดยมีรูปแบบดังนี้

$$t_p = t_{comm} + t_{comp}$$

เมื่อ t_p คือ เวลาทั้งหมดที่ใช้ในการประมวลผลแบบขนาน

t_{comm} คือ เวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล

t_{comp} คือ เวลาที่ใช้ในการคำนวณ

ซึ่งรูปสมการทั่วไปของเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผลเป็นดังนี้

$$t_{comm} = t_{startup} + Nt_{data}$$

เมื่อ $t_{startup}$ คือ ค่าคงที่ของเวลาในการจัดการกับข้อความ และเริ่มทำการส่ง

t_{data} คือ ค่าคงที่ของเวลาที่ใช้ในการส่งข้อมูล 1 ชิ้น

N คือ จำนวนข้อมูล

ในงานวิจัยนี้ได้นำเสนอวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้งหมด 3 วิธี คือ RC 2-D PFFT , VR 2-D PFFT และ PBRC 2-D PFFT ซึ่งสามารถทำการวิเคราะห์ความซับซ้อนด้านเวลาของขั้นตอนวิธีแบบขนานของแต่ละขั้นตอนวิธี เมื่อกำหนดให้ข้อมูลมีขนาด $N \times N$ และมีจำนวนหน่วยประมวลผลเป็น P หน่วยประมวลผล ได้ดังนี้

3.4.1 ความซับซ้อนด้านเวลาของขั้นตอนวิธี RC 2-D PFFT

ขั้นตอนวิธี RC 2-D PFFT จะมีขั้นตอนทั้งหมด 6 ขั้นตอน ซึ่งสามารถคำนวณความซับซ้อนด้านของขั้นตอนวิธีแบบขนาน (Time Complexity of a Parallel Algorithm) ได้ดังนี้

ขั้นตอนที่ 1 หน่วยประมวลผลหลัก (Master Processor) ทำการเตรียมข้อมูลเพื่อแบ่งให้แต่ละหน่วยประมวลผลทำงาน (Worker Processor) จำนวน P หน่วยประมวลผล ถ้าให้ข้อมูลมีขนาด $N \times N$ จะได้ว่าแต่ละหน่วยประมวลผลทำงานจะมีจำนวนข้อมูลเป็น $\frac{N^2}{P}$ โดยมีเวลาเป็น

$$t_{comp1} = N^2 / P$$

ขั้นตอนที่ 2 หน่วยประมวลผลหลักทำการส่งข้อมูลขนาด N^2 / P ให้แต่ละหน่วยประมวลผลทำงานโดยทำการส่งทั้งหมด P-1 ครั้ง เนื่องจากต้องคงเหลือข้อมูลจำนวนหนึ่งให้หน่วยประมวลผลหลักทำงานด้วย ซึ่งจะได้เวลาเป็น

$$t_{comm1} = (P-1)(t_{startup} + (N^2 / P)t_{data})$$

ขั้นตอนที่ 3 แต่ละหน่วยประมวลผลทำการคำนวณวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติด้วยขั้นตอนวิธีแถวและหลัก (Row-Column Algorithm) ซึ่งจะไม่มีการติดต่อสื่อสารระหว่างกันในส่วนนี้ โดยได้เวลาเป็น

$$t_{comp2} = (N^2 / P)\log(N^2 / P)$$

ขั้นตอนที่ 4 ทำการทรานสโพสข้อมูลทั้งหมดโดยทุกหน่วยประมวลผลทำการแลกเปลี่ยนข้อมูลระหว่างกัน ซึ่งจะได้เวลาเป็น

$$t_{comm2} = P(P-1)(t_{startup} + (N^2 / P)t_{data})$$

ขั้นตอนที่ 5 แต่ละหน่วยประมวลผลทำการคำนวณวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติด้วยขั้นตอนวิธีแถวและหลัก (Row-Column Algorithm) แบบขั้นตอนที่ 3 โดยได้เวลาเป็น

$$t_{comp3} = (N^2 / P)\log(N^2 / P)$$

ขั้นตอนที่ 6 หน่วยประมวลผลหลักทำการรวบรวมข้อมูลขนาด N^2 / P จากแต่ละหน่วยประมวลผลทำงาน ซึ่งจะได้เวลาเป็น

$$t_{comm3} = (P-1)(t_{startup} + (N^2 / P)t_{data})$$

ขั้นตอนที่ 4 หน่วยประมวลผลหลักทำการรวบรวมข้อมูลขนาด N^2/P จากแต่ละหน่วยประมวลผลทำงาน ซึ่งจะได้เวลาเป็น

$$t_{comm3} = (P-1)(t_{startup} + (N^2/P)t_{data})$$

ขั้นตอนที่ 5 หน่วยประมวลผลหลักทำการคำนวณข้อมูลทั้งหมดด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ ซึ่งจะได้เวลาเป็น

$$t_{comp3} = \frac{3}{4}N^2$$

ดังนั้น จะได้ว่าเวลาทั้งหมด (Total Parallel Execution Time) ที่ใช้สำหรับขั้นตอนวิธี VR 2-D PFFT เป็น

$$t_p = N^2/P + (P-1)(t_{startup} + (N^2/P)t_{data}) + \frac{3}{4}(N^2/P)\log(N^2/P) +$$
$$(N^2/P) + (P-1)(t_{startup} + (N^2/P)t_{data}) + \frac{3}{4}N^2$$

3.4.3 ความซับซ้อนด้านเวลาของขั้นตอนวิธี PBRC 2-D PFFT

ขั้นตอนวิธี PBRC 2-D PFFT จะมีขั้นตอนทั้งหมด 5 ขั้นตอน ซึ่งสามารถคำนวณความซับซ้อนด้านเวลาของขั้นตอนวิธีแบบขนาน (Time Complexity of a Parallel Algorithm) ได้ดังนี้

ขั้นตอนที่ 1 หน่วยประมวลผลหลัก (Master Processor) ทำการเตรียมข้อมูลเพื่อแบ่งให้แต่ละหน่วยประมวลผลทำงาน (Worker Processor) จำนวน P หน่วยประมวลผล ถ้าให้ข้อมูลมีขนาด $N \times N$ จะได้ว่าแต่ละหน่วยประมวลผลทำงานจะมีจำนวนข้อมูลเป็น $\frac{N^2}{P}$ ดังนี้

$$t_{comp1} = N^2/P$$

ขั้นตอนที่ 2 หน่วยประมวลผลหลักทำการส่งข้อมูลขนาด N^2/P ให้แต่ละหน่วยประมวลผลทำงานโดยทำการส่งทั้งหมด P-1 ครั้ง เนื่องจากต้องคงเหลือข้อมูลจำนวนหนึ่งให้หน่วยประมวลผลหลักทำงานด้วย ซึ่งจะได้เวลาเป็น

$$t_{comm1} = (P-1)(t_{startup} + (N^2/P)t_{data})$$

ขั้นตอนที่ 3 แต่ละหน่วยประมวลผลทำการคำนวณวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติด้วยขั้นตอนวิธีแถวและหลัก (Row-Column Algorithm) ซึ่งจะไม่มีการติดต่อสื่อสารระหว่างกันของการทราสโพส

ในส่วนนี้ โดยสามารถแบ่งการคำนวณเป็น 2 ส่วน คือ ส่วนการแปลงฟูเรียร์ทั้งทางแถวและหลัก ซึ่งได้เวลาเป็น $2(N^2/P)\log(N^2/P)$ และส่วนทรานสโพสได้เวลาเป็น N^2/P ดังนั้น จะได้เวลาในการคำนวณทั้งหมดเป็น

$$t_{comp2} = 2(N^2/P)\log(N^2/P) + N^2/P$$

ขั้นตอนที่ 4 หน่วยประมวลผลหลักทำการรวบรวมข้อมูลขนาด N^2/P จากแต่ละหน่วยประมวลผลทำงาน ซึ่งจะได้เวลาเป็น

$$t_{comm3} = (P-1)(t_{startup} + (N^2/P)t_{data})$$

ขั้นตอนที่ 5 หน่วยประมวลผลหลักทำการคำนวณข้อมูลทั้งหมดด้วยขั้นตอนวิธีเวกเตอร์เรดิกซ์ (Vector Radix Algorithm) ซึ่งจะได้เวลาเป็น

$$t_{comp3} = \frac{3}{4}N^2$$

ดังนั้น จะได้ว่าเวลาทั้งหมดที่ใช้ในการประมวลผล (Total Parallel Execution Time) ที่ใช้สำหรับขั้นตอนวิธี PBRC 2-D PFFT เป็น

$$t_p = N^2/P + (P-1)(t_{startup} + (N^2/P)t_{data}) + 2(N^2/P)\log(N^2/P) + N^2/P + (P-1)(t_{startup} + (N^2/P)t_{data}) + \frac{3}{4}N^2$$

ซึ่งสามารถนำมาสรุปจำนวนการดำเนินการคำนวณ (Number of Arithmetic Operation) เวลาทั้งหมดที่ใช้ในการประมวลผลของทั้ง 3 ขั้นตอนวิธี ได้ดังตารางที่ 3.1 และ 3.2

ตารางที่ 3.1 แสดงการเปรียบเทียบจำนวนการดำเนินการคำนวณ (Number of Arithmetic Operation) ของขั้นตอนวิธีทั้ง 3

ขั้นตอนวิธี	จำนวนการดำเนินการคำนวณ		
	การคูณ	การบวก	การดำเนินการกับข้อมูล
RC 2-D PFFT	$N^2 \log_2 N$	$2N^2 \log_2 N$	$2N \log_2 N$
VR 2-D PFFT	$\frac{3}{4}N^2 \log_2 N$	$2N^2 \log_2 N$	$2N + 2N \log_2 N$
PBRC 2-D PFFT	$\frac{3}{4}N^2 + N^2 \log_2 \frac{N}{2}$	$2N^2 \log_2 N$	$2N + 2N \log_2 N$

ตารางที่ 3.2 แสดงเวลาทั้งหมดที่ใช้ในการประมวลผล (Total Parallel Execution Time) ของทั้งขั้นตอนวิธี โดยแบ่งเป็นเวลาที่ใช้ในการคำนวณ และ เวลาที่ใช้ในการติดต่อสื่อสาร

ขั้นตอนวิธี	เวลาทั้งหมดที่ใช้ในการประมวลผล	
	เวลาที่ใช้ในการคำนวณ	เวลาที่ใช้ในการติดต่อสื่อสาร
RC 2-D PFFT	$N^2/P + 2(N^2/P)\log(N^2/P)$	$2(P-1)(t_{startup} + (N^2/P)t_{data}) + P(P-1)(t_{startup} + (N^2/P)t_{data})$
VR 2-D PFFT	$N^2/P + \frac{3}{4}(N^2/P)\log(N^2/P) + (N^2/P) + \frac{3}{4}N^2$	$2(P-1)(t_{startup} + (N^2/P)t_{data})$
PBRC 2-D PFFT	$N^2/P + 2(N^2/P)\log(N^2/P) + N^2/P + \frac{3}{4}N^2$	$2(P-1)(t_{startup} + (N^2/P)t_{data})$

จากตารางที่ 3.1 และ 3.2 จะพบว่าขั้นตอนวิธี VR 2-D PFFT แม้จะใช้ปริมาณการคูณน้อยกว่าวิธี RC 2-D PFFT อยู่ 25% แต่ในการทำงานแบบอนุกรม (Sequential Execution) นั้น วิธี VC 2-D PFFT จะต้องทำการแบ่งข้อมูลแบบควาดทรี (Quadtree) โดยจะทำการแบ่งออกทีละ 4 ส่วน ซึ่งการแบ่งปัญหาใหญ่ออกเป็นปัญหาย่อยๆ หลายๆ ปัญหานั้น ไม่ได้ช่วยให้การทำงานนั้นเร็วขึ้น ตรงกันข้ามมีโอกาสสูงมากที่จะทำให้การทำงานช้าลงมาก เนื่องมาจากต้องมีการจัดการกับเนื้อที่หน่วยความจำแบบสแต็ก (Stack) ในทุกๆครั้งที่ฟังก์ชั่นมีการเรียกตัวเองและเมื่อตอนย้อนกลับ แต่สำหรับโปรแกรมแบบขนานแล้ว การแบ่งปัญหาใหญ่ออกเป็นปัญหาย่อยในจำนวนมากเท่าใด ก็ยิ่งเป็นการเพิ่มให้มีการทำงานแบบขนานมาขึ้นเท่านั้น เนื่องจากปัญหาย่อยแต่ละปัญหาจะถูกแก้ไขโดยหน่วยประมวลผลหลายๆหน่วยพร้อมๆกัน ทำให้ช่วยลดเวลาที่ใช้ในการประมวลผล

ดังนั้น จึงได้ทำการพัฒนาขั้นตอนวิธี PBRC 2-D PFFT ขึ้นมาเพื่อแก้ปัญหาขั้นตอนการทำงานที่ซับซ้อนและใช้เวลาในการประมวลผลมากในการใช้ฟังก์ชั่นมีการเรียกตัวเองของขั้นตอนวิธี VR 2-D PFFT โดยจะนำเอาวิธีการคำนวณของขั้นตอนวิธี RC 2-D PFFT มาใช้แทนซึ่งจะใช้ปริมาณการคูณน้อยกว่าขั้นตอนวิธี RC 2-D PFFT อยู่ $\frac{N^2}{4}$ โดยที่ในส่วนขั้นตอนการติดต่อสื่อสารระหว่างหน่วยประมวลผลยังคงใช้ของวิธี VR 2-D PFFT ที่ใช้เวลาน้อยกว่าวิธี RC 2-D PFFT

บทที่ 4

การทดลองและผลการทดลอง

งานวิจัยนี้เป็นการนำเสนอการเพิ่มสมรรถนะ (Performance) ของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยการลดเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวล ซึ่งได้ทำการทดลองทั้งหมด 3 วิธี ดังนี้ 1) วิธีอาร์ซีทูดีพีเอฟเอฟที (Row Column Two-Dimensional Parallel Fast Fourier Transform : RC 2-D PFFT) 2) วิธีวีอาร์ทูดีพีเอฟเอฟที (Vector Radix Two-Dimensional Parallel Fast Fourier Transform : VR 2-D PFFT) และ 3) วิธีพีบีอาร์ซีทูดีพีเอฟเอฟที (Partition Block Row Column Two-Dimensional Parallel Fast Fourier Transform : PBRC 2-D PFFT) ดังที่ได้เสนอขั้นตอนวิธีไว้แล้วในบทที่ 3 โดยจะนำเอาวิธีทั้ง 3 มาพัฒนาโปรแกรมบนระบบพีซีคลัสเตอร์และทำการการเสนอผลการทดลองของทั้ง 3 วิธีเปรียบเทียบกัน

4.1 ระบบคอมพิวเตอร์คลัสเตอร์ที่ใช้ในการทดลอง

ระบบคลัสเตอร์ที่ใช้ในการทดลอง เป็นระบบคลัสเตอร์แบบ "Homogenous Cluster" ที่มีส่วนประกอบ ประสิทธิภาพในการประมวลผล และภาระงานที่รับผิดชอบในแต่ละเครื่องมีจำนวนเท่ากัน นอกจากนั้นในขณะทำการทดลองระบบคลัสเตอร์จะไม่มีภาระงานอื่นใดประมวลผลอยู่ในระบบคลัสเตอร์ที่ใช้ในการทดลอง มีส่วนประกอบพื้นฐานของระบบซึ่งจะประกอบด้วยส่วนประกอบด้านฮาร์ดแวร์ ระบบปฏิบัติการและซอฟท์แวร์ ที่ทำงานในระดับของแกน (Kernel) ของระบบปฏิบัติการซึ่งเรียกซอฟท์แวร์นี้ว่า "คลัสเตอร์มิดเดิลแวร์" และสุดท้ายคือการเชื่อมต่อแต่ละโหนดข้อมูลเข้าด้วยกันผ่านทางเครือข่ายความเร็วสูง [22]

1) ส่วนประกอบด้านฮาร์ดแวร์

โครงสร้างทางฮาร์ดแวร์ของระบบที่ใช้ในการทดลอง เป็นคลัสเตอร์ที่มีโครงสร้างเหมือนกัน (Homogenous Cluster) คือหน่วยประมวลผล ขนาดของหน่วยความจำ และส่วนประกอบอื่นๆ เหมือนกัน ระบบที่ใช้ในการทดลองเป็นระบบคลัสเตอร์ที่พัฒนาขึ้นโดยสำนักวิจัยและบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยระบบในปัจจุบันนี้ ประกอบด้วยคอมพิวเตอร์ส่วนบุคคลจำนวน 5 เครื่อง โดยใช้เครื่องคอมพิวเตอร์

ชนิดที่มีสองหน่วยประมวลผลในเครื่องเดียวกัน (Dual CPU) ซึ่งมีองค์ประกอบด้านฮาร์ดแวร์ที่จำเป็นในการพิจารณาเพื่อสร้างเป็นระบบคลัสเตอร์ต่างๆ ดังนี้

ก) แผงวงจรรวม (Mother Board) หรือ System Board

ข) หน่วยประมวลผล (CPUs) ในงานวิจัยนี้ใช้เครื่องคอมพิวเตอร์ชนิดที่มีสองหน่วยประมวลผลในเครื่องเดียวกัน(Dual CPUs) ยี่ห้ออินเทล (Intel Xeon) ความเร็วต่อรอบ 2.4 กิกะเฮิร์ต

ค) หน่วยความจำหลัก (RAM) ใช้ 1 กิกะไบต์

ง) หน่วยความจำสำรอง (Disk Storage) มีขนาดเท่ากับ 100 กิกะไบต์

จ) หน่วยความจำแคช (Cache Memory) ขนาด 512 kb

ฉ) เชื่อมต่อผ่านเครือข่ายความเร็ว 100 เมกกะบิตต่อวินาที (Mbps)

2) ระบบปฏิบัติการ

ระบบปฏิบัติการจะเป็นส่วนสำคัญอย่างมากต่อการทำงานของคอมพิวเตอร์และระบบคลัสเตอร์เพราะคอมพิวเตอร์ในระบบคลัสเตอร์ต้องสามารถทำงานเองได้โดยอิสระไม่ขึ้นกับเงื่อนไขของเครื่องอื่นถึงแม้มีเครื่องใดเครื่องหนึ่งในระบบหยุดทำงาน ระบบคลัสเตอร์ก็ยังสามารถทำงานได้ดังนั้นระบบปฏิบัติการจึงมีความจำเป็นในส่วนที่จะทำให้คอมพิวเตอร์แต่ละเครื่องสามารถทำงานโดยอิสระต่อกันได้ และสามารถติดต่อสื่อสารกันได้ ระบบปฏิบัติการที่สามารถใช้กับระบบคลัสเตอร์แบบพีซี (Cluster of PCs) มีหลากหลาย เช่น Linux, SUSE, FreeBSD, HP-UX และระบบปฏิบัติการที่สามารถใช้กับระบบคลัสเตอร์แบบเวิร์กสเตชั่น (Cluster of Workstation) คือ Tru64 UNIX, Solaris เป็นต้น ซึ่งการเลือกใช้ระบบปฏิบัติการนั้นขึ้นอยู่กับระบบคลัสเตอร์ด้วยว่าเป็นแบบใด (PCs or Workstation) นอกจากนี้ยังขึ้นอยู่กับความสะดวก ความเชี่ยวชาญและอุปกรณ์ที่เลือกใช้ อีกอย่างหนึ่งคือจะขึ้นอยู่กับซอฟท์แวร์และชุดคำสั่งที่เลือกใช้ด้วย เช่น ถ้าเลือกใช้ OpenMosix ที่ทำงานได้บนระบบลีนุกซ์เท่านั้น ก็จำเป็นต้องเลือกใช้ลีนุกต์เป็นระบบปฏิบัติการอีกทั้งฮาร์ดแวร์ที่สนับสนุนด้วย โดยส่วนใหญ่ซอฟท์แวร์และชุดคำสั่งที่ถูกพัฒนาขึ้นมาเพื่อใช้ในระบบคลัสเตอร์นี้จะทำงานเข้ากันได้กับระบบปฏิบัติการตระกูลยูนิกซ์เกือบทุกชนิดอยู่แล้ว

ระบบคลัสเตอร์ที่ใช้ในงานวิจัยนี้เป็นระบบปฏิบัติลีนุกต์รุ่นซูซี (SUSE Version) ที่ใช้บนระบบคลัสเตอร์แบบพีซีเนื่องจากมีความสามารถและรองรับการทำงานได้หลากหลาย อีกทั้งยังมีความสามารถในการเฝ้าระวัง (Monitoring) ในส่วนงานของผู้ดูแลระบบ (Administrator) เป็นแบบออนไลน์ (Online) ทำให้เราสามารถตรวจสอบระบบคลัสเตอร์ผ่านทางอินเตอร์เน็ต (Internet) ได้ตลอดเวลาอีกด้วย

3) คลัสเตอร์มิดเดิลแวร์

คลัสเตอร์มิดเดิลแวร์ คือซอฟท์แวร์ที่ทำงานในระดับเดียวกับแกน (Kernel) ของระบบปฏิบัติการ มีหน้าที่ในการกระจายงาน (Process) จากโหนดหนึ่งไปยังโหนดอื่นๆ ที่อยู่ในระบบคลัสเตอร์เดียวกัน โดยส่วนใหญ่แล้วคลัสเตอร์มิดเดิลแวร์นี้จะเป็นซอฟท์แวร์ที่เขียนเพิ่มเติมเข้าไปในแกนของระบบปฏิบัติการ เพื่อให้ทุกเครื่องที่อยู่ในระบบคลัสเตอร์เสมือนเป็นเครื่องคอมพิวเตอร์หลายหน่วยประมวลผลขนาดใหญ่ ที่เสมือนมีหน่วยความจำ หน่วยประมวลผลอยู่ที่เดียวกันเหมือนกับระบบคอมพิวเตอร์แบบ SMP (Symmetric Multi Processor) หรือแบบ MMP (Massive Multi Processor)

4) การเชื่อมต่อเครือข่าย

การสื่อสารระหว่างหน่วยประมวลผลในระบบคลัสเตอร์จะทำผ่านระบบเครือข่ายความเร็วสูงซึ่งอุปกรณ์เครือข่ายแต่ละชนิดจะมีความเร็วและราคาแตกต่างกันไป ตัวอย่างอุปกรณ์เครือข่ายที่นิยมนำมาใช้ในการสร้างระบบคลัสเตอร์ มีดังนี้

ก) Ethernet ในปัจจุบันอุปกรณ์ Ethernet นั้นได้ถูกพัฒนาให้มีความเร็วในการรับส่งข้อมูลสูงมากขึ้นจนถึงระดับกิกะบิตอีเทอร์เน็ต (Gigabit Ethernet) หรือ 10 กิกะบิตอีเทอร์เน็ต คือมีความเร็วในการส่งผ่านข้อมูลประมาณ 1-10 พันล้านบิตต่อวินาที ซึ่งเป็นแบบที่ใช้ในระบบคลัสเตอร์ที่ใช้ทำวิจัยนี้ด้วย

ข) Myrinet มีความเร็วในการส่งผ่านข้อมูลประมาณ 2 พันล้านบิตต่อวินาที และมีค่า Latency Time ต่ำกว่าเครือข่ายแบบ Ethernet มาก แต่มีราคาแพงกว่าอีเทอร์เน็ตสูงมาก

ค) Quadrics มีความเร็วในการส่งข้อมูลอยู่ที่ 340-900 MB/Second หรือประมาณ 2.65-7 กิกะบิตต่อวินาที และมีค่า Latency Time ต่ำมาก

ง) InfiniBand เป็นเทคโนโลยีที่มีความเร็วในการสื่อสารข้อมูลสูงมากถึง 5 กิกะบิตต่อวินาที และมีค่า Latency Time น้อยกว่า 10 ไมโครวินาที (Microsecond)

5) เอ็มพีไอซีเอช (MPICH)

เอ็มพีไอซีเอช (MPICH) เป็นการพัฒนาชุดคำสั่งตามมาตรฐานขึ้นมาใช้งานจริงมากที่สุด คือ มาตรฐานการส่งผ่านข้อความ (Message Passing Interface: MPI) ซึ่งสามารถพัฒนาโปรแกรมภาษาต่างๆ เช่น โปรแกรมภาษาซี (C Language) โปรแกรมภาษาฟอร์แทน (Fortran Language) และโปรแกรมภาษาจาวา (Java Language) เป็นต้น ที่สนับสนุนการโปรแกรมแบบขนาน ซึ่งมีฟังก์ชัน

พื้นฐานต่างๆ รวมทั้งมีคำสั่งใช้งานเพื่อทำการคอมไพล์โปรแกรมที่เขียนขึ้น โดยเอ็มพีไอซีเอชนี้จะสนับสนุนการพัฒนาโปรแกรมภาษาซี และโปรแกรมภาษาฟอร์แทนเท่านั้น [22]

การทดลองในงานวิจัยนี้ จะนำโปรแกรมที่ได้พัฒนาขึ้นไปประมวลผลบนระบบพีซีคลัสเตอร์ ของสำนักวิจัยและบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยคำสั่งที่ใช้ในการทดลองมีรูปแบบดังนี้

Mpirun –np <A1> 2D_PFFT <A2> <A3>

เมื่อ <A1> คือ จำนวนหน่วยประมวลผล

<A2> คือ จำนวนข้อมูลในรูปเมตริกซ์จัตุรัส

โดยมีค่าดังนี้

512 แทน ข้อมูลในรูปเมตริกซ์ขนาด 512 x 512

1024 แทน ข้อมูลในรูปเมตริกซ์ขนาด 1024 x1024

2048 แทน ข้อมูลในรูปเมตริกซ์ขนาด 2048 x 2048

4096 แทน ข้อมูลในรูปเมตริกซ์ขนาด 4096 x 4096

<A3> คือ วิธีการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติแบบขนาน

โดยมีค่าดังนี้

RC แทน วิธีอาร์ซีทูดีพีเอฟเอฟที (RC 2-D PFFT)

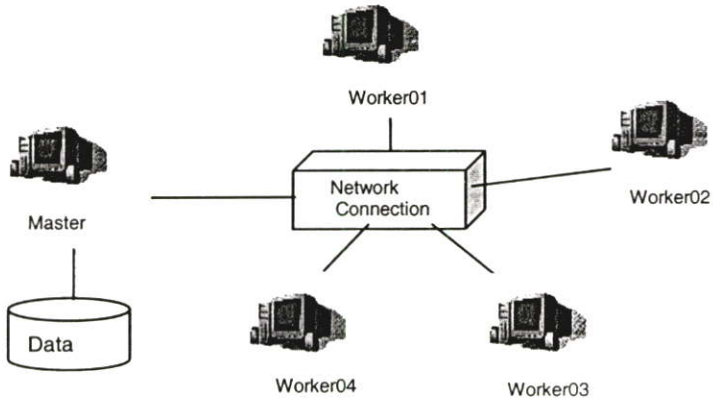
VR แทน วิธีวีอาร์ทูดีพีเอฟเอฟที (VR 2-D PFFT)

PBRC แทน วิธีพีบีอาร์ซีทูดีพีเอฟเอฟที (PBRC 2-D PFFT)

4.2 ลักษณะข้อมูลที่ใช้ในการทดลอง

ลักษณะของข้อมูลที่ใช้ในการทดลองเป็นข้อมูลชนิดเลขจำนวนจริงละเอียด 2 เท่า (Double Precision) โดยขนาดข้อมูลของเมตริกซ์จะกำหนดให้เมตริกซ์จัตุรัสมีขนาดดังนี้ คือ 512 × 512 , 1024 × 1024 , 2048 × 2048 , 4096 × 4096

งานวิจัยนี้ ข้อมูลที่ใช้ในการวัดสมรรถนะของการแปลงข้อมูลจะถูกจัดเก็บอยู่ในหน่วยความจำสำรองของหน่วยประมวลผลหลัก ซึ่งเรียกการเก็บข้อมูลแบบนี้ว่า "การเก็บข้อมูลแบบรวมศูนย์กลาง" (Centralized Data) ดังรูปที่ 4.1 โดยในงานวิจัยนี้จะทำการทดสอบบนระบบที่ให้หน่วยประมวลผลหลัก (Master Processor) ทำหน้าที่เหมือนหน่วยประมวลผลทำงาน (Worker Peocessor) คือ นอกจากทำการแบ่งข้อมูลแล้วต้องทำการประมวลผลข้อมูลด้วย



รูปที่ 4.1 ระบบคลัสเตอร์ที่มีการเก็บข้อมูลแบบรวมศูนย์กลาง

4.3 ผลการทดลอง

ในงานวิจัยนี้จะทำการเปรียบเทียบประสิทธิภาพของการแปลงฟูเรียร์แบบเร็วแบบ 2 มิติแบบขนาน โดยได้นำไปทดลองบนระบบพีซีคลัสเตอร์ ของสำนักวิจัยและบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งประกอบด้วย หน่วยประมวลผล 10 หน่วย ขนาด 2.4 GHz จำนวน 5 เครื่องหน่วยความจำหลัก 1 Gb หน่วยความจำสำรอง 100 Gb หน่วยความจำแคช (Cache memory) 512 Kb เชื่อมต่อกันด้วยความเร็วขนาด 1000 Mbps ระบบปฏิบัติการ Suse และโปรแกรม MPICH 1.2 ทำการวัดสมรรถนะ ซึ่งจะทำการประเมินผลจากเวลาทั้งหมดที่ใช้ในการประมวลผลการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนาน (Response Time) อัตราการเพิ่มขึ้นของความเร็ว (Speed Up) และประสิทธิภาพ (Efficiency) โดยในผลการทดลองนี้จะทำการเปรียบเทียบประสิทธิภาพของการแปลงฟูเรียร์แบบ 2 มิติแบบขนานทั้งหมด 3 วิธี คือ

1.) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธีอาร์ซีทูดีพีเอฟเอฟที
(Row-Column Two-Dimensional Parallel Fast Fourier Transform : RC 2-D PFFT)
2.) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธีวีอาร์ทูดีพีเอฟเอฟที
(Vector Radix Two-Dimensional Parallel Fast Fourier Transform : VR 2-D PFFT)
3.) การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานด้วยวิธีพีบีอาร์ซีทูดีพีเอฟเอฟที
(Partition Block Row-Column Two-Dimensional Parallel Fast Fourier Transform : PBRC 2-D PFFT)

ซึ่งทั้ง 3 วิธีได้ถูกพัฒนาโปรแกรมแบบขนานโดยใช้มาตรฐานภาษาเอ็มพีเอ (MPI Standard) บนระบบพีซีคลัสเตอร์ แล้วทำการตรวจสอบผลการทดลองกับการใช้ไลบรารี่มาตรฐานที่ใช้ในการแปลงฟูเรียร์แบบขนาน คือ FFTW เวอร์ชั่น 2.1.5 (ที่มา http://www.fftw.org) โดยจะใช้เฉพาะในขั้นตอนวิธีอาร์ซีทูดีพีเอฟเอฟที (RC 2-D PFFT) เท่านั้นส่วนอีก 2 ขั้นตอนวิธีคือ วิธีวีอาร์ทูดีพีเอฟเอฟที (VR 2-D PFFT) และ วิธีพีบีอาร์ซีทูดีพีเอฟเอฟที (PBRC 2-D PFFT) จะทำการพัฒนาโปรแกรมขึ้นมาใหม่แล้วทำการเปรียบเทียบทั้ง 3 ขั้นตอนวิธี

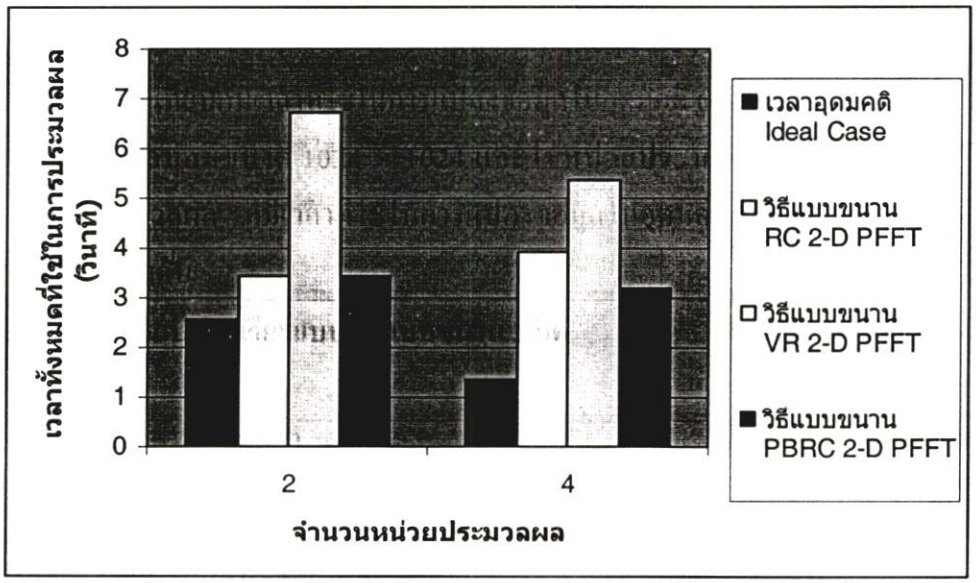
โดยในส่วนขั้นตอนการทดลองของงานวิจัยนี้ จะนำโปรแกรมที่ได้พัฒนาขึ้นไปประมวลผลบนระบบพีซีคลัสเตอร์ ของสำนักวิจัยและบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยทำการบันทึกเวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time) ซึ่งคือ เวลาตั้งแต่โปรแกรมเริ่มต้นทำการคำนวณจนกระทั่งสิ้นสุดการทำงาน แล้วทำการหาค่าเฉลี่ยของเวลาที่ประมวลผล ซึ่งเวลาส่วนนี้จะเป็นเวลาทั้งหมดที่ใช้ในการประมวลผล (Response Time) เพื่อนำเอาข้อมูลส่วนนี้ไปใช้ในการหาหาอัตราการเพิ่มของความเร็ว (Speedup) และประสิทธิภาพ (Efficiency) ของการประมวลผลแบบขนานในขั้นต่อไป

4.3.1. ผลการทดลองเปรียบเทียบกับเวลาในอุดมคติ

การทดลองเปรียบเทียบเวลาที่ประมวลผลจริงกับเวลาในอุดมคติ เป็นการวัดประสิทธิภาพของระบบ (System Performance) ที่ใช้ในการทดลองดังนี้ ตารางที่ 4.1 และรูปที่ 4.2 แสดงผลการเปรียบเทียบเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ทั้ง 3 แบบ ($T_p = T_s / P + T_c$) โดยในงานวิจัยนี้จะเน้นที่การเพิ่มประสิทธิภาพของเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล (Communication Time : T_c) ส่วนเวลาที่ใช้ในการเรียงลำดับข้อมูลในอุดมคติ (Response Time of Ideal Case) จะคำนวณโดยสมมติว่าเวลาที่ใช้ในการติดต่อสื่อสารเพื่อทำการแลกเปลี่ยนข้อมูลระหว่างหน่วยประมวลผลเท่ากับศูนย์ ($T_c = 0$) ดังนั้น จะได้ว่า $T_p = T_s / P$ ตามสมการที่ 2.10 ตารางที่ 4.2 และรูปที่ 4.3 แสดงอัตราการเพิ่มของความเร็วในอุดมคติ (Speedup) ทั้ง 3 วิธี ($S_p = T_s / T_p$) เปรียบเทียบกับอัตราการเพิ่มขึ้นของความเร็วในอุดมคติ ($S_p = P$) ในตารางที่ 4.3 และรูปที่ 4.4 แสดงการเปรียบเทียบประสิทธิภาพของการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ (Efficiency) ทั้ง 3 วิธี ($E_p = S_s / P$) เปรียบเทียบกับประสิทธิภาพในอุดมคติ ($E_p = 1$)

ตารางที่ 4.1 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ของทั้ง 3 วิธีโดยทำการเปรียบเทียบกับเวลาทีใช้ในการแปลงข้อมูลในอุดมคติ (Response Time of Ideal Case) เมื่อข้อมูลมีขนาด 1024 × 1024 และใช้หน่วยประมวลผลตั้งแต่ 2 และ 4 หน่วยประมวลผล ที่มีเวลาที่ใช้ในการแปลงข้อมูลในอุดมคติเป็น 2.56385 และ 1.37819 ตามลำดับ

วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ	จำนวนหน่วยประมวลผล	
	2	4
เวลาในอุดมคติ	2.56385	1.37819
เวลาที่ใช้ในวิธีการแปลงฟูเรียร์แบบเร็วแบบ " RC 2-D PFFT "	3.45473	3.92483
เวลาที่ใช้ในวิธีการแปลงฟูเรียร์แบบเร็วแบบ " VR 2-D PFFT "	6.70704	5.37878
เวลาที่ใช้ในวิธีการแปลงฟูเรียร์แบบเร็วแบบ " PBRC 2-D PFFT "	3.45576	3.19975

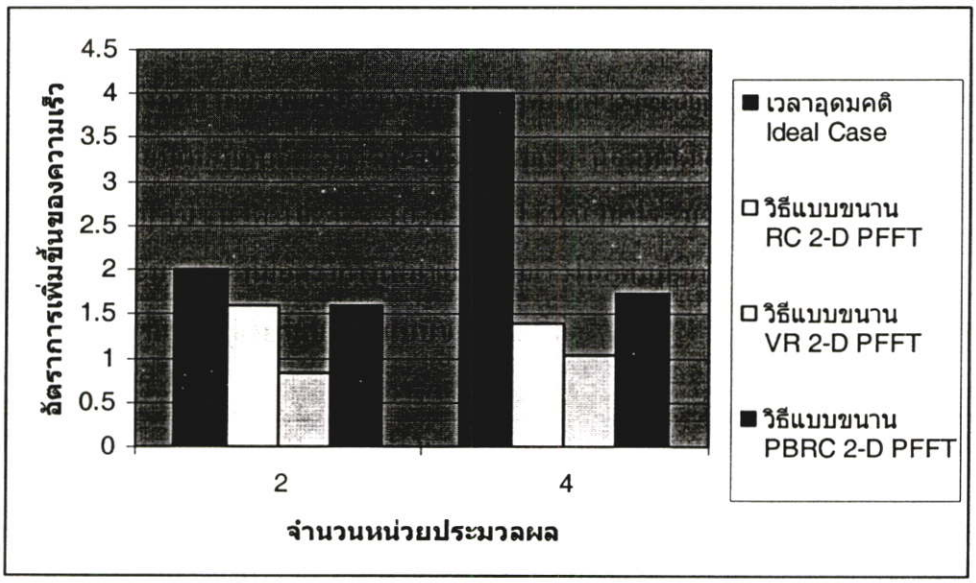


รูปที่ 4.2 แสดงกราฟเปรียบเทียบเวลาทั้งหมดที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานกับเวลาในอุดมคติ

จากรูปที่ 4.2 จะพบว่าวิธี PBRC 2-D PFFT เป็นวิธีที่ใช้เวลาทั้งหมดในการประมวลผล (Response Time) ได้ใกล้เคียงกับเวลาในอุดมคติมากที่สุด คือ 3.45576 และ 3.19975 เมื่อใช้หน่วยประมวลผล 2 และ 4 หน่วยประมวลผล ตามลำดับ

ตารางที่ 4.2 แสดงอัตราการเพิ่มของความเร็วในอุดมคติ (Speedup) ของทั้ง 3 วิธีโดยทำการเปรียบเทียบกับอัตราการเพิ่มของความเร็วในอุดมคติ (Speedup of Ideal Case) เมื่อข้อมูลมีขนาด 1024 × 1024 และใช้หน่วยประมวลผลตั้งแต่ 2 และ 4 หน่วยประมวลผล ที่มีอัตราการเพิ่มของความเร็วในอุดมคติเป็น 2 และ 4 ตามลำดับ

วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ	จำนวนหน่วยประมวลผล	
	2	4
อัตราการเพิ่มของความเร็วในอุดมคติ	2	4
อัตราการเพิ่มของความเร็วแบบ " RC 2-D PFFT "	1.59572	1.40459
อัตราการเพิ่มของความเร็วแบบ " VR 2-D PFFT "	0.82194	1.02491
อัตราการเพิ่มของความเร็วแบบ " PBRC 2-D PFFT "	1.59524	1.72288

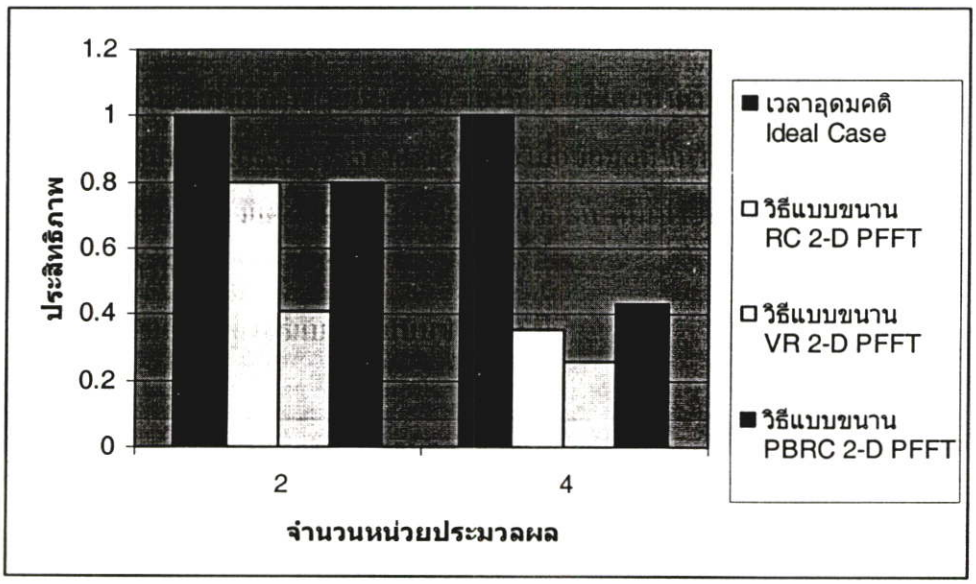


รูปที่ 4.3 แสดงกราฟเปรียบเทียบอัตราการเพิ่มของความเร็วที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานกับเวลาในอุดมคติ

จากรูปที่ 4.3 จะพบว่าวิธี PBRC 2-D PFFT เป็นวิธีที่มีอัตราการเพิ่มของความเร็ว ได้ใกล้เคียงกับเวลาในอุดมคติมากที่สุด คือ 1.59524 และ 1.72288 เมื่อใช้หน่วยประมวลผล 2 และ 4 หน่วยประมวลผล ตามลำดับ

ตารางที่ 4.3 แสดงประสิทธิภาพ (Efficiency) ของทั้ง 3 วิธีโดยทำการเปรียบเทียบกับประสิทธิภาพในอุดมคติ (Efficiency of Ideal Case) เมื่อข้อมูลมีขนาด 1024 × 1024 และใช้หน่วยประมวลผลตั้งแต่ 2 และ 4 หน่วยประมวลผล ที่มีประสิทธิภาพในอุดมคติเป็น 1 ทั้งสองกรณี

วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ	จำนวนหน่วยประมวลผล	
	2	4
ประสิทธิภาพในอุดมคติ	1	1
ประสิทธิภาพแบบ " RC 2-D PFFT "	0.79786	0.35512
ประสิทธิภาพแบบ " VR 2-D PFFT "	0.41097	0.25623
ประสิทธิภาพแบบ " PBRC 2-D PFFT "	0.79762	0.43072



รูปที่ 4.4 แสดงกราฟเปรียบเทียบประสิทธิภาพที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานกับเวลาในอุดมคติ

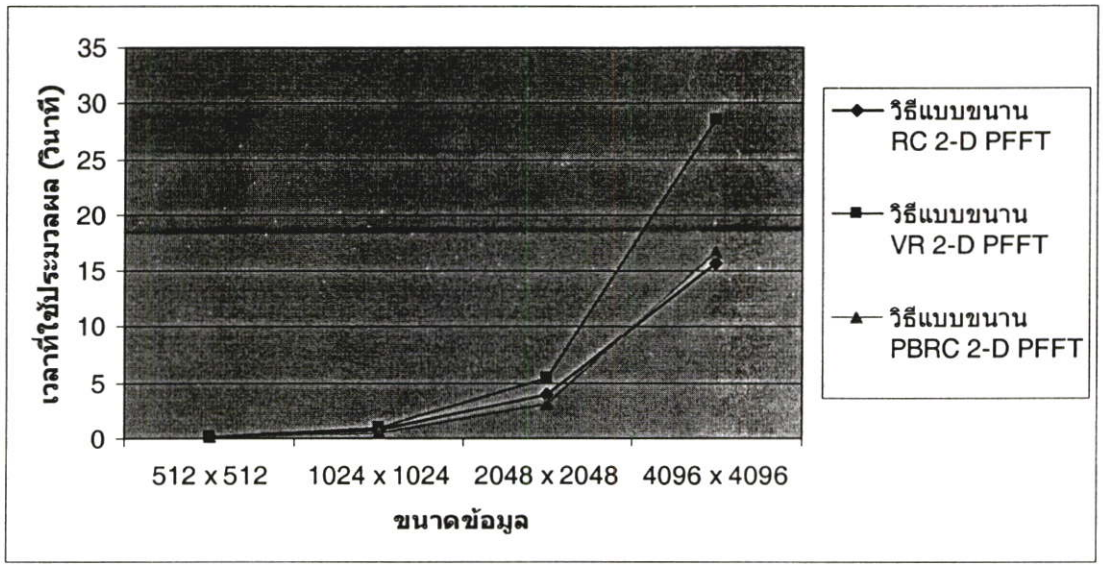
จากรูปที่ 4.4 จะพบว่าวิธี PBRC 2-D PFFT เป็นวิธีที่มีอัตราการเพิ่มของความเร็ว ได้ใกล้เคียงกับเวลาในอุดมคติมากที่สุด คือ 0.79762 และ 0.43072 เมื่อใช้หน่วยประมวลผล 2 และ 4 หน่วยประมวลผล ตามลำดับ

4.3.2 ผลการทดลองเปรียบเทียบเมื่อขนาดข้อมูลที่ใช้ทดลองต่างกัน

การทดลองเปรียบเทียบการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธีที่ได้กล่าวมาในข้างต้น สำหรับข้อมูลที่มีขนาดต่างกัน โดยทำการทดลองที่ขนาดของชุดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512×512 , 1024×1024 , 2048×2048 และ 4096×4096 ตามลำดับ และทำการกำหนดจำนวนหน่วยประมวลผลที่ใช้ในการคำนวณเท่ากับ 4 หน่วยประมวลผล ซึ่งแสดงผลการทดลองดังนี้ ตารางที่ 4.4 และรูปที่ 4.5 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ที่เป็นผลรวมของเวลาที่ใช้ในการติดต่อสื่อสารและเวลาที่ใช้ในการประมวลผล ($T_p = T_s / P + T_c$) นอกจากนี้เราสามารถนำเวลาที่ใช้ในการประมวลผลทั้งหมดมาทำการคำนวณหาอัตราการเพิ่มขึ้นของความเร็ว ดังแสดงในตารางที่ 4.5 และรูปที่ 4.6 จากนั้นนำค่าของอัตราการเพิ่มขึ้นของเร็วมาคำนวณหาประสิทธิภาพ จากนั้นนำค่าของอัตราการเพิ่มขึ้นของความเร็วมาคำนวณหาประสิทธิภาพ ตามสมการที่ 2.13 ($E_p = S_s / P$) ดังแสดงในตารางที่ 4.6 และรูปที่ 4.7

ตารางที่ 4.4 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512×512 , 1024×1024 , 2048×2048 และ 4096×4096 ตามลำดับ

ขนาดชุดข้อมูล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
512×512	0.19115	0.23946	0.13109
1024×1024	0.79670	0.97702	0.55540
2048×2048	3.92483	5.37878	3.19975
4096×4096	15.680375	28.5857	16.67015

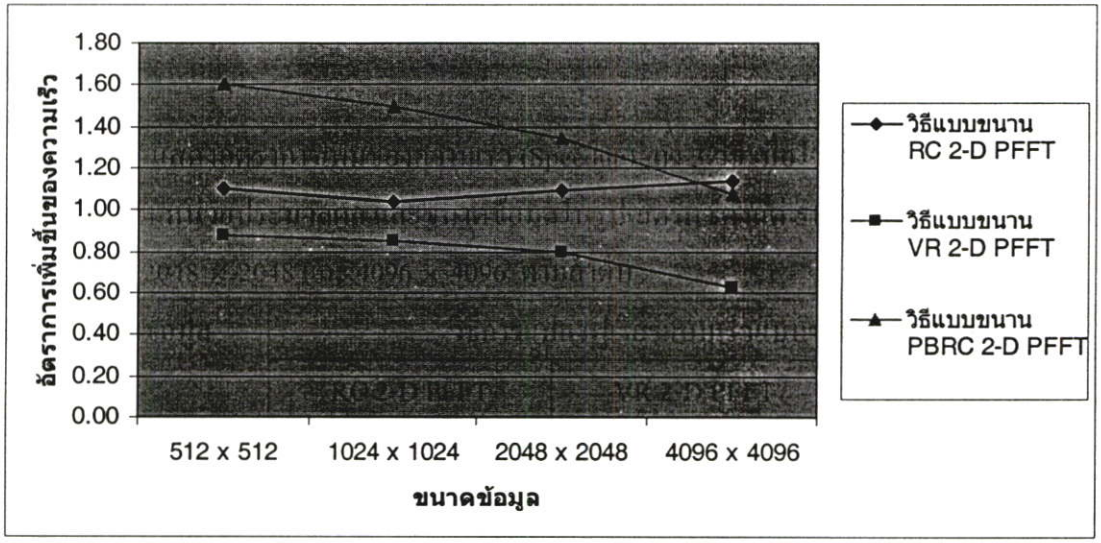


รูปที่ 4.5 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการประมวลผลทั้งหมดที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล

จากรูปที่ 4.5 แสดงเวลาทั้งหมดที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยทำการเปรียบเทียบเวลาทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผล 4 หน่วยประมวลผล ทำการแปลงข้อมูลชนิดเลขจำนวนจริงละเอียด 2 เท่า (Double Precision) ซึ่งอยู่ในรูปเมตริกซ์ 2 มิติโดยมีตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ จะเห็นว่าเมื่อข้อมูลมีขนาดใหญ่ขึ้นเวลาที่ใช้ในการแปลงข้อมูลนั้นจะเพิ่มขึ้นตามไปด้วย โดยวิธี PBRC 2-D PFFT ใช้เวลาในการประมวลผลน้อยที่สุด และวิธี VR 2-D PFFT ใช้เวลาในการประมวลผลมากกว่าวิธีอื่นๆ แต่จะพบว่าเมื่อข้อมูลมีขนาดใหญ่ถึงจุดหนึ่ง (จากการทดลองเมื่อใช้ข้อมูลขนาด 4096 × 4096) วิธี PBRC 2-D PFFT ใช้เวลาในการประมวลผลมากกว่าวิธี RC 2-D PFFT เล็กน้อย ซึ่งมีสาเหตุมาจากรูปแบบการส่งข้อมูลของระบบพีซีคลัสเตอร์ ที่เมื่อบล็อกข้อมูลมีขนาดใหญ่ขึ้นทำให้จำเป็นต้องใช้เวลาในการติดต่อสื่อสารระหว่างหน่วยประมวลผลที่มากขึ้นด้วย นอกจากนี้ยังเกี่ยวข้องกับเวลาที่ใช้ในการประมวลผลที่มากขึ้นตามขนาดของข้อมูลส่งผลให้เวลาโดยรวมมากขึ้น ซึ่งอาจแก้ปัญหาได้โดยการเพิ่มความเร็วของโครงข่าย (Network) ของระบบให้มีความเร็วเพิ่มมากขึ้น หรือนำไปพัฒนาบนเครื่องที่มีการประมวลผลแบบขนาน (Parallel Computer) ที่มีข้อจำกัดทางด้านการติดต่อสื่อสารระหว่างหน่วยประมวลผลไม่มาก จะเป็นการแก้ปัญหาในส่วนนี้ได้

ตารางที่ 4.5 แสดงอัตราการเพิ่มของความเร็ว (Speedup) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ

ขนาดชุดข้อมูล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
512 × 512	1.09924	0.87747	1.60287
1024 × 1024	1.04224	0.84989	1.49507
2048 × 2048	1.09222	0.79698	1.33972
4096 × 4096	1.13770	0.62407	1.07015

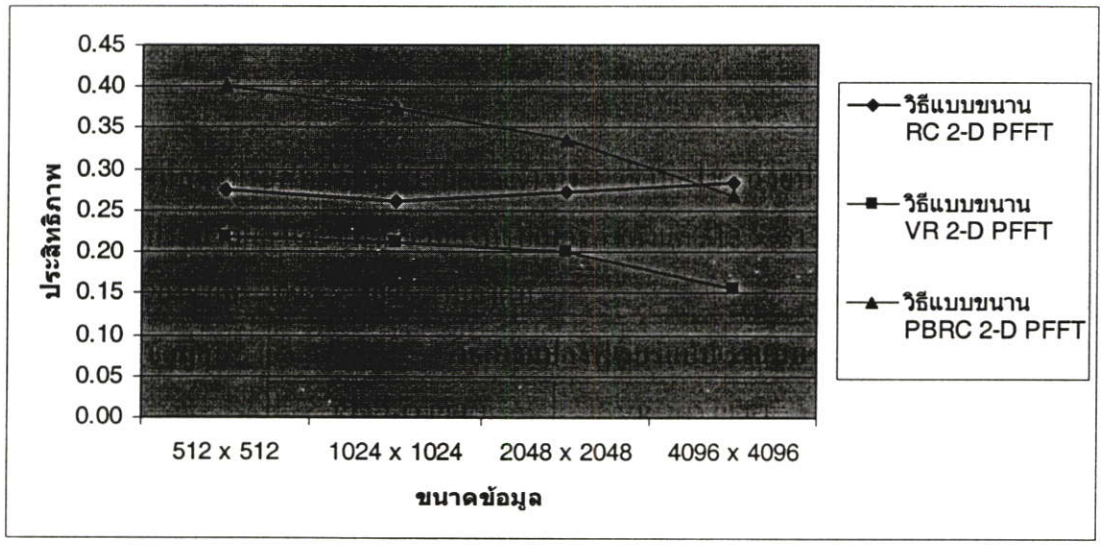


รูปที่ 4.6 แสดงกราฟเปรียบเทียบอัตราการเพิ่มของความเร็วที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติ แบบขนานทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล

จากรูปที่ 4.6 การเปรียบเทียบอัตราการเพิ่มขึ้นของความเร็วที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยทำการเปรียบเทียบเวลาทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผล 4 หน่วยประมวลผล ทำการแปลงข้อมูลชนิดเลขจำนวนจริงละเอียด 2 เท่า (Double Precision) ซึ่งอยู่ในรูปเมตริกซ์ 2 มิติ โดยมีตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ จะเห็นว่าเมื่อข้อมูลมีขนาดใหญ่ขึ้นอัตราการเพิ่มขึ้นของความเร็วจะลดลง โดยวิธี PBRC 2-D PFFT มีอัตราการเพิ่มขึ้นของความเร็วมากที่สุด และวิธี VR 2-D PFFT มีอัตราการเพิ่มขึ้นของความเร็วน้อยที่สุด แต่จะพบว่าเมื่อข้อมูลมีขนาดเป็น 4096 × 4096 แนวโน้มของกราฟของวิธี PBRC 2-D PFFT จะลดลงต่ำกว่าวิธี RC 2-D PFFT ซึ่งมีสาเหตุเช่นเดียวกับการวัดเวลาทั้งหมดที่ใช้ประมวลผลที่ได้กล่าวไว้ข้างต้น

ตารางที่ 4.6 แสดงประสิทธิภาพ (Efficiency) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ

ขนาดชุดข้อมูล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
512 × 512	0.27481	0.21937	0.40072
1024 × 1024	0.26056	0.21247	0.37377
2048 × 2048	0.27305	0.19924	0.33493
4096 × 4096	0.28443	0.15602	0.26754

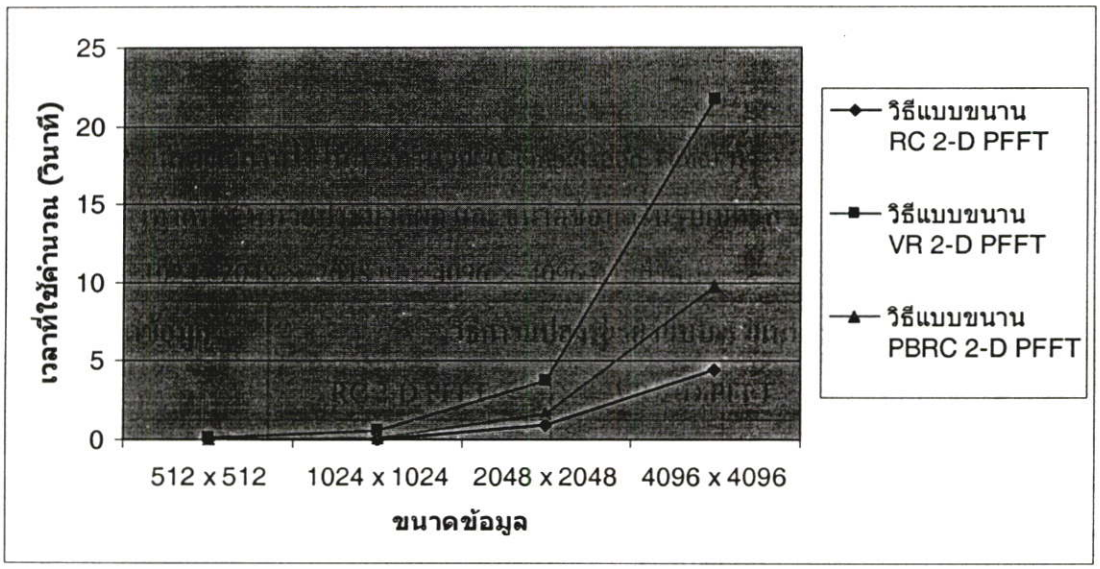


รูปที่ 4.7 แสดงกราฟเปรียบเทียบประสิทธิภาพที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล

จากรูปที่ 4.7 แสดงการเปรียบเทียบประสิทธิภาพที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยทำการเปรียบเทียบเวลาทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผล 4 หน่วยประมวลผล ทำการแปลงข้อมูลชนิดเลขจำนวนจริงละเอียด 2 เท่า (Double Precision) ซึ่งอยู่ในรูปเมตริกซ์ 2 มิติ โดยมีตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ จะเห็นว่าเมื่อข้อมูลมีขนาดใหญ่ขึ้นประสิทธิภาพที่ใช้ในการแปลงข้อมูลนั้นจะลดลง โดยวิธี PBRC 2-D PFFT มีประสิทธิภาพมากที่สุด และวิธี VR 2-D PFFT มีประสิทธิภาพน้อยที่สุด แต่จะพบว่าเมื่อข้อมูลมีขนาดเป็น 4096 × 4096 แนวโน้มของกราฟของวิธี PBRC 2-D PFFT จะลดลงต่ำกว่าวิธี RC 2-D PFFT ซึ่งมีสาเหตุเช่นเดียวกับการวัดเวลาทั้งหมดที่ใช้ประมวลผลที่ได้กล่าวไว้ข้างต้น

ตารางที่ 4.7 แสดงเวลาที่ใช้ในการคำนวณ (Computation Time) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ

ขนาดชุดข้อมูล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
512 × 512	0.01151	0.13070	0.04023
1024 × 1024	0.05418	0.55995	0.15133
2048 × 2048	0.99461	3.80808	1.63182
4096 × 4096	4.40803	21.68703	9.79970

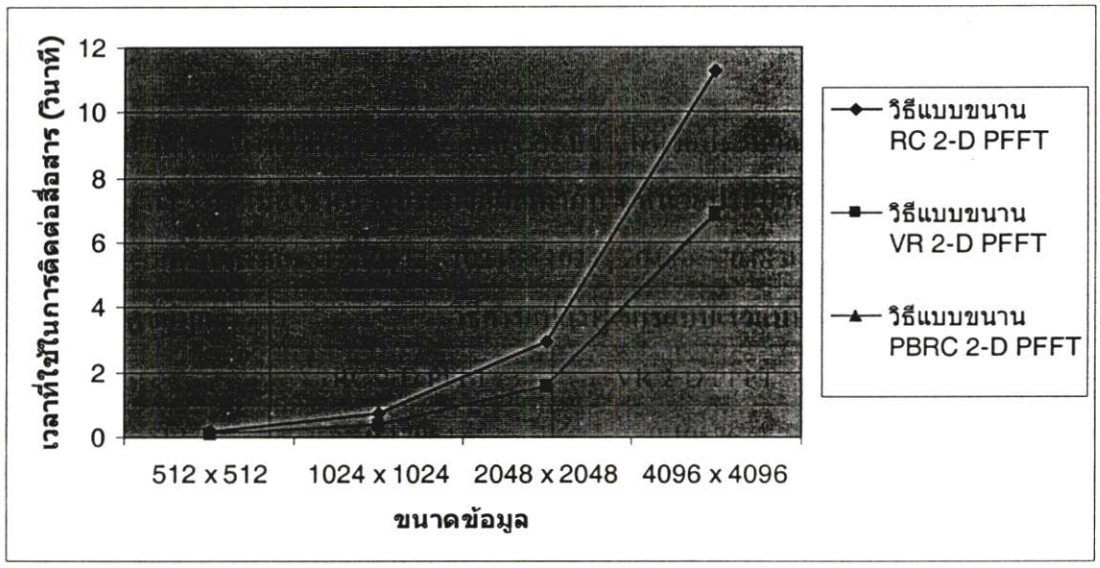


รูปที่ 4.8 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการคำนวณการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล

จากรูปที่ 4.8 แสดงการเปรียบเทียบเวลาที่ใช้ในการคำนวณแบบขนานการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานโดยไม่รวมเวลาที่ใช้ในการติดต่อสื่อสาร ซึ่งจะทำการเปรียบเทียบเวลาทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผล 4 หน่วยประมวลผล ทำการแปลงข้อมูลชนิดเลขจำนวนจริงละเอียด 2 เท่า (Double Precision) ซึ่งอยู่ในรูปเมตริกซ์ 2 มิติโดยมีตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ จะเห็นว่าวิธี RC 2-D PFFT จะใช้เวลาในการคำนวณน้อยที่สุด รองมาคือวิธี PBRC 2-D PFFT และวิธี VR 2-D PFFT จะใช้เวลาในการคำนวณมากที่สุด

ตารางที่ 4.8 แสดงเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล (Communication Time) ทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และขนาดข้อมูลในรูปเมตริกซ์ตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ

ขนาดชุดข้อมูล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
512 × 512	0.17964	0.10876	0.09086
1024 × 1024	0.74520	0.41707	0.40407
2048 × 2048	2.93021	1.57070	1.56793
4096 × 4096	11.27235	6.89867	6.87045



รูปที่ 4.9 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผลของการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล

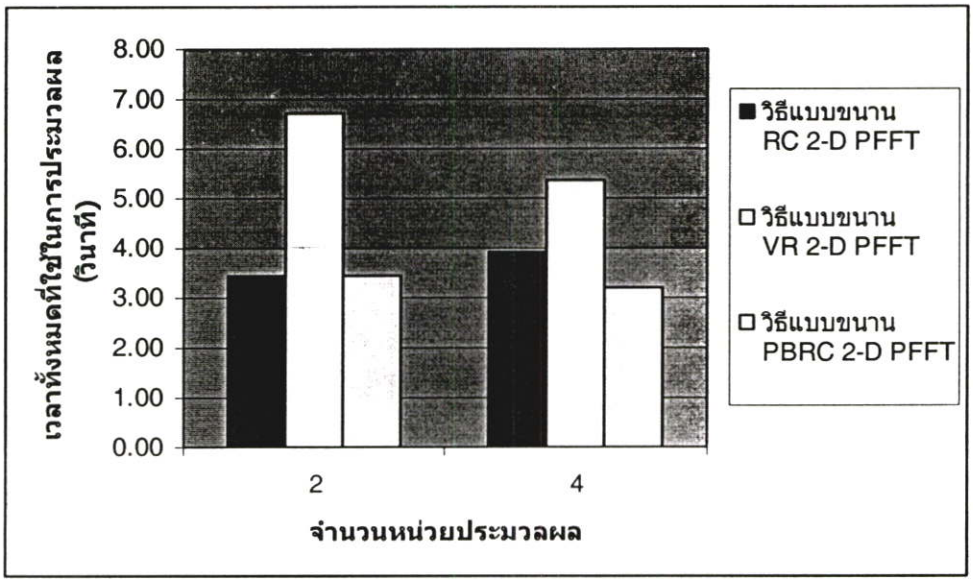
จากรูปที่ 4.9 แสดงการเปรียบเทียบเวลาที่ใช้ในการติดต่อสื่อสารเพื่อแลกเปลี่ยนข้อมูลระหว่างหน่วยประมวลผลเท่านั้น โดยทำการเปรียบเทียบเวลาทั้ง 3 วิธี เมื่อใช้หน่วยประมวลผล 4 หน่วยประมวลผล ทำการแปลงข้อมูลชนิดเลขจำนวนจริงละเอียด 2 เท่า (Double Precision) ซึ่งอยู่ในรูปเมตริกซ์ 2 มิติโดยมีตั้งแต่ 512 × 512 , 1024 × 1024 , 2048 × 2048 และ 4096 × 4096 ตามลำดับ จะเห็นว่าวิธี PBRC 2-D PFFT และวิธี VR 2-D PFFT จะใช้เวลาในการติดต่อสื่อสารใกล้เคียงกัน ซึ่งจะใช้เวลาน้อยกว่าวิธี RC 2-D PFFT

4.3.3 ผลการทดลองเปรียบเทียบเมื่อจำนวนหน่วยประมวลผลที่ใช้ทดลองต่างกัน

การทดลองเปรียบเทียบการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธีที่ได้กล่าวมาในข้างต้น เมื่อใช้จำนวนหน่วยประมวลผลต่างกัน โดยทำการทดลองที่จำนวนหน่วยประมวลผลเท่ากับ 2 และ 4 หน่วยประมวลผล กับขนาดของชุดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 ซึ่งแสดงผลการทดลองดังนี้ ตารางที่ 4.9 และรูปที่ 4.10 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ที่เป็นผลรวมของเวลาที่ใช้ในการติดต่อสื่อสารและเวลาที่ใช้ในการประมวลผล ($T_p = T_s / P + T_c$) นอกจากนี้เราสามารถนำเวลาที่ใช้ในการประมวลผลทั้งหมดมาทำการคำนวณหาอัตราการเพิ่มขึ้นของความเร็ว ดังแสดงในตารางที่ 4.10 และรูปที่ 4.11 จากนั้นนำค่าของอัตราการเพิ่มขึ้นของเร็วมาคำนวณหาประสิทธิภาพ จากนั้นนำค่าของอัตราการเพิ่มขึ้นของความเร็วมาคำนวณหาประสิทธิภาพ ตามสมการที่ 2.13 ($E_p = S_s / P$) ดังแสดงในตารางที่ 4.11 และรูปที่ 4.12

ตารางที่ 4.9 แสดงเวลาที่ใช้ในการประมวลผลทั้งหมด (Response Time) ทั้ง 3 วิธี เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 และใช้หน่วยประมวลผลเท่ากับ 2 และ4 หน่วยประมวลผล

จำนวนหน่วยประมวลผล	**วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ**		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
2	3.45473	6.70704	3.45576
4	3.92483	5.37878	3.19975

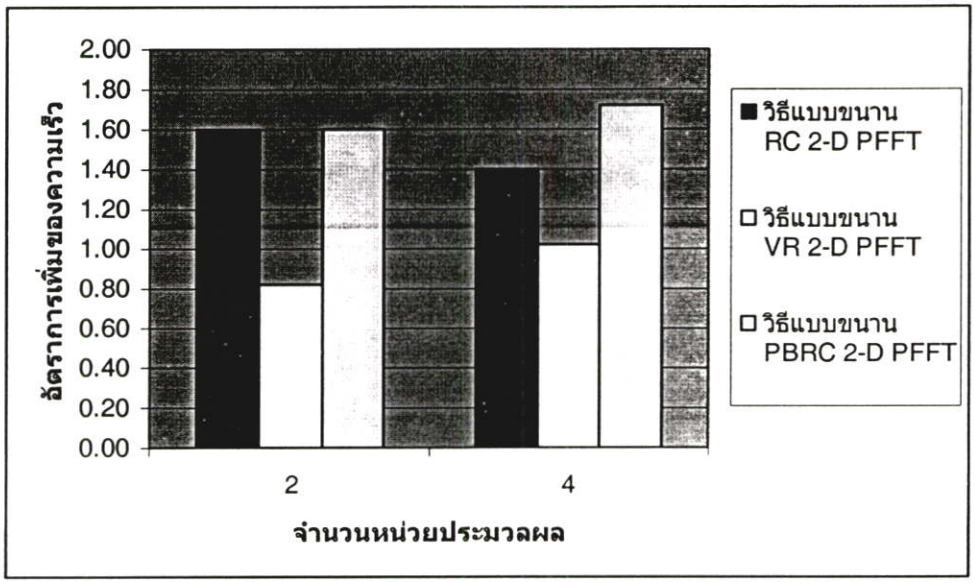


รูปที่ 4.10 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการประมวลผลทั้งหมดที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048

จากรูปที่ 4.10 แสดงเวลาทั้งหมดที่ใช้ในการประมวลผล ซึ่งจะพบว่า เมื่อเพิ่มจำนวนหน่วยประมวลผล เวลาทั้งหมดที่ใช้ในการประมวลผลจะลดลง โดยวิธี RC 2-D PFFT ใช้เวลาในการประมวลผลน้อยที่สุด เท่ากับ 3.45576 วินาที เมื่อจำนวนหน่วยประมวลผลเท่ากับ 2 หน่วยประมวลผล ส่วนโดยวิธี PBRC 2-D PFFT ใช้เวลาในการประมวลผลน้อยที่สุด เท่ากับ 3.19975 วินาที เมื่อจำนวนหน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และวิธี VR 2-D PFFT ใช้เวลาในการประมวลผลมากกว่าทุกๆวิธี

ตารางที่ 4.10 แสดงอัตราการเพิ่มของความเร็ว (Speedup) ทั้ง 3 วิธี เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 และใช้หน่วยประมวลผลเท่ากับ 2 และ4 หน่วยประมวลผล

จำนวนหน่วยประมวลผล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
2	1.59572	0.82194	1.59524
4	1.40459	1.02491	1.72288

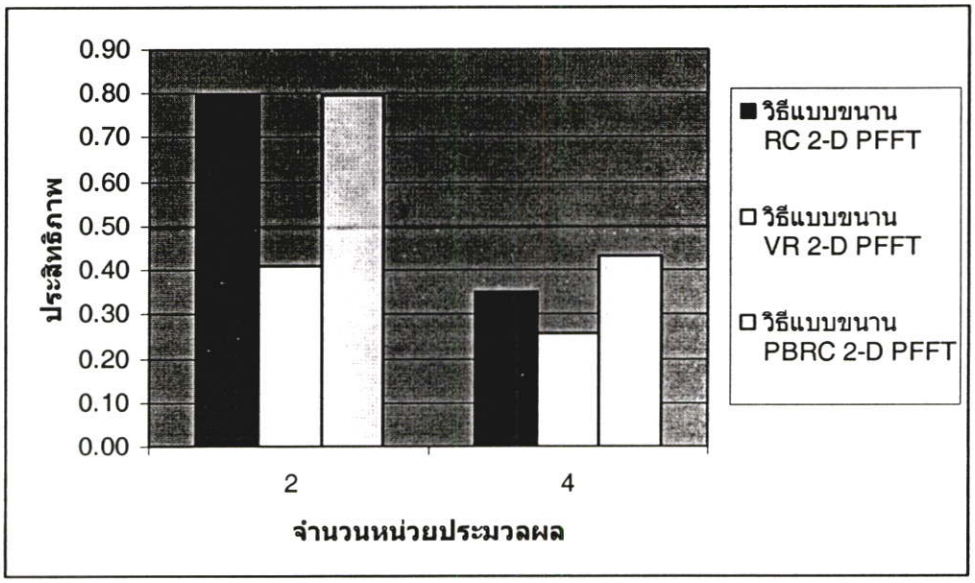


รูปที่ 4.11 แสดงกราฟเปรียบเทียบอัตราการเพิ่มของความเร็วที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048

จากรูปที่ 4.11 แสดงอัตราการเพิ่มขึ้นของความเร็ว จะพบได้ว่า เมื่อเพิ่มจำนวนหน่วยประมวลผล อัตราการเพิ่มขึ้นของความเร็วจะเพิ่มขึ้น โดยวิธี RC 2-D PFFT มีอัตราการเพิ่มขึ้นของความเร็วมากที่สุด เท่ากับ 1.59572 เมื่อจำนวนหน่วยประมวลผลเท่ากับ 2 หน่วยประมวลผล ส่วนโดยวิธี PBRC 2-D PFFT มีอัตราการเพิ่มขึ้นของความเร็วมากที่สุด เท่ากับ 1.40459 เมื่อจำนวนหน่วยประมวลผลเท่ากับ 4 หน่วยประมวลผล และวิธี VR 2-D PFFT มีอัตราการเพิ่มขึ้นของความเร็วน้อยกว่าทุกวิธี

ตารางที่ 4.11 แสดงประสิทธิภาพ (Efficiency) ทั้ง 3 วิธี เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048 และใช้หน่วยประมวลผลเท่ากับ 2 และ4 หน่วยประมวลผล

จำนวนหน่วยประมวลผล	วิธีการแปลงฟูเรียร์แบบเร็วแบบขนาน 2 มิติ		
	RC 2-D PFFT	VR 2-D PFFT	PBRC 2-D PFFT
2	0.79786	0.41097	0.79762
4	0.35115	0.25623	0.43072



รูปที่ 4.12 แสดงกราฟเปรียบเทียบประสิทธิภาพที่ใช้ในการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธี เมื่อใช้ขนาดข้อมูลในรูปเมตริกซ์ขนาด 2048 × 2048

จากรูปที่ 4.12 แสดงผลการเปรียบเทียบประสิทธิภาพ จะพบได้ว่า เมื่อเพิ่มจำนวนหน่วยประมวลผล ประสิทธิภาพจะลดลง โดยที่วิธี RC 2-D PFFT มีประสิทธิภาพมากที่สุดเมื่อใช้หน่อยประมวลผลเท่ากับ 2 หน่วยประมวลผล โดยมีค่าเป็น 0.79786 ส่วนวิธี PBRC 2-D PFFT มีประสิทธิภาพมากที่สุดเมื่อใช้หน่อยประมวลผลเท่ากับ 4 หน่วยประมวลผล โดยมีค่าเป็น 0.43072 ส่วนวิธี VR 2-D PFFT มีค่าประสิทธิภาพน้อยกว่าทุกๆ วิธี

<u>หมายเหตุ</u>

จากการทดลองการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานบนระบบพีซีคลัสเตอร์ พบข้อจำกัดในการประมวลผลแบบขนานบนระบบพิซีคลัสเตอร์ ซึ่งมี ดังนี้

1. **ข้อจำกัดเกี่ยวกับหน่วยความจำของระบบ**

   ในการประมวลผลแบบขนานซึ่งมักจะใช้กับข้อมูลที่มีขนาดใหญ่จำเป็นต้องมีการจัดสรรหน่วยความจำที่เหมาะสมเพื่อการทำงานที่มีประสิทธิภาพ ซึ่งรวมถึงขนาดของหน่วยความจำของระบบที่ต้องมีขนาดเพียงพอกับขั้นตอนวิธีและขนาดข้อมูลที่นำมาใช้

2. **ข้อจำกัดเกี่ยวกับความเร็วเครือข่ายของระบบ**

   ในระบบพิซีคลัสเตอร์นั้น แต่ละหน่วยประมวลผลจะทำการติดต่อสื่อสารกันผ่านเครือข่ายโดยที่มีตัวสวิตช์เป็นตัวจัดการ ดังนั้นความเร็วในการติดต่อระหว่างแต่ละหน่วย

ประมวลผลจึงขึ้นอยู่กับความเร็วของการส่งข้อมูลของระบบเครือข่าย จึงจำเป็นต้องใช้ระบบเครือข่ายที่มีความเร็วมากเพื่อเป็นการลดเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผลของระบบพีซีคลัสเตอร์

3. ข้อจำกัดเกี่ยวกับการจัดการหน่วยประมวลผลของระบบ

ในการประมวลผลแบบขนานนั้นเป้าหมายที่สำคัญ คือ การแบ่งงานที่มีขนาดใหญ่ให้แต่ละหน่วยประมวลผลช่วยทำการประมวลผลพร้อมกัน ซึ่งในบางครั้งจำเป็นต้องใช้ข้อมูลจากหน่วยประมวลผลอื่นในการประมวลผลบนหน่วยประมวลผลของตัวเอง ซึ่งถ้าหากหน่วยประมวลผลใดทำงานช้าก็จะส่งผลไปถึงเวลาที่ใช้ประมวลผลทั้งระบบ เนื่องจากระบบปัจจุบันที่ใช้ทำการทดลองเป็นระบบเปิด ซึ่งอาจมีบางหน่วยประมวลผลทำงานอื่นอยู่ด้วย ถึงแม้ว่าจะทำการจัดสรรงานอย่างสมดุลแล้วก็ตาม ดังนั้น ระบบพีซีคลัสเตอร์ที่ใช้ในการทดลองที่ดีควรเป็นระบบแบบปิด ซึ่งก็คือในระหว่างการทดลองควรให้หน่วยประมวลผลทำการประมวลผลงานที่ใช้ในการทดลองเพียงอย่างเดียวเพื่อให้เวลาที่ได้มีประสิทธิภาพมากที่สุด

บทที่ 5
สรุปผลการทดลองและแนวทางการพัฒนางานวิจัย

5.1 สรุปผลและวิเคราะห์ผลการทดลอง

การแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานที่ได้นำเสนอไว้ในบทที่ 3 ซึ่งมีทั้งหมด 3 วิธี คือ วิธีที่ 1) อาร์ซีทูดีพีเอฟเอฟที (Row Column Two-Dimensional Parallel Fast Fourier Transform : RC 2-D PFFT) วิธีที่ 2) วีอาร์ทูดีพีเอฟเอฟที (Vector Radix Two-Dimensional Parallel Fast Fourier Transform : VR 2-D PFFT) วิธีที่ 3) พีบีอาร์ซีทูดีพีเอฟเอฟที (Partition Block Row Column Two-Dimensional Parallel Fast Fourier Transform : PBRC 2-D PFFT) โดยแต่ละวิธีดังกล่าวมีวัตถุประสงค์และขั้นตอนการทำงานที่แตกต่างกันดังนี้ คือ วิธีการแปงฟูเรียร์แบบเร็ว 2 มิติแบบขนานวิธีที่ 1 อาร์ซีทูดีพีเอฟเอฟที (RC 2-D PFFT) เป็นวิธีแบบดั้งเดิม และพัฒนาง่ายที่สุดเมื่อเทียบกับวิธีอื่นๆ เนื่องจากในงานวิจัยส่วนใหญ่จะใช้ขั้นตอนวิธีนี้ในการพัฒนาวิธีการแปลงฟูเรียร์ วิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานวิธีที่ 2 วีอาร์ทูดีพีเอฟเอฟที (VR 2-D PFFT) เป็นขั้นตอนวิธีที่พัฒนาขึ้นเพื่อใช้ในการคำนวณวิธีการแปลงฟูเรียร์อีกวิธีหนึ่งซึ่งจะมีประสิทธิภาพในการติดต่อระหว่างหน่วยประมวลผล แต่จะมีรูปแบบการทำงานที่ซับซ้อนกว่าวิธีแรก และวิธีสุดท้าย วิธีการแปงฟูเรียร์แบบเร็ว 2 มิติแบบขนานวิธีที่ 3 พีบีอาร์ซีทูดีพีเอฟเอฟที (PBRC 2-D PFFT) เป็นวิธีที่นำเอาข้อดีของขั้นตอนของทั้ง 2 วิธีแรกมาร่วมกันเพื่อเพิ่มประสิทธิภาพในการทำงานโดยจะเน้นไปที่การเพิ่มประสิทธิภาพในการติดต่อสื่อสารระหว่างหน่วยประมวลผลแต่ยังคงใช้ขั้นตอนวิธีในการคำนวณที่ง่ายไม่ซับซ้อน

จากการทดลองที่ผ่านมาในบทที่ 4 ซึ่งเป็นการทดลองบนระบบพีซีคลัสเตอร์ของสำนักบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง จะเห็นว่าการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานในแต่ละวิธีจะมีประสิทธิภาพในด้านต่างๆกัน วึ่งเมื่อใช้หน่วยประมวลผล 4 หน่วยประมวลผล กับข้อมูลในรูปเมตริกซ์ขนาดตั้งแต่ 512×512 , 1024×1024 , 2048×2048 และ 4096×4096 ตามลำดับ แล้วนำผลการทดลองมาหาค่าเฉลี่ย ซึ่งสามารถสรุปได้ผลการทดลองดังนี้

ขั้นตอนวิธีการแปลงฟูเรียร์	ประสิทธิภาพเฉลี่ย
RC 2-D PFFT	0.273212
VR 2-D PFFT	0.196775
PBRC 2-D PFFT	0.344240

จากผลการทดลองพบว่า ขั้นตอนวิธีพีบีอาร์ซีทูดีพีเอฟเอฟที (PBRC 2-D PFFT) มีสมรรถนะโดยเฉลี่ยดีกว่าขั้นตอนวิธีอาร์ซีทูดีพีเอฟเอฟที (RC 2-D PFFT) และขั้นตอนวิธีวีอาร์ทูดีพีเอฟเอฟที (VR 2-D PFFT) อย่างน้อย 7.1% และ 14.75% ตามลำดับ ซึ่งสามารถแบ่งสมรรถนะของแต่ละขั้นตอนวิธีได้เป็น 2 ด้านใหญ่ๆ ดังนี้

1) สมรรถนะในด้านเวลาที่ใช้ในการคำนวณวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติ

2) สมรรถนะในด้านเวลาที่ใช้ในการติดต่อสื่อสารระหว่างหน่วยประมวลผล

ซึ่งผลสรุปจากการทดลองสามารถเขียนเป็นตาราง ดังนี้

ตารางที่ 5.1 แสดงการเปรียบเทียบข้อดี-ข้อเสียของวิธีการแปลงฟูเรียร์แบบเร็ว 2 มิติแบบขนานทั้ง 3 วิธีบนระบบพีซีคลัสเตอร์

วิธีการแปลงฟูเรียร์	ข้อดี	ข้อเสีย
วิธี RC 2-D PFFT	1. เหมาะกับการคำนวณบนเครื่องที่มีหน่วยประมวลผลเดียว 2. ง่ายต่อการพัฒนาเนื่องจากมีขั้นตอนวิธีในการคำนวณที่ง่าย ไม่ซับซ้อนและมีสมรรถนะในด้านเวลาที่ใช้ในการคำนวณ	1. เมื่อนำมาพัฒนาบนระบบที่มีการประมวลผลแบบขนานจะมีขั้นตอนการติดต่อสื่อสารระหว่างหน่วยประมวลผลที่ซับซ้อนและใช้เวลาในการติดต่อสื่อสารระหว่างหน่วยประมวลผลมาก
วิธี VR 2-D PFFT	1. เมื่อนำมาพัฒนาบนระบบที่มีการประมวลผลแบบขนานจะมีขั้นตอนการติดต่อสื่อสารระหว่างหน่วยประมวลผลที่ง่าย ไม่ซับซ้อนและใช้เวลาในการติดต่อสื่อสารระหว่างหน่วยประมวลผลน้อย	1. ไม่เหมาะกับการคำนวณบนเครื่องที่มีหน่วยประมวลผลเดียว เนื่องจากมีขั้นตอนการทำงานที่ซับซ้อนและใช้เวลาในการคำนวณมาก 2. มีข้อจำกัดของจำนวนหน่วยประมวลผลที่นำมาใช้เมื่อพัฒนาบนระบบแบบขนาน คือ ต้องไม่เกิน 4 หน่วยประมวลผล

วิธี PBRC 2-D PFFT	1. เมื่อนำมาพัฒนาบนระบบที่มีการประมวลผลแบบขนานจะมีขั้นตอนการติดต่อสื่อสารระหว่างหน่วยประมวลผลที่ง่ายไม่ซับซ้อนและใช้เวลาในการติดต่อสื่อสารระหว่างหน่วยประมวลผลน้อย 2. มีขั้นตอนการคำนวณที่ง่ายต่อการพัฒนาและมีสมรรถนะในด้านเวลาที่ใช้ในการคำนวณตามรูปแบบของวิธี RC 2-D PFFT	1. มีข้อจำกัดของจำนวนหน่วยประมวลผลที่นำมาใช้เมื่อพัฒนาบนระบบแบบขนาน คือ ต้องไม่เกิน 4 หน่วยประมวลผล โดยควรอยู่ในรูปฐาน 2 (เป็น 2 หรือ 4)

จากตารางที่ 5.1 พบว่าวิธี PBRC 2-D PFFT วิธีที่เหมาะสมในการประมวลผลแบบขนาน โดยเฉพาะกับระบบพีซีคลัสเตอร์ (Cluster of PCs) เนื่องจากมีสมรรถนะ (Performance) ทั้งในด้านการคำนวณและการติดต่อสื่อสารระหว่างหน่วยประมวลผล

5.2 แนวทางการพัฒนางานวิจัย

- นำโปรแกรมที่ได้พัฒนาขึ้นไปใช้กับระบบคลัสเตอร์แบบเนื้อผสม (Heterogeneous Cluster) จำเป็นต้องให้ความสำคัญในเรื่องของการสร้างสมดุลของงาน (Load Balancing) ในระบบ เนื่องจากคอมพิวเตอร์ที่เชื่อมต่ออยู่ในระบบนี้ สามารถมีองค์ประกอบภายในที่แตกต่างกันได้ ทำให้คอมพิวเตอร์แต่ละเครื่องมีความสามารถและประสิทธิภาพในการประมวลผลงานที่แตกต่างกัน

- นำไปพัฒนาโปรแกรมให้สามารถใช้งานได้ง่ายขึ้นและใช้ได้หลายระบบปฏิบัติการ เช่น พัฒนาบนระบบปฏิบัติการไมโครซอฟท์วินโดวส์ เป็นต้น และสามารถนำโปรแกรมที่พัฒนาขึ้นไปประยุกต์ใช้กับงานทางด้านการประมวลผลสัญญาณดิจิทัล (Digital Signal Processing) หรือ งานทางด้านการประมวลผลรูปภาพดิจิทัล (Digital Image Processing) เป็นต้น

เอกสารอ้างอิง

[1] Barua , Sajib. , Thulasiram , Ruppa , K. and Thulasiraman , Parimala. **Improving Data Locality in Parallel Fast Fourier Transform Algorithm for Pricing Financial Derivatives**. Proceeding of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04). IEEE , 2004.

[2] Chu , Clare Y. **Comparison of Two-Dimensional FFT Methods on the Hypercube.** ACM. 1988.

[3] Cui-xiang , Zhong. and Guo-qiang , Han. **Some New Parallel Fast Fourier Transform Algorithm.** Proceeding of the Sixth International Conference on Parallel and Distributed Computing. Applications and Technologies (PDCAT'05). IEEE , 2005.

[4] Frigo , M. **A Fast Fourier Transform Compiler**. Proceeding of the 1999 ACM SIGPLIAN Conference on Programming Language Design and Implementation (PLDI). Atlanta. 1999.

[5] Frigo , M. and Johnson , S. G. **The Fast Fourier Transform in the West**. MIT-LCS-TR-728. Massachusetts Institute of Technology. 1997.

[6] Frigo , M. and Johnson , S. G. **FFTW : An Adaptive Software Architecture for the FFT**. ICASSP Confernence proceedings Vol. 3. 1998.

[7] Harris , David B. , McClellan , James H. , Chan , David S. K.. and Schuessler Han W. **Vector Radix Fast Fourier Transfrom.** IEEE Int. Conf. On Acoustics, Speech and Signal Procesing. May 1977.

[8] Honec , J. , Honec , P. , Petyovsky , P. , Valach , S. and Brambor , J. **Parallel 2-D FFT Implementation With DSPs**. 13th Int. Conference on Process Control. Slovakia. June 11-14 2001.

[9] Hwang , K. **Advance Computer Architecture : Parallelism Scalability Programmability**. New York : Mcgraw-Hill. 1993.

[10] Kumar , V. **Introduction to parallel computing : design and analysis of algorithms.** Redwood City, CA : Benjamin/Cummings. 1994.

[11] Kuniobo , Tanno. Toshihiro , Takeda. and Suumu , Horiguchi. **Parallel Radix 4 FFT Algorithms On An Eight-Neighbor Processor Array.** IEEE Region 10 Conference,Tencon 92. November 1992.

[12] Lim , Jae , S. **Two-Dimensional Signal and Image Processing.** London : Prentice-Hill International Edition. 1990.

[13] Marina , Francescomaria. , and Swartzlander , Earl E. **Parallel Implementation of Multidimensional Transforms without Interprocessor Communication.** IEEE Transactions On Computer, Vol. 48, No. 9. September 1999.

[14] Marios , S. Pattichis , Ruhai , Zhou and Balaji , Raman. **New Algorithm For Computing Directional Discrete Fourier Transforms.** IEEE. 2001.

[15] Miller , R. **Algorithms sequential and parallel : a unified approach.** Upper Saddle River, NJ : Prentice Hall. 2000.

[16] Pytosh , Teresa M. and Magnani , Alberto M. **A New Parallel 2-D FFT Architecture.** IEEE. 1990.

[17] Quinn , J. M. **Parallel Programming in C with MPI and OpenMP.** International Edition. Singapore : McGraw Hill. 2003.

[18] Quinn , J. M. **Parallel computing : theory and practice.** 2nd ed. New York : McGraw-Hill. 1994.

[19] Walker , David W. **Portable Programming within a Mesage-Passing Model : the FFT as an Example.** ACM. 1988.

[20] Wilkinson , Barry. and Allen , Michael. **Parallel Programming : Techniques and Applications Using Networked Workstations and Parallel Computers.** 2nd ed. Upper Saddle River, NJ : Prentice Hall. 2005.

[21] ชิดชนก เหลือสินทรัพย์. **Analyis & Design of Algorithms.** กรุงเทพมหานคร : บ.ซีเอ็ดยูเคชั่น จำกัด (มหาชน). 2543.

[22] นพรัตน์ พันธุ์เสนา **"การสร้างภาพเชิงปริมาตรบนระบบคลัสเตอริ่งโดยวิธีการแปลงเฉือนและบิด."** วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2547.

[23] ภุชงค์ อุทโยภาส, "แนะนำเทคโนโลยีระบบพีซีคลัสเตอร์" **วิศวกรรมสาร มก.** ปีที่ 16, ฉบับที่47, สิงหาคม-พฤศจิกายน 2545. หน้า 1-8.

[24] สมชาย จิตะพันธ์กุล. **"การแปลงฟูเรียร์และการประยุกต์"** กรุงเทพมหานคร : บ.ซีเอ็ดยูเคชั่น จำกัด (มหาชน). 2543.

[25] อำไพ พรประเสริฐกุล. **Introduction to Computer Organization.** กรุงเทพมหานคร : บ.ซีเอ็ดยูเคชั่น จำกัด (มหาชน). 2543.

ภาคผนวก ก.

วิธีการติดตั้งและใช้งานไลบรารี่มาตรฐานในการคำนวณวิธีการแปลงฟูเรียร์ FFTW Library (ที่มา www.fftw.org) บนระบบที่มีการประมวลผลแบบขนาน ซึ่งใช้ร่วมกับมาตรฐานภาษา MPI และภาษาซี (C Language)โดยทำการพัฒนาบนระบบปฏิบัติการยูนิกซ์ (Unix Systems)

ขั้นตอนการติดตั้ง

1. ทำการดาวโหลดตั้งไลบรารี่ FFTW จากเวป์ไซด์ www.fftw.org
2. พิมพ์คำสั่งเพื่อทำการติดตั้งอย่างง่ายดังนี้ ดังนี้

```
. / configure
make
make  install
```

ขั้นตอนการใช้งาน

ในการใช้งาน MPI FFTW กับข้อมูลที่เป็นจำนวนเชิงซ้อน (Complex Number)เพื่อทำการคำนวณการแปลงฟูเรียร์นั้นมีรูปแบบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
# include < fftw_mpi.h >
int  main (int  arc,char ** argv)
{
        conts  int NX = ... , NY = ... ;
        fftwnd_mpi_plan  plan;
        fftw_complex  *data;

        MPI_Init(&argc , &argv);
        Plan =  fftw2d_mpi_create_plan (MPI_COMM_WORLD ,
                                        NX , NY
                                        FFTW_FORWARD , FFTW_ESTIMATE);
       ... allocate  and  initialize data ...
        fftwnd_mpi (p , data , NULL , FFTW_NORMAL_ORDER);
```

```
    ...
    fftwnd_mpi_destroy_plan (plan);
    MPI_Finalize();
}
```

โดยในขั้นตอนแรกต้องทำการสร้าง plan สำหรับใช้งานขึ้นมาก่อน โดยมีรูปแบบดังนี้

```
fftwnd mpi_plan fftw2d_mpi_create_plan (MPI_COMM comm ,
                                        int nx , int ny ,
                                        fftw_direction dir , int flag);
```

หลังจากนั้นให้ทำการจองเนื้อที่และสร้างข้อมูลเพื่อทำการแปลงฟูเรียร์ ซึ่งจะทำการแปลงข้อมูลดังรูปแบบนี้

```
void  fftwnd_mpi (fftwnd_mpi_plan  p ,
                  int n_fields ,
                  fftw_complex  * local_data , fftw_complex  * work ,
                  fftwnd_mpi_output_order  output_order);
```

เมื่อทำการแปลงฟูเรียร์เสร็จแล้วให้ทำการทำลาย plane ที่สร้างขึ้นโดยใช้รูปแบบดังนี้

```
fftwnd_mpi_destroy_plan(plan);
```

ในส่วนขั้นตอนคอมไพล์โปรแกรมให้ทำการพิมพ์ – lfftw_mpi -lfftw –lm เพื่อทำการลิงค์ไปยังไลบรารี่ FFTW

ประวัติผู้เขียน

ชื่อ – สกุล	นายอภินันทน์ เสริมศรี
วัน เดือน ปีเกิด	10 พฤษภาคม 2524
ที่อยู่	2 หมู่ 2 ตำบลแหลมงอบ
	อำเภอแหลมงอบ จังหวัดตราด 23120
ประวัติการศึกษา	
	2547 จบการศึกษาประกาศนียบัตรบัณฑิตวิชาชีพครู
	สาขาการศึกษา มหาวิทยาลัยบูรพา
	2546 จบการศึกษาปริญญาวิทยาศาสตรบัณฑิต
	สาขาคณิตศาสตร์ มหาวิทยาลัยบูรพา
ทุนการศึกษา	
พ.ศ. 2548	ทุนมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสาร
พ.ศ. 2547 - 2549	โครงการพัฒนาครูแกนนำในสาขาคณิตศาสตร์ วิทยาศาสตร์และ
	เทคโนโลยีและสิ่งแวดล้อม (Project Guiding Light)
พ.ศ. 2542 - 2547	โครงการส่งเสริมการผลิตครูที่มีความสามารถพิเศษทางวิทยาศาสตร์และ
	คณิตศาสตร์ (สควค.)