บทที่ 5
โปรแกรมแบบขนาน

5.1 บทนำ

การประมวลผลแบบขนานนอกจากต้องมีระบบคอมพิวเตอร์แบบขนานแล้วยังจำเป็นต้องมีโปรแกรมแบบขนานเพื่อใช้สำหรับประมวลผลบนระบบคอมพิวเตอร์แบบขนาน ลักษณะของโปรแกรมแบบขนานจะขึ้นอยู่กับลักษณะของระบบคอมพิวเตอร์แบบขนานโดย โปรแกรมแบบขนานสำหรับระบบคอมพิวเตอร์ขนานแบบหนึ่งๆอาจไม่สามารถทำงานได้บนระบบคอมพิวเตอร์ขนานแบบอื่น

การสร้างโปรแกรมแบบขนานจะมีความยากง่ายแตกต่างกันไปขึ้นอยู่กับการประยุกต์ใช้งาน เครื่องมือที่ใช้ในการสร้างโปรแกรมแบบขนานและชนิดของระบบคอมพิวเตอร์แบบขนาน นอกจากนั้นประสิทธิภาพของโปรแกรมแบบขนานของงานหนึ่งๆอาจไม่เท่ากันเมื่อสร้างโปรแกรมแบบขนานนี้ด้วยวิธีที่ต่างกัน

การสร้างโปรแกรมแบบขนานมีสองวิธี

1. การสร้างโปรแกรมแบบขนานโดยนัย (Implicit Parallelism) วิธีนี้สามารถทำได้ง่าย โดยใช้ภาษาแบบขนานหรือตัวแปลภาษา (Compiler) แบบขนาน แต่วิธีนี้ผู้สร้างโปรแกรม (Programmer) ไม่สามารถควบคุมการจัดการ การแบ่งส่วนคำนวณ และการแบ่งข้อมูลได้ ซึ่งโดยทั่วไปแล้วจะมีประสิทธิภาพต่ำกว่าการสร้างโปรแกรมแบบขนานโดยตรง (Explicit Parallelism)

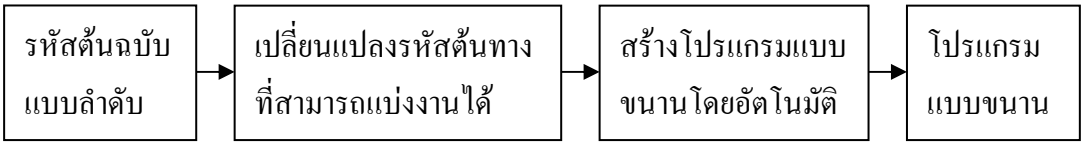
2. การสร้างโปรแกรมแบบขนานโดยตรง วิธีนี้ผู้สร้างโปรแกรมต้องทำการจัดแบ่งการทำงานเองทั้งหมด ซึ่งรูปแบบ และวิธีการแบ่งงานในแต่ละการประยุกต์ใช้จะไม่เหมือนกัน ซึ่งจะทำให้มีความยากง่ายแตกต่างกันด้วย แต่การใช้วิธีการแบ่งงานที่เหมาะสมกับการประยุกต์ใช้จะทำให้ได้ประสิทธิภาพในการประมวลผลสูงกว่าการสร้างโปรแกรมแบบขนานโดยนัย

5.2 รูปแบบของการพัฒนาโปรแกรมแบบขนาน

ก่อนที่จะสร้างโปรแกรมแบบขนานส่วนมากผู้สร้างโปรแกรมมักจะสร้างโปรแกรมในแบบลำดับ (Sequential Program) ซึ่งเป็นโปรแกรมที่ใช้ประมวลผลในเครื่องคอมพิวเตอร์แบบธรรมดาให้สามารถทำงานได้อย่างถูกต้องก่อน จากนั้นจะพัฒนาโปรแกรมแบบลำดับนี้ไปเป็นโปรแกรมแบบนานได้จากวิธีเหล่านี้

5.2.1 การสร้างโปรแกรมแบบขนานโดยอัตโนมัติ (Automatic Parallelism)

การสร้างโปรแกรมแบบขนานด้วยวิธีนี้เป็นวิธีที่ง่ายที่สุด แต่ก็จะมีประสิทธิภาพต่ำที่สุด โดยหน้าที่ของการสร้างโปรแกรมแบบขนานจะเป็นหน้าที่ของตัวแปรภาษาซึ่งตัวแปรภาษาจะตรวจสอบรหัสต้นฉบับ (Source Code) ซึ่งอาจประกอบด้วยส่วนของรหัสที่มีการวนซ้ำ และการประมวลผลกับข้อมูลที่ซ้ำๆกัน โดยตัวแปรภาษานี้จะทำการเปลี่ยนรหัสต้นฉบับนี้ไปเป็นรหัสที่สนับสนุนการประมวลผลแบบขนานจากนั้นใช้ตัวแปรภาษาเปลี่ยนรหัสที่สนับสนุนการประมวลผลแบบขนานนี้ไปเป็นโปรแกรมแบบขนาน ข้อจำกัดของการสร้างโปรแกรมแบบขนานด้วยวิธีนี้ขึ้นอยู่กับเทคโนโลยีที่ตัวแปรภาษาใช้ในการเปลี่ยนรหัสต้นฉบับเป็นรหัสที่สนับสนุนการประมวลผลแบบขนาน ขั้นตอนการทำงานของการสร้างโปรแกรมแบบขนานโดยอัตโนมัติซึ่งแสดงดังรูปที่ 5.1



รูปที่ 5.1 ขั้นตอนการทำงานของการสร้างโปรแกรมแบบขนานโดยอัตโนมัติ

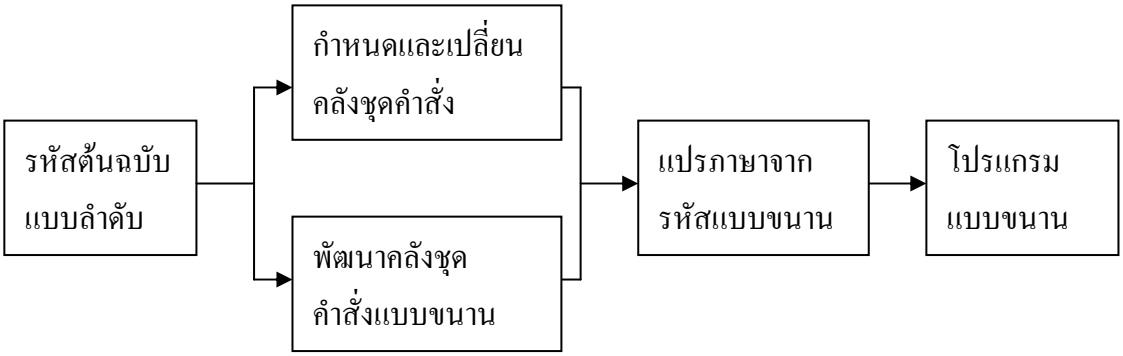
5.2.2 การสร้างโปรแกรมแบบขนานโดยใช้คลังของชุดคำสั่งแบบขนาน (Parallel Library)

วิธีนี้เป็นวิธีที่มีประสิทธิภาพดีกว่าวิธีแรก โดยใช้คลังของชุดคำสั่งแบบขนานที่พัฒนาไว้แล้วเช่น คลังชุดคำสั่งของการแปลงอนุกรมฟูเรียร์(Fourier) แบบขนาน และการคูณเมตริกแบบขนานเป็นต้น โดยการเปลี่ยนชุดคำสั่งจากรหัสต้นฉบับเป็นรหัสที่สนับสนุนการประมวลผลแบบขนานจะถูกทำโดยผู้สร้างโปรแกรมแบบขนาน จากนั้นใช้ตัวแปรภาษาทำการเปลี่ยนรหัสที่สนับสนุนการประมวลผลแบบขนานไปเป็นโปรแกรมแบบขนาน ขั้นตอนการสร้างโปรแกรมแบบขนานโดยใช้คลังของชุดคำสั่งแบบขนานแสดงดังรูปที่5.2 การใช้คลังของชุดคำสั่งแบบขนานสามารถสร้างโปรแกรมแบบขนานในสองรูปแบบ

- ใช้ควบคุมโครงสร้างของโปรแกรมแบบขนานในการกระจายงานไปตามส่วนต่างๆ ของหน่วยการประมวลผล (Processing Elements)
- ใช้ในการทดแทนชุดคำสั่งต้นฉบับที่สามารถเปลี่ยนการทำงานเป็นแบบขนานได้โดยใช้คลังชุดคำสั่งที่สนับสนุนการทำงานแบบขนานที่พัฒนาขึ้น

ในวิทยานิพนธ์ฉบับนี้ใช้วิธีการสร้างโปรแกรมแบบขนานโดยใช้คลังของชุดคำสั่งแบบขนาน ซึ่งใช้เพื่อควบคุมโครงสร้างของโปรแกรมแบบขนานในการกระจายงานไปตามโพรเซส (Processes)

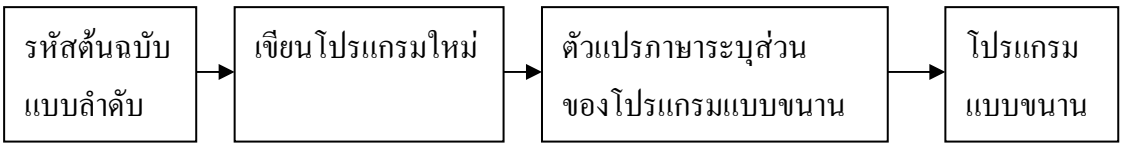
ต่างๆที่อยู่ในหน่วยการประมวลผล (Processors) แต่ไม่ได้ใช้คลังชุดคำสั่งในการเปลี่ยนรหัสต้นฉบับเป็นรหัสแบบขนาน



รูปที่ 5.2 ขั้นตอนการสร้างโปรแกรมแบบขนานโดยใช้คลังของชุดคำสั่งแบบขนาน

5.2.3 การสร้างโปรแกรมแบบขนานด้วยตัวเอง (Major Recoding)

การสร้างโปรแกรมแบบขนานด้วยวิธีนี้เป็นแบบที่ยืดหยุ่นมากที่สุด ผู้สร้างโปรแกรมสามารถเลือกตัวแปรภาษาใดก็ได้ที่ต้องการ รวมถึงสามารถเลือกรูปแบบและโปรแกรมที่ใช้ในการสื่อสาร แต่วิธีนี้นับเป็นวิธีที่ยากและเสียเวลามากที่สุด ขั้นตอนการสร้างโปรแกรมแบบขนานด้วยตัวเองแสดงดังรูปที่ 5.3



รูปที่ 5.3 ขั้นตอนการสร้างโปรแกรมแบบขนานด้วยตัวเอง

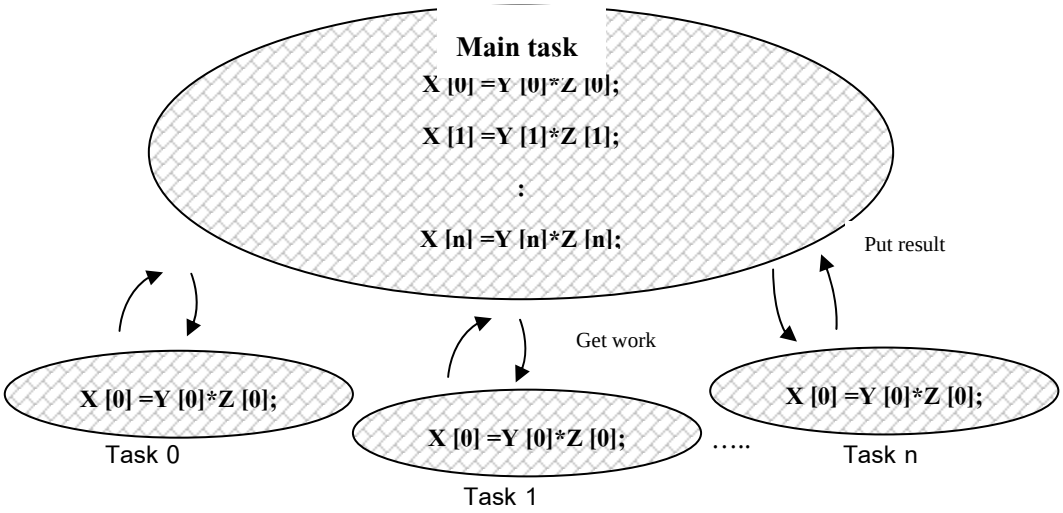
5.3 วิธีการออกแบบอัลกอริทึมของโปรแกรมแบบขนาน (Methodical Design of Parallel Algorithms)

การออกแบบอัลกอริทึมของโปรแกรมแบบขนานไม่มีข้อกำหนดตายตัว Ian Foster ได้แนะนำวิธีการออกแบบอัลกอริทึมแบบขนานของโปรแกรมไว้สี่ขั้นตอน คือ

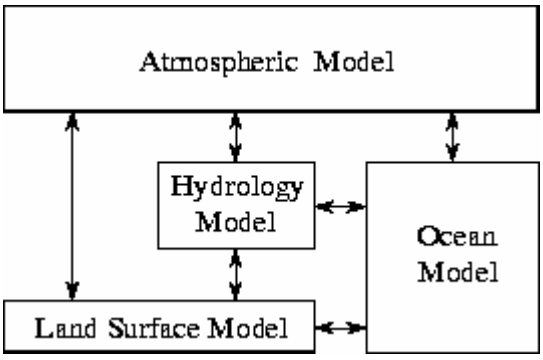
- ขั้นตอนการแบ่งงาน (Partitioning)
- ขั้นตอนออกแบบการสื่อสาร (Communication)
- ขั้นตอนการรวมกลุ่มงาน (Agglomeration)
- ขั้นตอนกำหนดงาน (Mapping)

5.3.1 ขั้นตอนการแบ่งงาน

การแบ่งงานเป็นหัวใจของการประมวลผลแบบขนาน ซึ่งเป็นการแยกการคำนวณออกเป็นส่วยย่อย การแบ่งงานสามารถทำได้สองแบบคือแบ่งข้อมูลของปัญหาออกเป็นส่วนย่อย (Domain Decomposition) แบบที่สองคือแบ่งปัญหาตามหน้าที่ของการคำนวณออกเป็นส่วนย่อย (Functional Decomposition) ตัวอย่างของการแบ่งข้อมูลของปัญหาแสดงดังรูปที่ 5.4 ซึ่งเป็นการแบ่งข้อมูลในการคำนวณออกเป็นส่วนๆ ส่วนการแบ่งงานตามหน้าที่ของปัญหาแสดงดังรูปที่ 5.5 ซึ่งเป็นตัวอย่างของการสร้างแบบจำลองของสภาพภูมิอากาศ ซึ่งประกอบด้วยงานย่อยๆคือการสร้างแบบจำลองของทะเล พื้นดิน และก๊าซ โดยทั่วไปแล้วการแบ่งข้อมูลจะมีความยืดหยุ่นและสามารถทำได้ง่ายกว่าการแบ่งงานตามหน้าที่



รูปที่ 5.4 การแบ่งขอบเขตข้อมูลของปัญหา



รูปที่ 5.5 การแบ่งขอบเขตหน้าที่ของปัญหา

หรือสามารถแบ่งงานโดยการกำหนดลักษณะการติดต่อกันโดยจำแนกตามจำนวนของชุดคำสั่งและข้อมูลได้ดังนี้

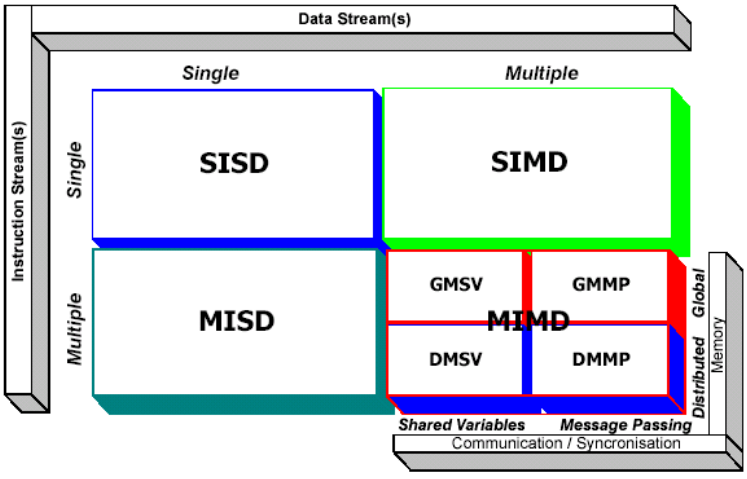
- SISD : ชุดคำสั่งเดียว ข้อมูลเดี่ยว (Single Instruction Stream / Single Data Stream)
- SIMD: ชุดคำสั่งเดี่ยว หลายข้อมูล (Single Instruction Stream / Multiple Data Stream)
- MISD: หลายชุดคำสั่ง ข้อมูลเดี่ยว (Multiple Instruction Stream / Single Data Stream) ยังไม่มีระบบแบบนี้เกิดขึ้นจริง
- MIMD: หลายชุดคำสั่ง หลายข้อมูล (Multiple Instruction Stream / Multiple Data Stream)

ตัวอย่างของระบบชุดคำสั่งเดี่ยวข้อมูลเดี่ยวคือเครื่องคอมพิวเตอร์ส่วนบุคล (Personal Computer) ซึ่งชุดคำสั่งจะถูกประมวลผลตามลำดับแต่อาจจะมีการซ้อนทับ (Overlap) ของชุดคำสั่งในขั้นตอนประมวลผลได้โดยใช้หลักการของท่อลำเลียง (Pipelining) ระบบคอมพิวเตอร์แบบขนานจะจัดอยู่ในพวกระบบหลายชุดคำสั่งหลายข้อมูลซึ่งอาจเป็นระบบหลายหน่วยประมวลผล (Multiprocessor) และระบบหลายเครื่องคอมพิวเตอร์ (Multiple Computer)

ระบบคอมพิวเตอร์แบบหลายชุดคำสั่งหลายข้อมูลนี้สามารถแบ่งย่อยได้สี่แบบคือ

- GMSV: หน่วยความจำรวม ตัวแปรร่วม(Global Memory / Shared Variables)
- GMMP: หน่วยความจำรวม ส่งข้อความ(Global Memory / Message Passing) ยังไม่มีระบบแบบนี้เกิดขึ้นจริง
- DMSV: หน่วยความจำแบบกระจาย ตัวแปรร่วม(Distributed Memory / Share Variables)
- DMMP: หน่วยความจำแบบกระจาย ส่งข้อความ(Distributed memory / Message Passing)

แผนผังการจำแนกประเภทของระบบคอมพิวเตอร์แสดงดังรูปที่ 5.6



รูปที่ 5.6 แผนผังการจำแนกประเภทของระบบคอมพิวเตอร์

ระบบคอมพิวเตอร์แบบขนานที่ใช้ในวิทยานิพนธ์ฉบับนี้เป็นระบบแบบ DMMP ซึ่งเป็นระบบที่ใช้เครื่องคอมพิวเตอร์มาทำการเชื่อมต่อกันผ่านระบบเครือข่ายความเร็วสูงและทำงานร่วมกันเสมือนเป็นระบบเดียว โดยระบบชนิดนี้เรียกว่าระบบคลัสเตอร์

5.3.2 ขั้นตอนออกแบบการสื่อสาร

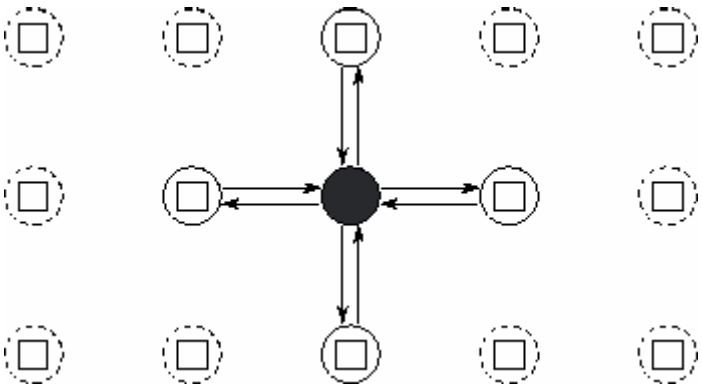
ขั้นตอนการสื่อสารจะถูกทำโดยพิจารณาการสื่อสารระหว่างงานที่แบ่งออกไป ในงานบางอย่าง งานย่อยที่ถูกแบ่งไม่สามารถประมวลผลได้ในเวลาเดียวกันเนื่องจากต้องอาศัยข้อมูลจากงานย่อยอื่นเพื่อใช้ในการประมวลผลดังนั้นจึงต้องมีการสื่อสารระหว่างงานย่อย การเลือกวิธีการสื่อสารที่เหมาะสมจะทำให้การปะมวลผลมีประสิทธิภาพมากยิ่งขึ้น การออกแบบขั้นตอนการสื่อสารจะแบ่งเป็นสองขั้นตอน ขั้นตอนแรกออกแบบโครงสร้างของการเชื่อมโยงว่างานใดต้องการข้อมูลและงานใดทำหน้าที่ให้ข้อมูล ขั้นตอนที่สองคือกำหนดโครงสร้างของข้อความ (Message) ที่ใช้ในการส่งและรับของแต่ละการเชื่อมโยง ลักษณะของการสื่อสารอาจแบ่งได้เป็นสี่แบบดังนี้

5.3.2.1 การสื่อสารระยะใกล้ และการสื่อสารแบบคลอบคลุม (Local and Global Communication)

การสื่อสารระยะใกล้คือการสื่อสารที่เกิดขึ้นระหว่างงานจำนวนน้อย เช่นการวิเคราะห์เชิงเลข (Numerical Analysis) ในวิธีไฟไนต์ดิฟเฟอเรนต์ของ Jacobi ซึ่งการหาค่า $X_{i,j}^{(t+1)}$ ทำได้จากสมการ 5.1 โดยใช้ข้อมูลห้าจุดในการคำนวณ

$$X_{i,j}^{(t+1)} = \frac{4X_{i,j}^{(t)} + X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}}{8} \qquad (5.1)$$

งานแต่ละงานทำหน้าที่หาค่าของ $X_{i,j}^{(t+1)}$ ที่ตำแหน่งต่างๆกัน และในขณะเดียวกันก็ทำหน้าที่ส่งค่าของตัวเองให้กับงานอื่นด้วยดังรูปที่ 5.7

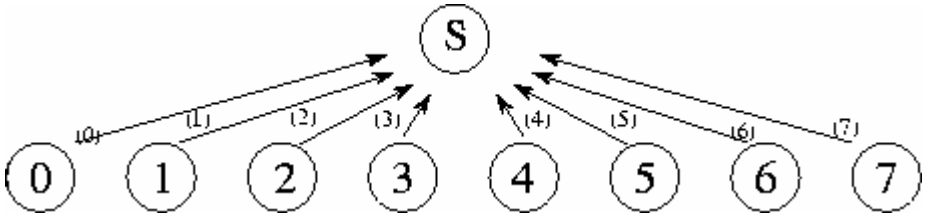


รูปที่ 5.7 ลักษณะการสื่อสารในการคำนวณไฟไนต์ดิฟเฟอเรนต์ในสองมิติ

การสื่อสารแบบคลอบคลุมจะเป็นการสื่อสารในลักษณะที่งานๆหนึ่งต้องติดต่อกับงานหลายๆงานเช่นการคำนวณหาค่าผลรวมดังสมการ 5.2

$$S = \sum_{i=0}^{N-1} X_i \qquad (5.2)$$

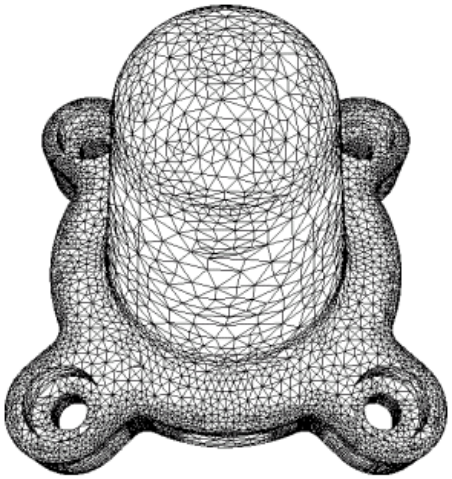
การคำนวณแบบขนานจะเกิดจากงานที่เป็นผู้จัดการ(Manager) จะรับค่า X_i จากงานที่เป็นคนงาน (Worker) เพื่อทำการหาค่า S โดยการสื่อสารจะมีลักษณะดังรูปที่ 5.8



รูปที่ 5.8 การสื่อสารเพื่อรวมค่าจากงานที่เป็นผู้จัดการ

5.3.2.2 การสื่อสารแบบไม่มีโครงสร้างและการสื่อสารแบบเปลี่ยนแปลงได้ (Unstructured and Dynamic Communication)

ในหัวข้อที่ผ่านมาเป็นการสื่อสารแบบคงที่และมีโครงสร้างซึ่งไม่มีการเปลี่ยนแปลงรูปแบบการสื่อสารตามเวลาเช่นการสื่อสารในรูปแบบตารางและแบบโครงสร้างต้นไม้ การสื่อสารแบบไม่มีโครงสร้างและสามารถเปลี่ยนแปลงได้จะใช้ในปัญหาเช่นการวิเคราะห์ไฟไนต์อีลีเมนต์ ซึ่งแสดงดังรูปที่ 5.9



รูปที่ 5.9 ตาราง (Grid) ซึ่งสร้างจากวิธีวิเคราะห์ไฟไนต์อีลีเมนต์

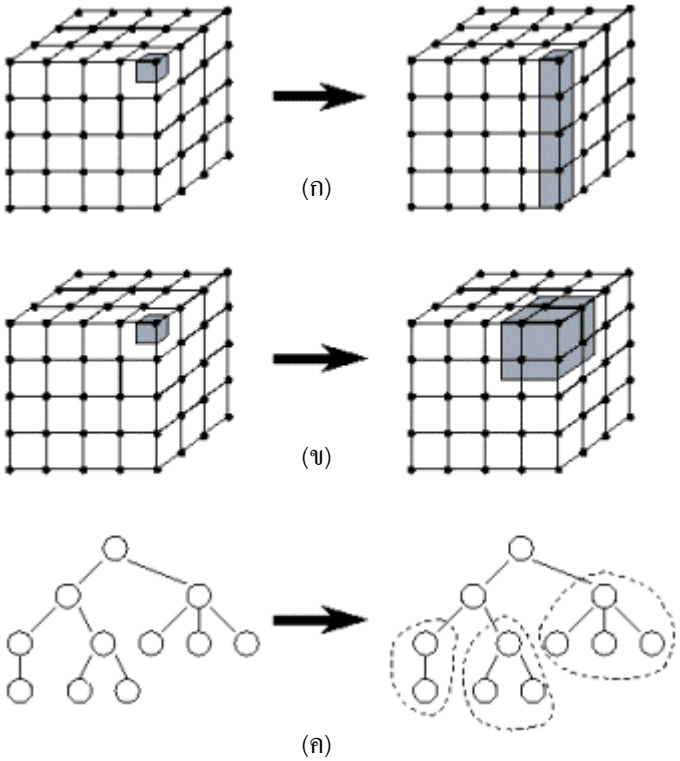
รูปร่างของตารางที่สร้างจากวิธีไฟไนต์อีลีเมนต์นี้จะไม่มีรูปแบบที่แน่นอนทำให้ไม่สามารถออกแบบการสื่อสารไว้ก่อนได้ วิธีแบ่งงานอาจทำได้โดยกำหนดให้แต่ละจุดต่อ (Vertex) ของเส้นตารางเป็นงานหนึ่งงาน และการสื่อสารจะทำตามจุดต่อตารางที่แบ่งไว้

5.3.2.3 การสื่อสารแบบประสานกันและการสื่อสารแบบไม่ประสานกัน (Synchronous and Asynchronous Communication)

การสื่อสารแบบประสานกัน ผู้ส่งและผู้รับรู้ว่าจะเกิดการสื่อสารกันก่อนจะเกิดการสื่อสารกันจริง ส่วนการสื่อสารแบบไม่ประสานกันผู้ส่งจะไม่รู้ว่าเมื่อใดผู้ระบจะต้องการข้อมูล ผู้รับข้อมูลจะต้องทำการร้องขอข้อมูลโดยตรงจากผู้ส่งข้อมูลเมื่อต้องการ

5.3.3 ขั้นตอนรวมกลุ่มงาน

ขั้นตอนรวมกลุ่มงานจะถูกทำเพื่อลดเวลาที่ใช้ในการสื่อสารระหว่างงานลงและทำให้หน่วยประมวลผล (Processors) สามารถประมวลผลงานได้มากขึ้น



รูปที่ 5.10 การรวมกลุ่มงานในการประมวลผลแบบขนานเพื่อลดเวลาในการสื่อสาร

การรวมกลุ่มงานอาจทำได้หลายลักษณะขึ้นอยู่กับประเภทของงานเช่นจากรูปที่ 5.10 ก. เป็นการรวมกลุ่มงานจากงานในตารางสามมิติเป็นงานในตารางสองมิติ ข.เป็นการรวมกลุ่มงานจาก

งานในตารางสามมิติเป็นงานที่มีกลุ่มใหญ่ขึ้นในสามมิติ ค. เป็นการรวมกลุ่มงานในรูปแบบแผนภูมิต้นไม้เป็นต้น

5.3.4 ขั้นตอนกำหนดงาน

การกำหนดงานจะถูกทำเป็นขั้นตอนสุดท้ายในการออกแบบอัลกอริทึมแบบขนาน ในคอมพิวเตอร์แบบขนาน การกำหนดงานจะต่างจากระบบคอมพิวเตอร์ที่มีหน่วยประมวลผลเดี่ยว (Uniprocessor) โดยระบบปฏิบัติการจะเป็นผู้กำหนดและจัดลำดับงาน (Scheduling) ให้กับหน่วยประมวลผลเพื่อทำการประมวลผล ในคอมพิวเตอร์แบบขนานการกำหนดงานอาจทำโดยอัตโนมัติจากระบบปฏิบัติการแบบขนานหรืออาจกำหนดจากผู้สร้างโปรแกรมโดยตรง การกำหนดงานเป็นขั้นตอนที่ซับซ้อนซึ่งวิธีการกำหนดงานที่ดีจะทำให้ใช้เวลาในการประมวลผลงานน้อยและมีความยืดหยุ่นในการใช้งาน หลักการในการกำหนดงานที่ดีมีสองข้อคือ

- กำหนดงานที่สามารถประมวลผลพร้อมๆกันได้ให้ทำงานในหน่วยประมวลผลต่างกัน
- กำหนดงานที่ต้องมีการสื่อสารกันบ่อยให้อยู่ในหน่วยประมวลผลเดียวกัน

การกำหนดงานอาจไม่มีรูปแบบที่แน่นอนขึ้นอยู่กับประเภทของงานและลักษณะคอมพิวเตอร์แบบขนาน ในระบบที่งานย่อยมีปริมาณงานไม่เท่ากันหรือระบบคอมพิวเตอร์แบบขนานเป็นแบบเนื้อผสม (Heterogeneous) ซึ่งความสามารถในการประมวลผลของแต่ละหน่วยประมวลผลอาจไม่เท่ากันจะทำให้วิธีการกำหนดงานมีความยากและซับซ้อนขึ้นไปอีก

5.4 กลวิธีการโปรแกรมแบบขนาน

การสร้างโปรแกรมแบบลำดับขั้น(Sequential Algorithm) บนเครื่องที่มีเพียงหน่วยประมวลผลเพียงเป็นเรื่องที่ไม่ยุ่งยากหรือสลับซับซ้อนมากมายนัก เพราะลำดับการทำงานของโปรแกรมเป็นไปอย่างตรงไปตรงมากับความต้องการหรือทฤษฎีและวิธีการที่อธิบายไว้แล้ว การทำงานของโปรแกรมใช้เพียงหน่วยประมวลผลเดียว การเข้าใช้งานหน่วยความจำก็สามาระที่เข้าถึงและใช้งานได้โดยไม่มีปัจจัยอย่างอื่นเข้ามาเกี่ยวข้อง การสื่อสารระหว่างโปรเซสที่ทำงานอยู่ก็มักจะไม่เกิดขึ้นในวิธีการการโปรแกรมแบบนี้ ทำให้การออกแบบโปรแกรมสามรถทำได้ง่าย ซึ่งแตกต่างจากกลวิธีการสร้างโปรแกรมแบบขนานเพราะปัจจัยในเรื่องการใช้งานหน่วยประมวลผล การเข้าใช้งานหน่วยความจำและการสื่อสารระหว่างโปรเซสจะเริ่มเข้ามามีบทบาทในการทำงานของโปรแกรมมากขึ้น อีกทั้งถ้าหากต้องการให้โปรกแกรมทำงานได้อย่างมีประสิทธิภาพแล้วต้องคำนึงถึงปัจจัยอื่น ๆ ที่มาเกี่ยวข้องด้วย

ดังนั้นในการสร้างโปรแกรมแบบขนานจึงได้มีผู้นำเสนอเทคนิคและวิธีการต่าง ๆ หลายอย่างในการสร้างโปรแกรมขึ้นมา เพื่อให้การทำงานของโปรแกรมเป็นไปอย่างเหมาะสมและมีประสิทธิภาพ Lester (1993:13) ได้นำเสนอวิธีการต่าง ๆ โดยรวมไว้ทั้งหมด 6 วิธีการดังนี้

- การรวมกลุ่มชุดข้อมูลที่มีความเหมือนกัน (Data Parallelism) ซึ่งเป็นวิธีการที่สามารถใช้งานได้โดยทั่วไป หลักการของการสร้างโปรแกรมแบบนี้ จะทำการรวมเอาข้อมูลอินพุท (Input) ที่มีความเหมือนกันเข้าไว้ด้วยกันเป็นกลุ่ม หลังจากนั้นทำการประมวลผลแจกเป็นกลุ่ม ๆ ไป เมื่อข้อมูลที่อย่าในกลุ่มเดียวกันมีความสัมพันธ์กันแล้ว ทำให้เกิดความง่ายในการคำนวณมากยิ่งขึ้น

- การแบ่งกลุ่มข้อมูลทีอยู่ใกล้กันได้ด้วยกัน (Data Partitioning) ข้อมูลอินพุทที่อยู่ใกล้กันจะมีความสัมพันธ์อยู่แล้วส่วนหนึ่ง ดังนั้นจึงสามารถที่จะรวมกลุ่มชุดของข้อมูลที่อยู่ใกล้กันไว้ด้วยกันได้ วิธีนี้ทำให้เกิดความสะดวกในการแบ่งกลุ่มของข้อมูลมากยิ่งขึ้น เพราะเพียงพิจารณาข้อมูลที่อยู่ติดกัน แจกเป็นส่วน ๆ เท่านั้น เมื่อทำการแยกข้อมูลได้แล้วก็ทำการประมวลผลข้อมูลกลุ่มต่าง ๆ ไปพร้อมกันตามหลักการของการประมวลผลแบบขนาน

วิธีการนี้เหมาะกับการคำนวณบนระบบประมวลผลแบบขนานแบบหลายเครื่องประมวลผลอย่างเช่นระบบคลัสเตอร์เป็นอย่างมากเพราะแต่ละเครื่องจะมีหน่วยความจำเป็นของตัวเองและเมื่อทำการคำนวณแต่ละเครื่องหรือแต่ละโหนดจะเกี่ยวข้องกับขุดข้อมูลที่ถูกแบ่งออกมาอย่างชัดเจน การเกี่ยวข้องกันกับชุดข้อมูลข้างเคียงมีน้อย ทำให้เกิดการแลกเปลี่ยนข้อมูลระหว่างโหนดน้อยลงตามไปด้าย มีผลทำให้เวลาที่สูญเสียไปในการสื่อสารน้อย เวลารวมในการทำงานก็ลดลงด้วย

- การทำงานแยกกันโดยอิสระ (Relaxed Algorithm) ถ้าหากแต่ละหน่วยประมวลผลในระบบสามารถทำงานแยกจากกันได้โดยอิสระ โดยที่ไม่ต้องรอข้อมูลผลลัพธ์จากหน่วยประมวลผลหรือโปรเซสข้างเคียงและไม่ต้องสื่อสารหรือตรวจทานความถูกต้องกับหน่วยประมวลผลหรือโปรเซสข้างเคียง และไม่ต้องสื่อสารหรือตรวจทานความถูกต้องกับหน่วยประมวลผลอื่น ๆ ทำให้ไม่ต้องสูญเสียเวลาในการสื่อสารหรือการรอในระหว่างการทำงานเลย หลักการนี้สามารถนำไปประยุกต์ใช้ได้ทั้งระบบแบบหลายหน่วยประมวลผลและหลายเครื่องประมวลผล

- การทำงานพร้อมกันในแต่ละรอบการทำงาน (Synchronous Iteration) ในการคำนวณบางประเภทอาจจะต้องมีการทำงานไปพร้อม ๆ กันในแต่ละรอบการทำงาน วิธีการนี้จะให้แต่ละโปรเซสทำการตรวจสอบการทำงานของโปรเซสข้างเคียงอื่น ๆ ในแต่ละรอบเพื่อให้คำตอบที่ได้มีค่าถูกต้อง การทำงานไปพร้อม ๆ กันจะลดความเร็วในการประมวลผลลงเล็กน้อย และเหมาะกับระบบที่เป็นแบบหลายหน่วยประมวลผลมากกว่าระบบที่เป็นแบบหลายเครื่องประมวลผล เพราะจะต้องมีการสื่อสารระหว่างกันค่อนข้างมาก

- การกระจายการทำงาน วิธีการนี้จะทำการรวมศูนย์งานไว้ที่เดียวเรียกว่า Work Pool เครื่องอื่น ๆ หรือโหนดอื่น ๆในระบบจะถูกมองว่าเป็นผู้ทำงานหรือ Worker ซึ่งผู้ทำงานนี้จะต้องเข้ามารับงาน

จาก Work Pool ไปทำจนกระทั้งงานที่อยู่ใน Pool หมดลง การทำงานแบบนี้อาจกล่าวง่าย ๆ ว่าเป็นการทำงานแบบรวมศูนย์เพราะทก Worker จะทำการติดต่อและร้องของานจากที่เดียวกัน

- การส่งต่อค่าผลลัพธ์ (Pipelined Computation) เทคนิคนี้ได้อาศัยรูปแบบของการส่งต่อค่าข้อมูลผลลัพธ์จากโปรเซสหนึ่ง ไปยังอีกโปรเซสหนึ่งในระบบเป็นลำดับเรื่อยไป ซึ่งการคำนวณงานบางชนิดต้องใช้รูปแบบการคำนวณงานแบบนี้ การส่งต่อค่าผลลัพธ์จะสามารถใช้ได้ดีทั้งระบบแบบหลายหน่วยประมวลผลและระบบแบบหลายเครื่องประมวลผล

เพื่อให้โปรแกรมทำงานอย่างมีประสิทธิภาพสูงสุด เมื่อจะทำการออกแบบโปรแกรมนั้น Lester (1993:15) ได้เสนอึงปัญหาต่าง ๆ ที่ควรคำนึง รวมทั้งปัญหาที่ควารหลีกเลี่ยงไม่หเกิดขึ้นในโปรแกรม ปัญหาสำคัญ ๆ มีดังนี้

- ความคับคั่งในการเข้าใช้งานหน่วยความจำ (Memory Contention) ปัญหานี้เกิดจาการเข้าใช้ตัวแปรร่วม (Global variable) ในหน่วยความจำพร้อมกันของหน่วยประมวลผล เมื่อหน่วยประมวลผลหนึ่งใช้งานหน่วยความจำอยู่ หน่วยประมวลผลอื่น ๆ จะไม่สามารถเข้าใช้งานได้ทำให้เกิดการหยุดรอของหน่วยประมวลผลขณะที่กำลังคำนวณ ซึ่งจะทำให้สูญเสียเวลาไปกับการรอนี้ แต่ปัญหานี้มักจะเกิดในระบบหลายหน่วยประมวลผลที่ใช้หน่วยความจำร่วมมากกาว่าส่วนระบบแบบหลายเครื่องประมวลผลมักไม่เป็นปัญหา

- การมีส่วนของโปรแกรมที่ทำงานแบบลำดับมากเกินไป (Excessive Sequential Code) ซึ่งอาจจะเกิดจากตอนที่ออกแบบโปรแกรมนั้นผู้เขียนโปรแกรมอาจจะสร้างโปรแกรมขึ้นมาจากวิธีการแบบลำดับ และยังใช้วิธีการคิดแบบเดิมในการโปรแกรมแบบขนานอยู่ทำให้โปรแกรมที่ถูกสร้างขึ้นไม่ได้ทำงานไปอย่างพร้อม ๆ กันอย่างแท้จริง และประสิทธิภาพของโปรแกรมก็ลดลง

- การสูญเสียเวลาไปในการสร้างโปรเซส (Process Creation Time) ในการทำงานแบบขนานจะต้องมีเวลาส่วนหนึ่งที่สูญเสียไปเพราะการสร้างโปรเซสขึ้นมาใหม่ เพราะฉะนั้นการสร้างโปรเซสควารสร้างขึ้นมาเท่าที่จำเป็นเท่านั้น เพราะบางทีการสร้างโปรเซสขึ้นมามากเกินไปก็ไม่ได้ช่วยทำให้การทำงานของโปรแกรมเร็วขึ้นแต่อย่างใด

- การสูญเสียเวลาในการสื่อสาร (Communication Delay) ปัญหานี้จะเกิดขึ้นกับระบบแบบหลายเครื่องประมวลผลค่อนข้างมาก เพราะแต่ละหน่วยประมวลผลจะต้องทำการติดต่อกันผ่านทางการส่งข้อความข้ามไปมาระหว่างโปรเซสที่ทำงานอยู่ในระบบ ดังนั้นเมื่อทำการออกแบบโปรแกรมแล้วต้องให้มีการสื่อสารระหว่าโปรเซสน้อยที่สุด และขนาดของข้อมูลที่รับและส่งนั้นควารมีขนาดเล็กมากที่สุดด้วยเพื่อให้ เสียเวลาในการสื่อสารน้อยที่สุดนั่นเอง

- การสูญเสียเวลาในการตรวจทางสถานะ (Synchronization Delay) เมื่อมีการออกแบบโปรแกรมให้แต่ละโปรเซสต้องทำการตรวจทางสถานะซึ่งกันและกันแล้ว จะต้องมีเวลาส่วนหนึ่งสูญเสียไปเพื่อ

การนี้ ดังนั้น ทางที่ดีถ้าหากลดการตรวจทานสถานะไปได้หรือไม่มีเลยก็จะทำให้โปรแกรมที่ได้มีประสิทธิภาพดีขึ้นด้วย

- การกระจายงานไม่สมดุล หรือไม่เหมาะสม (Load Imbalance) เกิดจากการที่มีวิธีการกระจายการทำงานไม่เหมาะสม ทำให้มีหน่วยประมวลผลบางส่วนเท่านั้นที่มีโอกาสได้ทำงาน ส่วนหน่วยประมวลผลอื่นอาจจะอยู่ในสถานะรอเพียงอย่างเดียว ดังนั้นเพื่อแก้ปัญหานี้จะต้องมีวิธีการที่เหมาะสมในการกระจายงานไปยังหน่วยประมวลผลในระบบ